UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DAVID LOPES DE MACÊDO

TOWARDS ROBUST DEEP LEARNING USING ENTROPIC LOSSES

Recife

2022

DAVID LOPES DE MACÊDO

TOWARDS ROBUST DEEP LEARNING USING ENTROPIC LOSSES

A Ph.D. Thesis presented to the Center of Informatics of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Concentration Area: Computational Intelligence
Advisor: Teresa Bernarda Ludermir
Co-Advisor: Cleber Zanchettin

Recife
2022

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

M141t Macêdo, David Lopes de
Towards robust deep learning using entropic losses / David Lopes de Macêdo. – 2022.
173 f.: il., fig., tab.

Orientadora: Teresa Bernarda Ludermir.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
.
Inclui referências e apêndices.

1. Inteligência computacional. 2. Aprendizado profundo. I. Ludermir, Teresa Bernarda (orientadora). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2022-114

David Lopes de Macêdo

“Towards Robust Deep Learning using Entropic Losses”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Inteligência Computacional

Aprovado em: 27/06/2022.

Orientadora: Profa. Dra. Teresa Bernarda Ludermir

To my family.

ACKNOWLEDGEMENTS

This work would not have been possible without the support of many. I thank and dedicate this doctoral thesis to the following people:

I thank my advisor, Teresa Ludermir. Teresa is an exceptional researcher and professor, and her support was fundamental to motivating me throughout this research. I also thank my co-advisor, Cleber Zanchettin, for reviewing this work.

There is no doubt they contributed fundamentally to broadening my knowledge[1] and to allowing me to become a Top Conferences (NeurIPS, ICLR, ICML) and IEEE Reviewer. Moreover, the work we did together led me to be a Member of the Program Committees of the NeurIPS Workshop on Machine Learning Safety and the ICML Workshop on Uncertainty & Robustness in Deep Learning, in addition to an ICLR Highlighted Reviewer[2].

I thank Valdemar Cardoso da Rocha Junior, responsible for introducing me to the wonderful world of scientific research in the 1990s. I also thank him for provoking me about the power of Information Theory, which undoubtedly contributed decisively to this work.

To Yoshua Bengio for the personal discussions we had regarding the out-of-distribution detection problem in deep learning during my stay at the Montreal Institute for Learning Algorithms (MILA) as Visiting Researcher[3]. The first question he asked was: "Why does it work?"

To Adriano Oliveira, Cleber Zanchettin, and the alumni of the Deep Learning Course of the CIn/UFPE Computer Science Graduate Program for the extremely pleasant discussions that we have had and all interesting work that we have done together since 2016 when we started this rewarding journey[4,5].

To my family, especially my parents, José and Mary, my wife, Janaina, and my children, Jéssica and Daniel, for giving me love throughout my life.

[1]https://dlmacedo.com
[2]https://iclr.cc/Conferences/2022/Reviewers
[3]https://mila.quebec/en/person/david-macedo
[4]https://deeplearning.cin.ufpe.br
[5]https://dlmacedo.com/courses/deeplearning

ABSTRACT

Despite theoretical achievements and encouraging practical results, deep learning still presents limitations in many areas, such as reasoning, causal inference, interpretability, and explainability. From an application point of view, one of the most impactful restrictions is related to the robustness of these systems. Indeed, current deep learning solutions are well known for not informing whether they can reliably classify an example during inference. Modern neural networks are usually overconfident, even when they are wrong. Therefore, building robust deep learning applications is currently a cutting-edge research topic in computer vision, natural language processing, and many other areas. One of the most effective ways to build more reliable deep learning solutions is to improve their performance in the so-called out-of-distribution detection task, which essentially consists of "know that you do not know" or "know the unknown". In other words, out-of-distribution detection capable systems may reject performing a nonsense classification when submitted to instances of classes on which the neural network was not trained. This thesis tackles the defiant out-of-distribution detection task by proposing novel loss functions and detection scores. Uncertainty estimation is also a crucial auxiliary task in building more robust deep learning systems. Therefore, we also deal with this robustness-related task, which evaluates how realistic the probabilities presented by the deep neural network are. To demonstrate the effectiveness of our approach, in addition to a substantial set of experiments, which includes state-of-the-art results, we use arguments based on the principle of maximum entropy to establish the theoretical foundation of the proposed approaches. Unlike most current methods, our losses and scores are seamless and principled solutions that produce accurate predictions in addition to fast and efficient inference. Moreover, our approaches can be incorporated into current and future projects simply by replacing the loss used to train the deep neural network and computing a rapid score for detection.

Keywords: deep learning robustness; out-of-distribution detection; uncertainty estimation; isotropy maximization loss; enhanced isotropy maximization loss; distinction maximization loss.

RESUMO

Apesar das conquistas teóricas e resultados práticos encorajadores, o aprendizado profundo ainda apresenta limitações em muitas áreas, como raciocínio, inferência causal, interpretabilidade e explicabilidade. Do ponto de vista da aplicação, uma das restrições mais impactantes está relacionada à robustez desses sistemas. De fato, as soluções atuais de aprendizado profundo são bem conhecidas por não informar se podem classificar um exemplo de maneira confiável durante a inferência. As redes neurais modernas geralmente são superconfiantes, mesmo quando estão erradas. Portanto, construir aplicativos robustos de aprendizado profundo é atualmente um tópico de pesquisa de ponta em visão computacional, processamento de linguagem natural e muitas outras áreas. Uma das maneiras mais eficazes de construir soluções de aprendizado profundo mais confiáveis é melhorar seu desempenho na chamada tarefa de detecção fora de distribuição, que consiste essencialmente em “saber que você não sabe” ou “conhecer o desconhecido”. Em outras palavras, sistemas com capacidade de detecção fora de distribuição podem rejeitar a realização de uma classificação sem sentido quando submetidos a instâncias de classes nas quais a rede neural não foi treinada. Esta tese aborda a desafiadora tarefa de detecção fora da distribuição, propondo novas funções de perda e pontuações de detecção. A estimativa de incerteza também é uma tarefa auxiliar crucial na construção de sistemas de aprendizado profundo mais robustos. Portanto, tratamos também dessa tarefa relacionada à robustez, que avalia quão realistas são as probabilidades apresentadas pela rede neural profunda. Para demonstrar a eficácia de nossa abordagem, além de um conjunto substancial de experimentos, que incluí resultados estado-da-arte, utilizamos argumentos baseados no princípio da máxima entropia para estabelecer a fundamentação teórica das abordagens propostas. Ao contrário da maioria dos métodos atuais, além de apresentarem inferência rápida e eficiente, nossas perdas e pontuações são soluções baseadas em princípios e não produzem efeitos colaterais indesejados. Além disso, nossas abordagens podem ser incorporadas em projetos atuais e futuros simplesmente substituindo a perda usada para treinar a rede neural profunda e computando uma pontuação rápida para detecção.

Palavras-chave: aprendizado profundo robusto; detecção de fora da distribuição; estimação de incerteza; perda de maximização de isotropia; perda de maximização de isotropia melhorada; perda de maximização de distinção.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACET	Adversarial Confidence Enhancing Training
AUPR	Area Under the Precision-Recall Curve
AUROC	Area Under the Receiver Operating Characteristic Curve
DisMax	Distinction Maximization Loss
DTACC	Detection Accuracy
DUQ	Deterministic Uncertainty Quantification
ECE	Expected Calibration Error
FPR	Fractional Probability Regularization
GODIN	Generalized Out-of-DIstribution Detector for Neural networks
ID	In-Distribution
IsoMax	Isotropy Maximization Loss
IsoMax+	Enhanced Isotropy Maximization Loss
MDS	Minimum Distance Score
MMLES	Maximum-Mean Logit Entropy Score
ODIN	Out-of-DIstribution Detector for Neural networks
OOD	Out-of-Distribution
SC	Scaled Cosine
SNGP	Spectral-Normalized Neural Gaussian Process
TNR@TPR95	True Negative Rate at 95% True Positive Rate

CONTENTS

1 INTRODUCTION

"All models are wrong, but some are useful."

–George Box

"All models are wrong, but some that know when they are wrong are useful."

–Balaji Lakshminarayanan (NeurIPS 2020)

"One of the greatest challenges in this world is knowing enough a subject to think you are right, but not enough to know you are wrong!"

–Neil deGrasse Tyson

"Modern deep neural networks are just like humans. They are usually overconfident, even when they are entirely wrong!"

–David Macêdo

In this introductory chapter, we explain the context, motivation, and objective of this work. Then we describe the concept of Out-of-Distribution (OOD) detection, our specific problem of interest in the sub-area of deep learning robustness. Next, we define the objectives of this study. Among them is the design of high-performance solutions for OOD detection. Moreover, we present the peer review publications we authored in this and related research fields. Finally, we present an outline of the subject of the following chapters.

1.1 CONTEXT: WHY DEEP LEARNING?

AI is a field in computer science. Nowadays, it is hard to present an unique definition on which everybody could agree on. We understand AI as the study of systems capable of solving problems that are not handled well by using algorithm-based programming paradigms. It usually involves tasks for which a standard algorithmic solution is hard or impractical to design, implement, or even conceive. Otherwise, we could simply use a straightforward conventional so-called algorithmic-based programming approach to tackle the mentioned task.

This AI concept is strongly based on the phase "field of study that gives computers the ability to learn without being explicitly programmed" commonly attributed to Arthur Samuel. Although he probably never explicitly wrote or said the mentioned quotation, it works as a good abstract of thoughts he expressed in his seminal papers (Samuel, 1959, 1967).

Machine learning is the field of AI in which the system learns from exploring the available data rather than relying on humans to algorithmize them. Machine learning established a significant advance by allowing AI systems to acquire their knowledge directly from data.

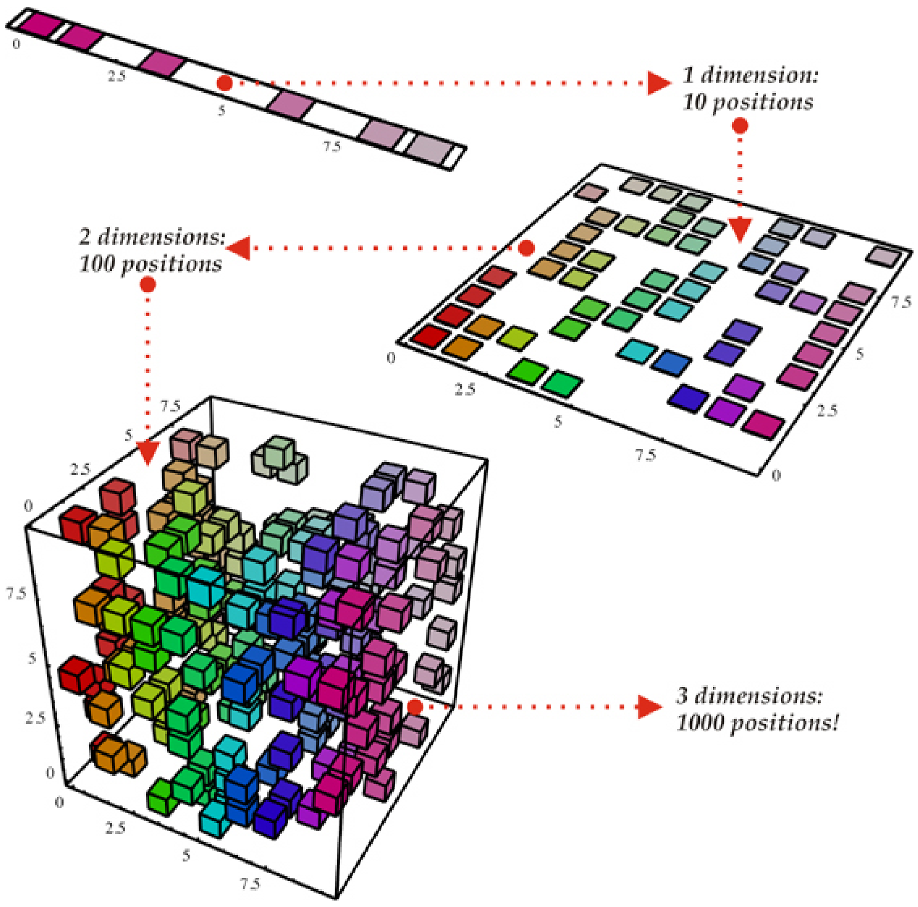
Nevertheless, the conventional machine learning approach to AI has its limitations. Indeed, in this context, the curse of dimensionality (Bellman & Karush, 1964; Bellman & Collection, 1961; Hughes, 1968; Bengio *et al.*, 2005) may be understood as the empirical tendency to a dramatic training examples density rarefaction as the data dimensionality grows (Fig. 1). In such cases, at least apparently, it is reasonable to expect that the extremely low training example density could make generalization very difficult, as the test examples would be extremely distant from the examples used for training. Hence, at first sight, it appears that the number of training examples should grow exponentially faster than the data dimensionality to allow generalization.

To mitigate the curse of dimensionality, machine learning methods generally rely on hand-designed high-level features, which allow such systems to sometimes produce high performance and scale appropriately (Goodfellow *et al.*, 2016, pg. 3). These engineered features are presented to the learning algorithm as low-dimensional vectors that, in some sense, bring the problem back to a treatable dimension, somehow avoiding extreme example density rarefaction.

Indeed, Table 1 presents datasets and their more relevant characteristics. We can see that the example density is reasonably high in datasets with few features (three first lines). Therefore, we may tackle these problems practically without relying on feature extraction or selection. However, for datasets with a high number of features (three last lines), we usually use feature engineering to tackle them to obtain high performance. In simple terms, the use of features brings the problem back to a dimensional that classical machine learning approaches can handle appropriately.

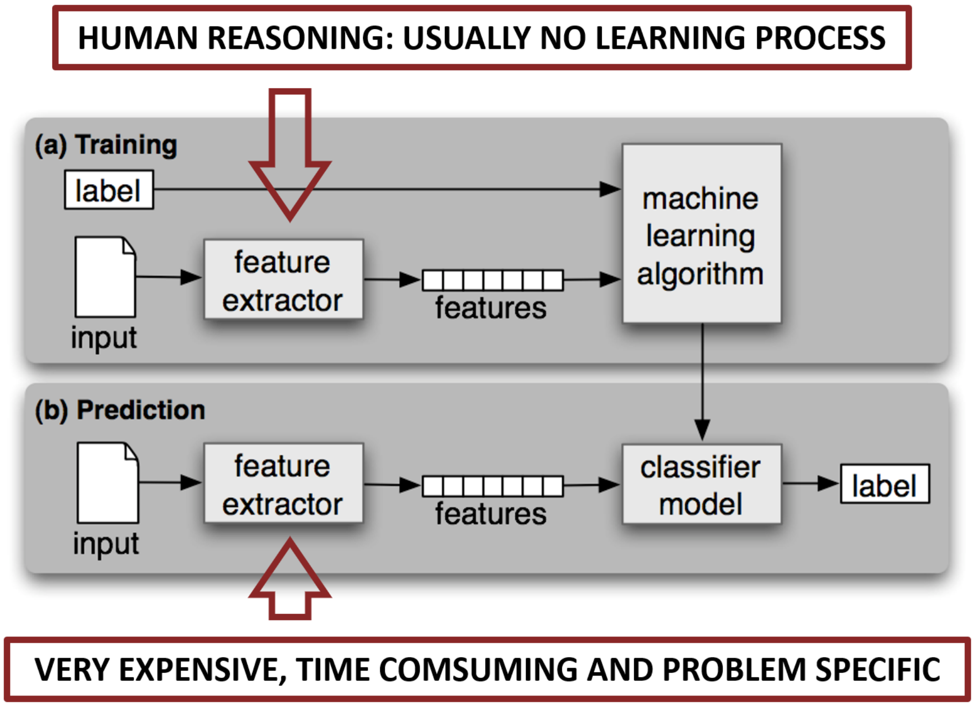
However, relying on hand-designed high-level features brings significant drawbacks to systems based on traditional shallow machine learning. In fact, it is undesirable for many reasons. First, as a specific feature engineering is usually needed for each particular task, the overall solution becomes much less general-purpose than it could otherwise be.

Figure 1 - The Curse of Dimensionality



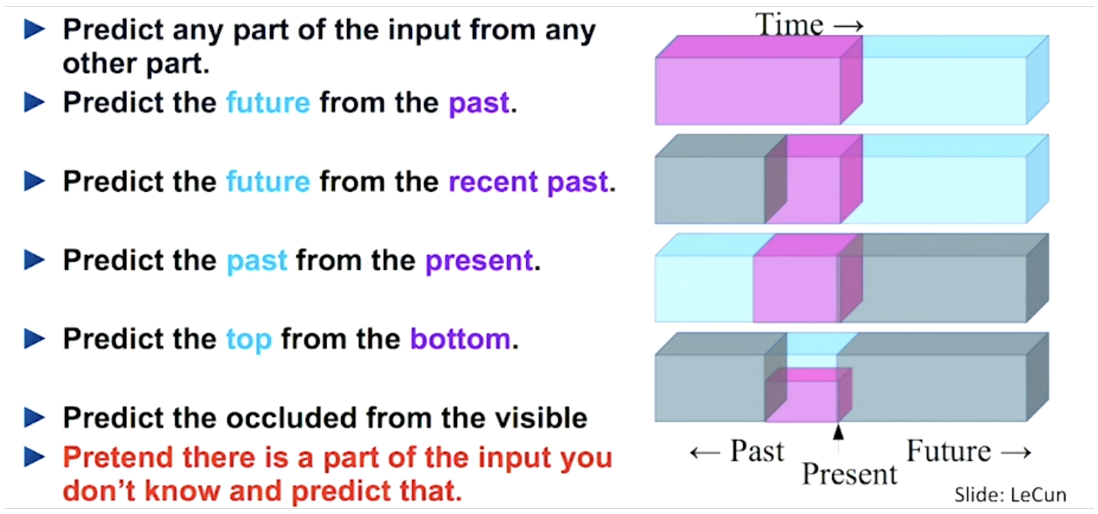
Source: Bengio (2015). The curse of dimensionality: As the dimension grows, we need exponentially more training examples to avoid example density rarefaction in the feature space.

Figure 2 - The Fundamental Shallow Machine Learning Limitation


HUMAN REASONING: USUALLY NO LEARNING PROCESS
(a) Training
label
input
feature extractor
features
machine learning algorithm
(b) Prediction
input
feature extractor
features
classifier model
label
VERY EXPENSIVE, TIME COMSUMING AND PROBLEM SPECIFIC

Source: Adapted from https://www.nltk.org/book/ch06.html. The shallow machine learning limitation: Numerous relevant real-world problems are high-dimensional. In such cases, the hand-designed feature engineering usually required by classical machine learning approaches presents drawbacks.

Figure 3 - Self-Supervised Learning



- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don't know and predict that.

Time →
← Past
Present
Future →
Slide: LeCun

Source: https://www.youtube.com/watch?v=7I0Qt7GALVk. Self-supervision allows training deep neural networks without labels using a pretext task. Given a portion of the data, the model is trained to predict what is missing.

Table 1 - Example Density of Traditional Datasets

Dataset	Examples	Features	Volumes (2^{Features})	Example Density (Examples/Volumes)
Iris	150	4	16	9.375
Tic-Tac-Toe	958	9	512	1.871
Adult	48,842	14	16,384	2.981
MNIST	60,000	784 (28x28)	1.017×10^{236}	5.897×10^{-232}
SVHN	600,000	3,072 (3x32x32)	5.809×10^{924}	1.032×10^{-919}
ImageNet	1,200,000	196,608 (3x256x256)	2.003×10^{19728}	5.991×10^{-19723}

Source: The Author (2022). In this table, the first three lines present examples of datasets in which classical machine learning works properly without relying on feature engineering. The last three lines give examples of datasets in which deep learning works appropriately.

Second, feature engineering is commonly a costly and time-consuming process that requires human knowledge much more specialized than labeling data. Finally, for many real-world relevant tasks, it is challenging (if not impossible) to design high-quality features manually.

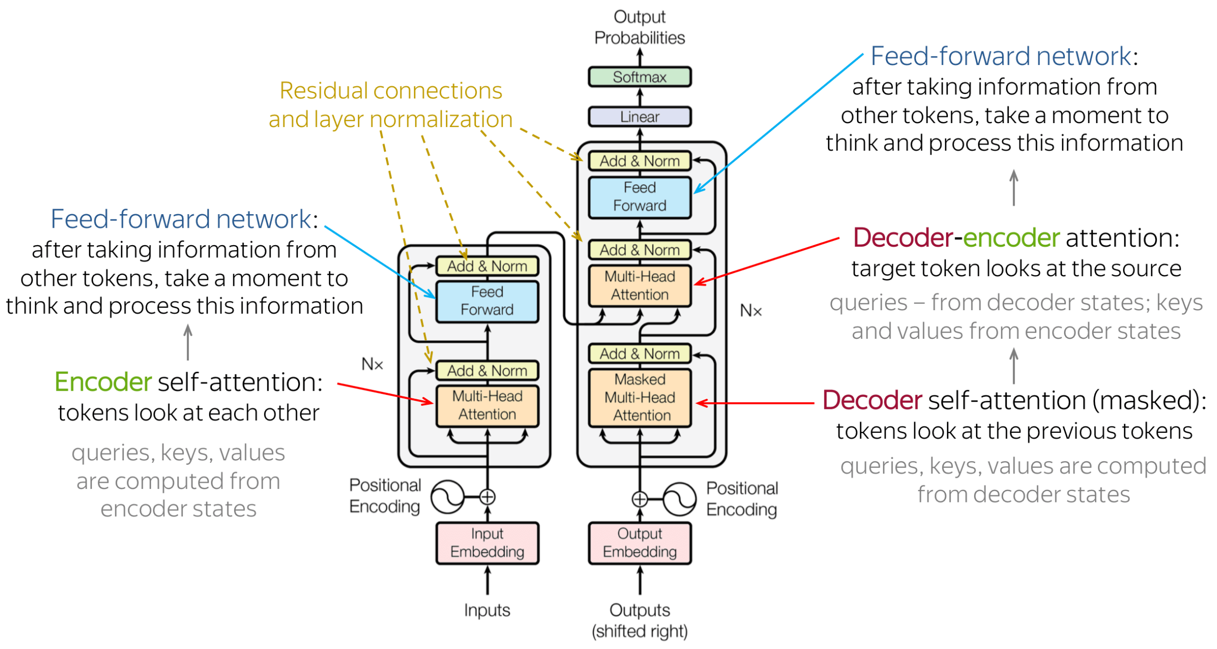
Recently, theoretical studies have shown that increasing the depth of AI models is an extremely effective way to deal with the curse of dimensionality (Bengio & Lecun, 2007; Delalleau & Bengio, 2011; Pascanu *et al.*, 2014; Montúfar *et al.*, 2014). Indeed, some research results suggest that growing the depth of a learning model is exponentially more efficient than increasing the width of traditional shallow machine learning solutions (Bengio *et al.*, 2013, 2005; Bengio, 2009; Montúfar & Morton, 2015; Montúfar *et al.*, 2014).

Therefore, the recognition that deep architectures are exponentially efficient in tackling the curse of dimensionality made deep neural networks extremely successful, as they virtually altogether avoid time-consuming, specific purpose, and expensive feature engineering (Fig. 2). Moreover, modern self-supervised[1] techniques are making even labeling data unnecessary (Le-Khac *et al.*, 2020; Jaiswal *et al.*, 2020; Jing & Tian, 2021; Liu *et al.*, 2020c) (Fig. 3).

Table 1 presents datasets on which shallow machine learning (first three lines) and deep learning (last three lines) are usually applied *without* relying on feature engineering. The difference in the density of examples in each case is staggering. The fact that deep learning allows satisfactory generalization in datasets with such small example density as ImageNet (Deng *et al.*, 2009) without using feature engineering is extraordinary. Indeed, it was something unthinkable years ago.

[1]https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence

Figure 4 - Transformer Architecture


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Feed Forward
Multi-Head Attention
Nx
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Residual connections and layer normalization
Feed-forward network: after taking information from other tokens, take a moment to think and process this information
Feed-forward network: after taking information from other tokens, take a moment to think and process this information
Encoder self-attention: tokens look at each other
queries, keys, values are computed from encoder states
Decoder-encoder attention: target token looks at the source
queries – from decoder states; keys and values from encoder states
Decoder self-attention (masked): tokens look at the previous tokens
queries, keys, values are computed from decoder states

Source: http://www.ai2news.com/blog/42747/. The Transformer understands everything as a mapping from an input set (or sequence) of tokens to an output set (or sequence) of tokens. All input need to be converted to tokens to be processed by the Transformer-based architectures (Vaswani *et al.*, 2017). It was highly inspired by the seminal Sequence-To-Sequence paper (Sutskever *et al.*, 2014).

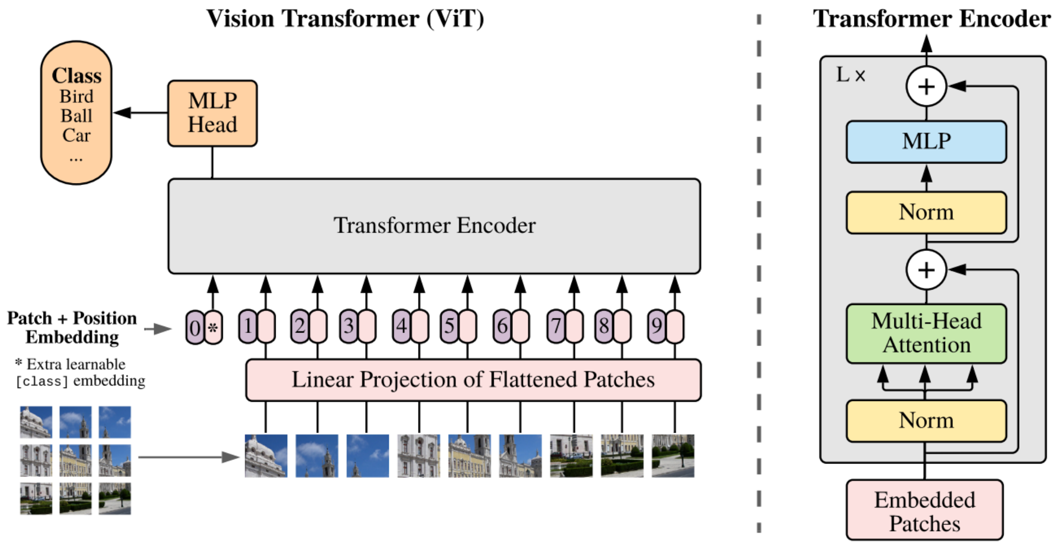
After the breakthroughs in speech recognition (Hinton *et al.*, 2012) and computer vision (Krizhevsky *et al.*, 2012), other important advances followed in natural language processing (Sutskever *et al.*, 2014; Bahdanau *et al.*, 2015), speech processing (Alam *et al.*, 2020), and even in structured data such as tabular data (Borisov *et al.*, 2021) and time series (Lim & Zohren, 2020).

Currently, deep learning is applied to solve tasks in a broad area of applications such as mathematics, physics, chemistry, engineering, biology, health care, finance, agriculture, and many others (Goodfellow *et al.*, 2016, pg. 9) (Lemos *et al.*, 2022). The advances in the use of deep learning for scientific discoveries[2] (Deiana *et al.*, 2021) are so profound that some argue that we are witnessing a new scientific revolution[3].

Transforms[4,5] (Vaswani *et al.*, 2017) allowed us to use a single universal architecture to construct image, text, video, audio, and multimodal models (Fig. 4). After BERT (Devlin *et al.*, 2019) and GPT (Brown *et al.*, 2020) successfully applied the groundbreaking Transformer architecture for text, the so-called Vision Transformers (ViT) (Dosovitskiy *et al.*, 2021) (Fig. 5) effectively applied this template architecture for images.

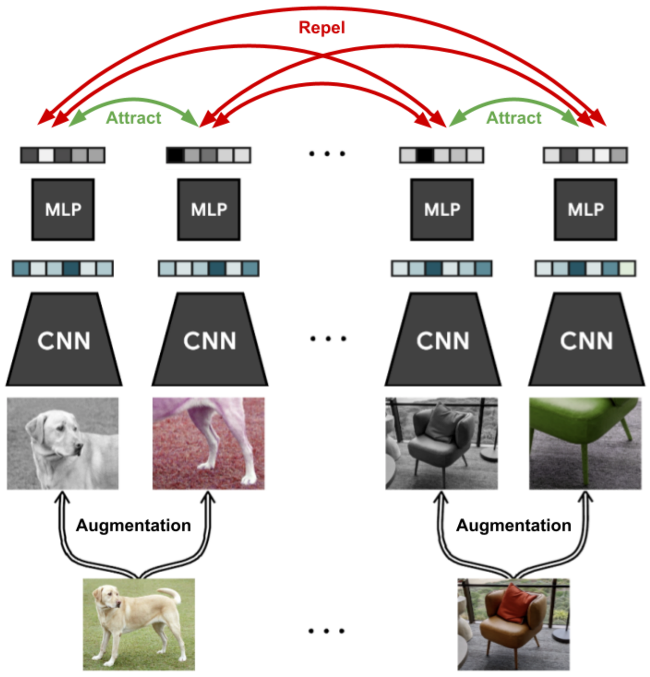
[2]https://www.youtube.com/watch?v=XtJVLOe4cfs
[3]https://thegradient.pub/ai-scientific-revolution/
[4]https://jalammar.github.io/illustrated-transformer/
[5]https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Figure 5 - Vision Transformer (ViT)



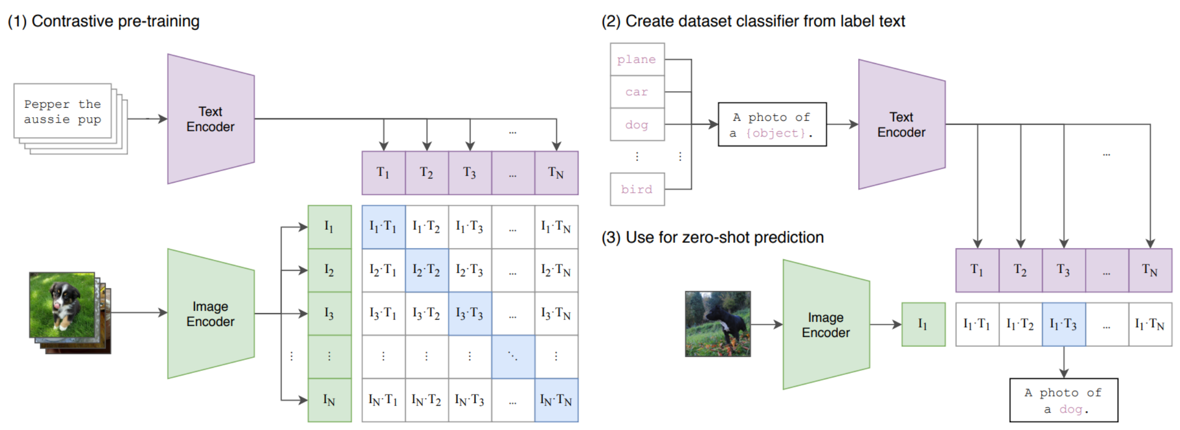
Source: Dosovitskiy *et al.* (2021). The Vision Transformer (ViT) creates tokens by splitting the images into patches, which then feed the Transformer architecture as proposed in Vaswani *et al.* (2017). The classification token is inspired by the BERT paper (Devlin *et al.*, 2019).

Figure 6 - Contrastive Learning



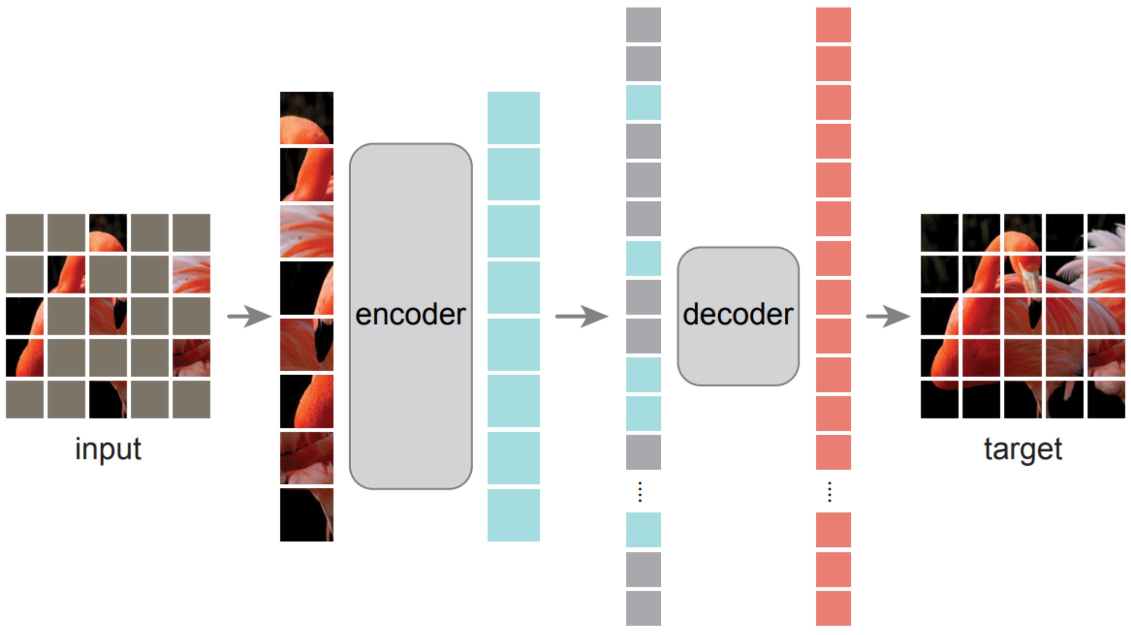
Source: Chen *et al.* (2020a). Contrastive learning allows supervised or unsupervised training of deep neural networks. Positive examples (i.e., examples we desire to approximate in the feature space) may be constructed using only data augmentation without any supervision.

Figure 7 - CLIP Architecture


(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ... I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
(3) Use for zero-shot prediction
Image Encoder
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
A photo of a dog.

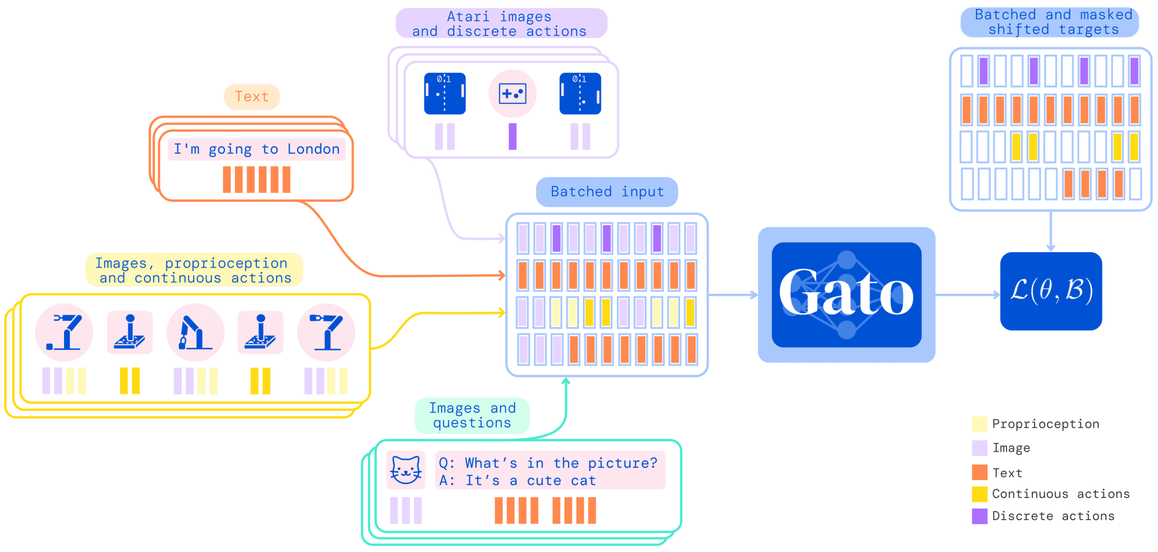
Source: Radford *et al.* (2021). OpenAI CLIP uses supervised contrastive learning to approximate embeddings of images and related texts in a joint embedding space. Zero-shot classification is allowed during inference.

Figure 8 - Masked Autoencoder (MAE)



Source: He *et al.* (2021). In the Masked Autoencoder (MAE), about 75% of the image patches are hidden from the decoder during the self-supervised pretraining. The decoder is trained with patches produced by the encoder and mask tokens and needs to reconstruct the original image.

Figure 9 - Generalist Agent (Gato)


Atari images and discrete actions
Batched and masked shifted targets
Text
I'm going to London
Batched input
Images, proprioception and continuous actions
Gato
$\mathcal{L}(\theta, \mathcal{B})$
Images and questions
Q: What's in the picture?
A: It's a cute cat
Proprioception
Image
Text
Continuous actions
Discrete actions

Source: Reed *et al.* (2022). DeepMind Generalist Agent (Gato) can simultaneously handle many media and tasks using a unified Transformer-based architecture.

Contrastive learning[6] essential idea consists of maximizing similarity (i.e. minimizing the distance) between the (high-level) representation of input data that we desire or know to be similar, while doing the opposite when we desire the input data to be semantically different (Chen *et al.*, 2020a; Albelwi, 2022) (Fig. 6).

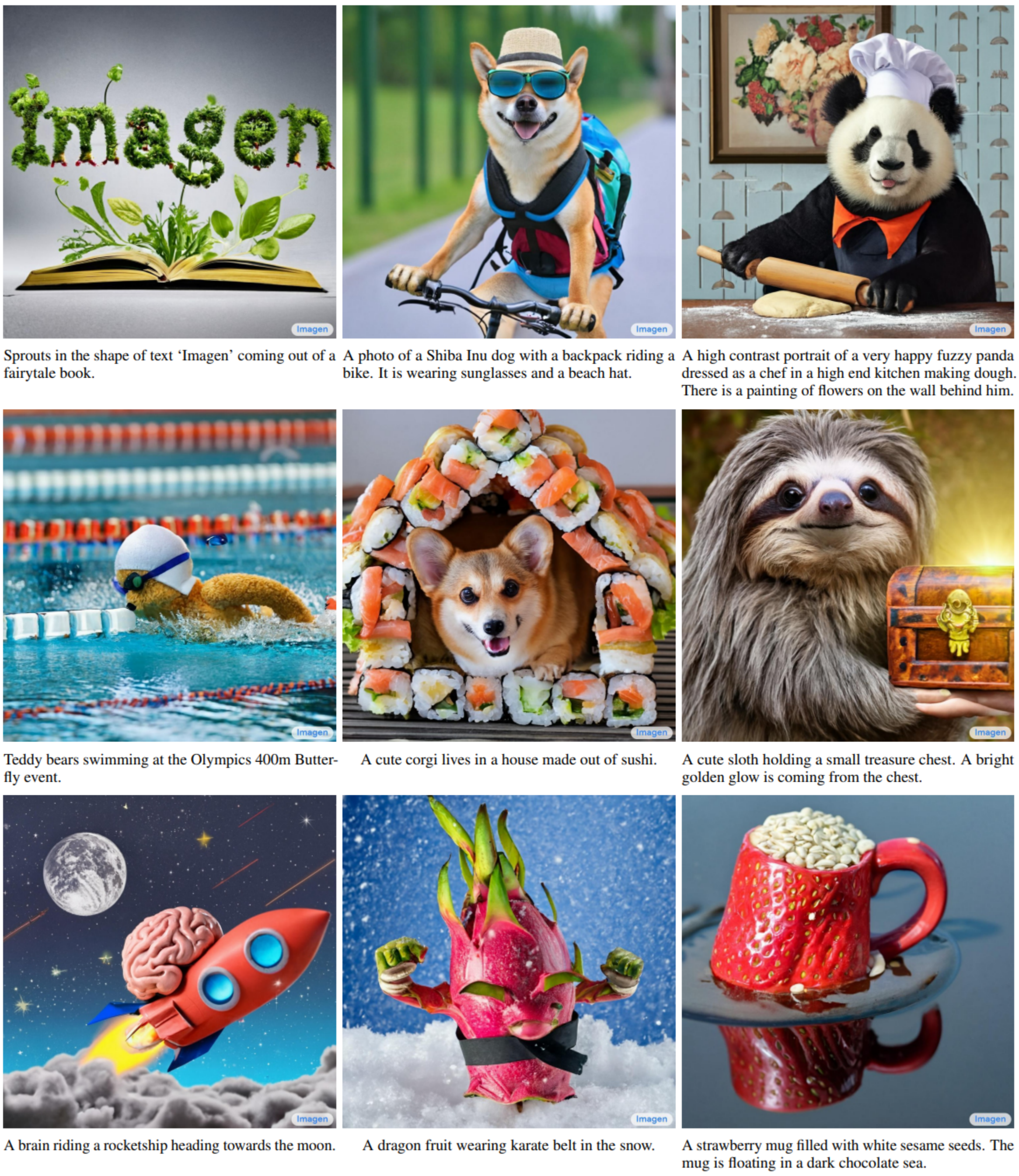
Notice that contrastive learning can be used in conjunction with data augmentation to create input data that we want to produce similar representations, which allows unsupervised training. Therefore, contrastive learning can be used in both supervised and unsupervised settings. Self-supervised learning may or not be used in combination with contrastive learning.

CLIP (Radford *et al.*, 2021) from OpenAI leverage supervised contrastive learning to build a vision-language model that constructs a *joint embedding space* in which representations of images are near to representations of related texts. CLIP[7] allows zero-shot classification during inference and is a relevant example of supervised contrastive learning (Fig. 7).

Self-supervision using masking and autoregressive techniques are making possible pretraining without labels large models that subsequently can be downloaded and fine-tuned with few examples of supervised data. Sometimes, they may even be used in zero-shot or few-shot settings. Currently, self-supervision is largely used regardless of media type (image, speech, text, etc.). For example, while BERT and MAE (He *et al.*, 2021) (Fig. 8) used a masking pretext, the GPT used an autoregressive pretext task. Currently, some researches are exploring single models (Reed *et al.*, 2022) that are both multimodal and multitask[8] (Fig. 9).

[6]https://ai.googleblog.com/2020/04/advancing-self-supervised-and-semi.html
[7]https://openai.com/blog/clip/
[8]https://www.deepmind.com/publications/a-generalist-agent

Figure 10 - Imagen: Photorealistic Examples



Sprouts in the shape of text 'Imagen' coming out of a fairytale book.

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

A high contrast portrait of a very happy fuzzy panda dressed as a chef in a high end kitchen making dough. There is a painting of flowers on the wall behind him.

Teddy bears swimming at the Olympics 400m Butterfly event.

A cute corgi lives in a house made out of sushi.

A cute sloth holding a small treasure chest. A bright golden glow is coming from the chest.

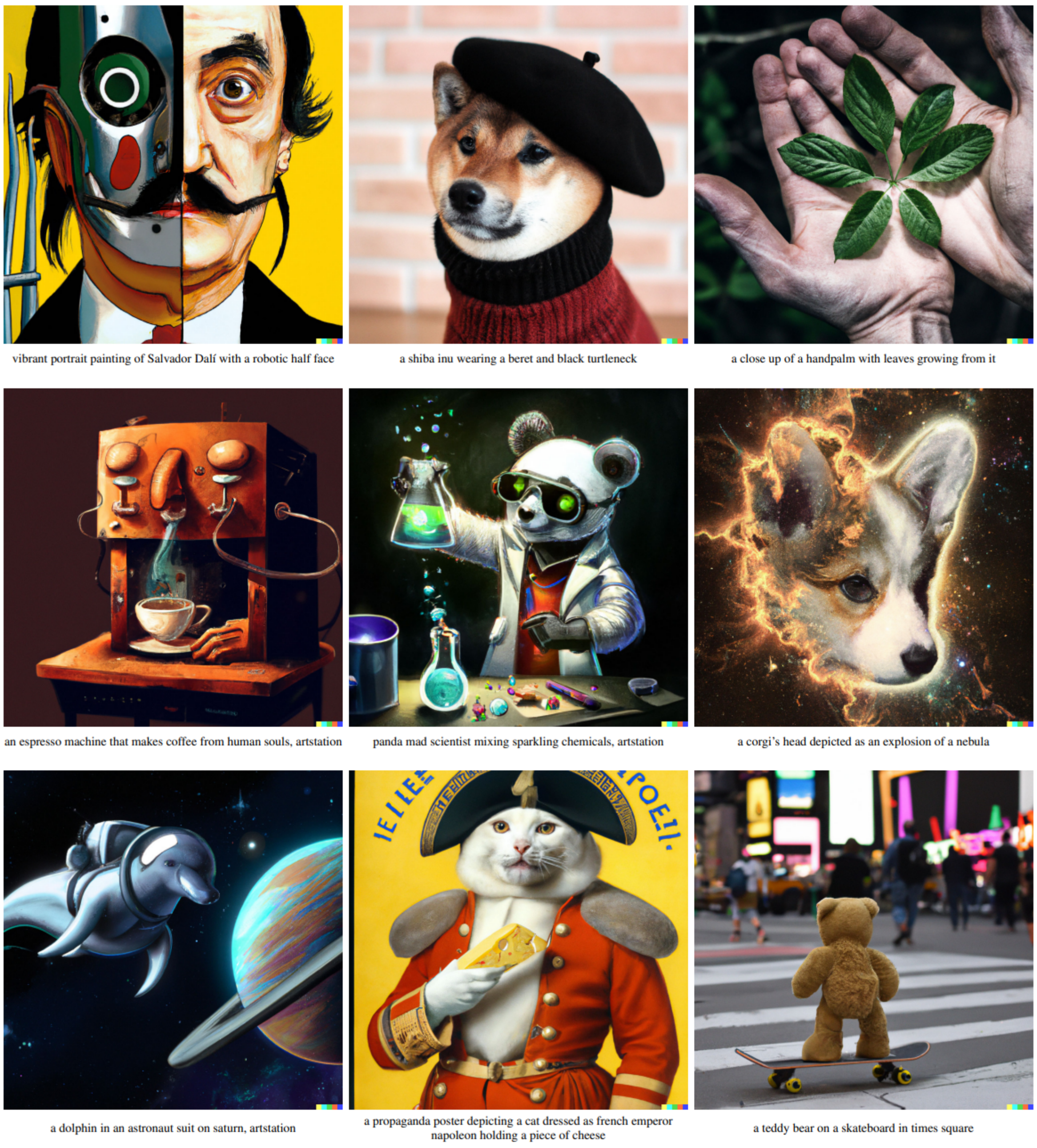
A brain riding a rocketship heading towards the moon.

A dragon fruit wearing karate belt in the snow.

A strawberry mug filled with white sesame seeds. The mug is floating in a dark chocolate sea.

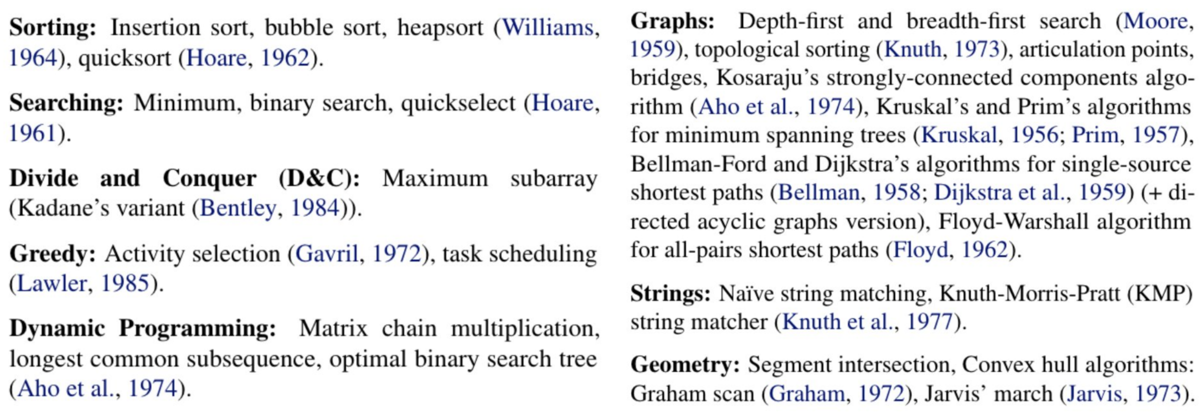
Source: Saharia *et al.* (2022). Google Imagen: Examples of photorealistic 1024x1024 high-resolution images produced conditioned on texts. It is also possible to generate artistic content.

Figure 11 - DALL-E 2: Photorealistic Examples


vibrant portrait painting of Salvador Dalí with a robotic half face
a shiba inu wearing a beret and black turtleneck
a close up of a handpalm with leaves growing from it
an espresso machine that makes coffee from human souls, artstation
panda mad scientist mixing sparkling chemicals, artstation
a corgi's head depicted as an explosion of a nebula
a dolphin in an astronaut suit on saturn, artstation
a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese
a teddy bear on a skateboard in times square

Source: Ramesh *et al.* (2022). OpenAI DALL-E 2: Examples of 1024x1024 high-resolution images generated conditioned on the texts shown below each image.

Figure 12 - CLRS Algorithmic Reasoning Benchmark



Sorting: Insertion sort, bubble sort, heapsort (Williams, 1964), quicksort (Hoare, 1962).

Searching: Minimum, binary search, quickselect (Hoare, 1961).

Divide and Conquer (D&C): Maximum subarray (Kadane's variant (Bentley, 1984)).

Greedy: Activity selection (Gavril, 1972), task scheduling (Lawler, 1985).

Dynamic Programming: Matrix chain multiplication, longest common subsequence, optimal binary search tree (Aho et al., 1974).

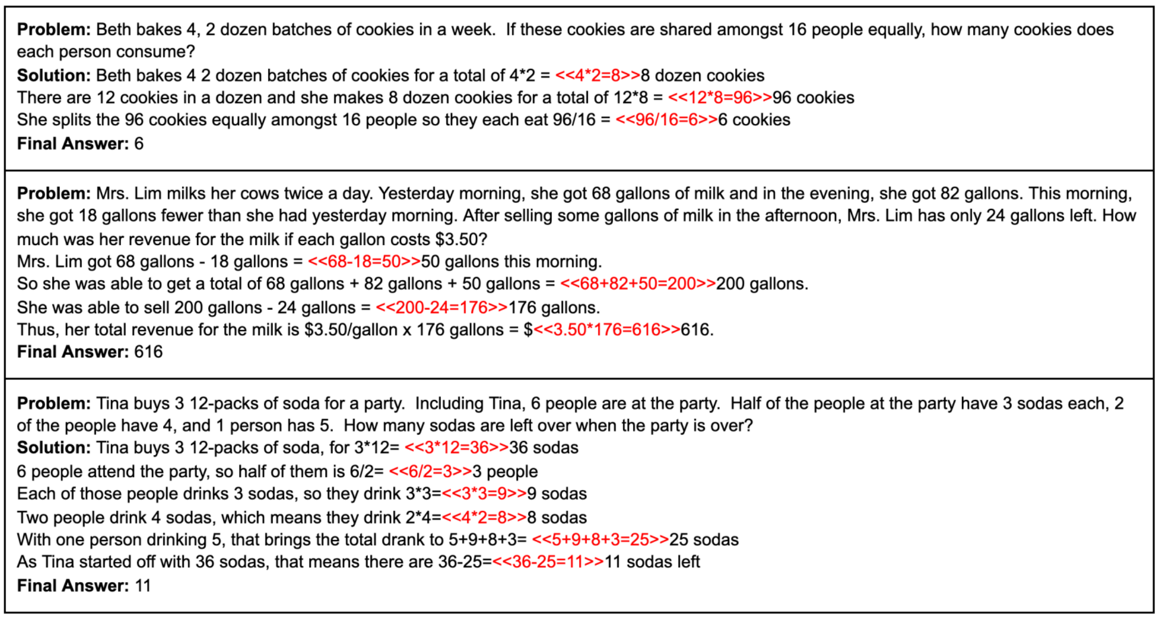
Graphs: Depth-first and breadth-first search (Moore, 1959), topological sorting (Knuth, 1973), articulation points, bridges, Kosaraju's strongly-connected components algorithm (Aho et al., 1974), Kruskal's and Prim's algorithms for minimum spanning trees (Kruskal, 1956; Prim, 1957), Bellman-Ford and Dijkstra's algorithms for single-source shortest paths (Bellman, 1958; Dijkstra et al., 1959) (+ directed acyclic graphs version), Floyd-Warshall algorithm for all-pairs shortest paths (Floyd, 1962).

Strings: Naïve string matching, Knuth-Morris-Pratt (KMP) string matcher (Knuth et al., 1977).

Geometry: Segment intersection, Convex hull algorithms: Graham scan (Graham, 1972), Jarvis' march (Jarvis, 1973).

Source: Veličković *et al.* (2022). Examples of classical algorithms that neural networks are being shown to be able to learn. The CLRS Algorithmic Reasoning Benchmark provides datasets for these tasks. CLRS stands for "Cormen, Leiserson, Rivest, and Stein" in homage to the authors of the "Introduction to Algorithms" classical textbook (Cormen *et al.*, 2009).

Figure 13 - Math Word Problems



Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?
Solution: Beth bakes 4 2 dozen batches of cookies for a total of 4*2 = <<4*2=8>>8 dozen cookies
There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of 12*8 = <<12*8=96>>96 cookies
She splits the 96 cookies equally amongst 16 people so they each eat 96/16 = <<96/16=6>>6 cookies
Final Answer: 6

Problem: Mrs. Lim milks her cows twice a day. Yesterday morning, she got 68 gallons of milk and in the evening, she got 82 gallons. This morning, she got 18 gallons fewer than she had yesterday morning. After selling some gallons of milk in the afternoon, Mrs. Lim has only 24 gallons left. How much was her revenue for the milk if each gallon costs $3.50?
Mrs. Lim got 68 gallons - 18 gallons = <<68-18=50>>50 gallons this morning.
So she was able to get a total of 68 gallons + 82 gallons + 50 gallons = <<68+82+50=200>>200 gallons.
She was able to sell 200 gallons - 24 gallons = <<200-24=176>>176 gallons.
Thus, her total revenue for the milk is $3.50/gallon x 176 gallons = $<<3.50*176=616>>616.
Final Answer: 616

Problem: Tina buys 3 12-packs of soda for a party. Including Tina, 6 people are at the party. Half of the people at the party have 3 sodas each, 2 of the people have 4, and 1 person has 5. How many sodas are left over when the party is over?
Solution: Tina buys 3 12-packs of soda, for 3*12= <<3*12=36>>36 sodas
6 people attend the party, so half of them is 6/2= <<6/2=3>>3 people
Each of those people drinks 3 sodas, so they drink 3*3=<<3*3=9>>9 sodas
Two people drink 4 sodas, which means they drink 2*4=<<4*2=8>>8 sodas
With one person drinking 5, that brings the total drank to 5+9+8+3= <<5+9+8+3=25>>25 sodas
As Tina started off with 36 sodas, that means there are 36-25=<<36-25=11>>11 sodas left
Final Answer: 11

Source: Cobbe *et al.* (2021). Large language models are being adapted to tackle grade school math problems. Considering that humans think using natural language, it is reasonable to tackle reasoning-based tasks using improved language models.

The advances are not restricted to discriminative tasks such as classification, object detection, and semantic segmentation. Instead, they are present in generative tasks as well. In fact, diffusion-based models such as Imagen[9] from Google (Saharia *et al.*, 2022) (Fig. 10) and DALL-E 2[10] from OpenAI (Fig. 11) are producing promising results (Ramesh *et al.*, 2022).

We have recently seen advances in areas that we may have believed that neural networks could never reach. For example, some works show that it is possible to learn reasoning to perform classical algorithmic tasks such as sorting, searching, dynamic programming, graph algorithms, string algorithms, and geometric algorithms (Fig. 12) (Veličković *et al.*, 2022).

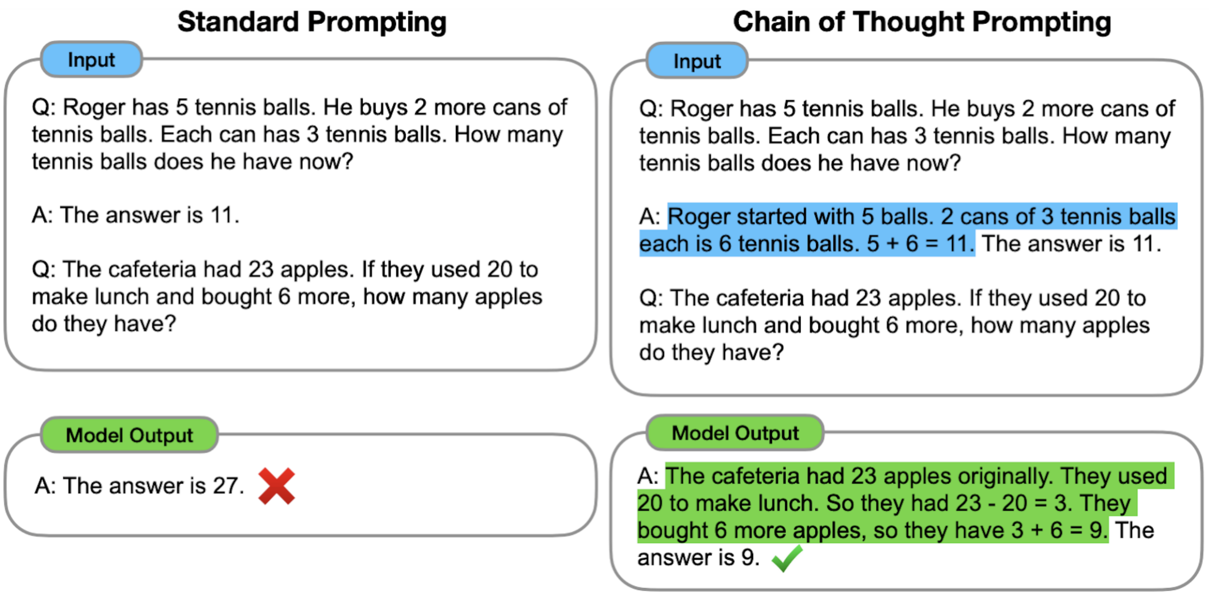
Related to approaches that use deep learning for reasoning-related tasks, we have seen that large language models may be adapted to solve math[11,12] (Cobbe *et al.*, 2021) (Fig. 13), arithmetic, logical, and symbolic reason-based problems expressed in natural language. One possible idea is to ask the model to tackle the problem by prompting the model. Similarly to humans, "thinking" step by step helps to solve logical statements[13] (Wei *et al.*, 2022; Kojima *et al.*, 2022) (Fig. 14). Some recent works are using language models for robotics[14].

Nowadays, we have much more resources efficient models (Tan & Le, 2019; Mehta & Rastegari, 2021) and also techniques such as quantization (Gholami *et al.*, 2021), pruning (Liang *et al.*, 2021), and distillation (Gou *et al.*, 2021) that allow us to deploy deep neural networks into constrained embedded systems and devices. Finally, we have expectations[15] that these models will consume much less energy in the near future (Patterson *et al.*, 2022; Wright *et al.*, 2022).

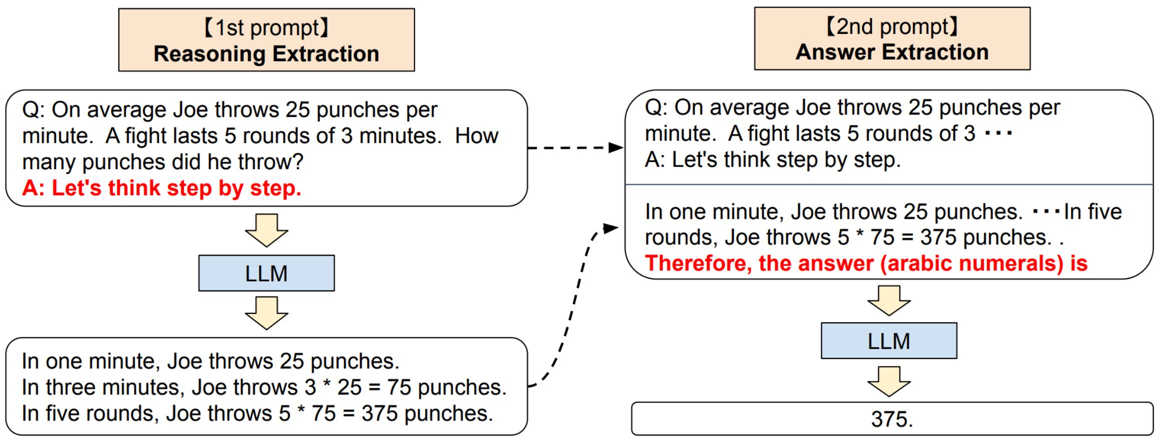
[9]https://imagen.research.google/
[10]https://openai.com/dall-e-2/
[11]https://openai.com/blog/formal-math/
[12]https://openai.com/blog/grade-school-math/
[13]https://ai.googleblog.com/2022/05/language-models-perform-reasoning-via.html
[14]https://say-can.github.io/
[15]https://spectrum.ieee.org/analog-ai

Figure 14 - Prompt Engineering: Chain of Thought (CoT)

(a)

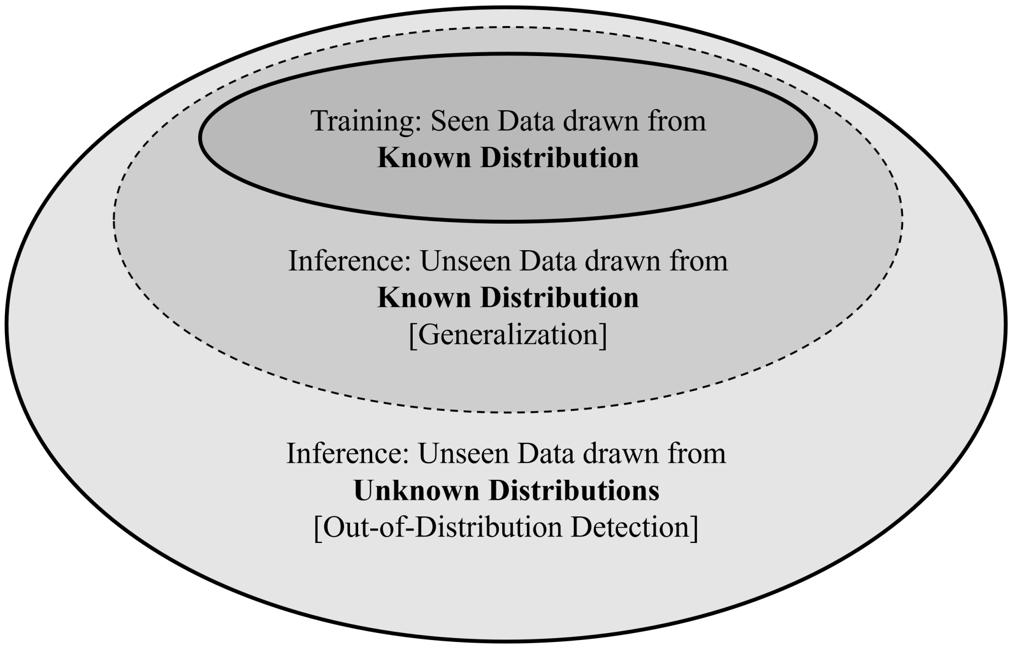

Standard Prompting
Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Output
A: The answer is 27.
Chain of Thought Prompting
Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Output
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9.

(b)


【1st prompt】
Reasoning Extraction
Q: On average Joe throws 25 punches per minute. A fight lasts 5 rounds of 3 minutes. How many punches did he throw?
A: Let's think step by step.
LLM
In one minute, Joe throws 25 punches.
In three minutes, Joe throws 3 * 25 = 75 punches.
In five rounds, Joe throws 5 * 75 = 375 punches.
【2nd prompt】
Answer Extraction
Q: On average Joe throws 25 punches per minute. A fight lasts 5 rounds of 3 ···
A: Let's think step by step.
In one minute, Joe throws 25 punches. ···In five rounds, Joe throws 5 * 75 = 375 punches. .
Therefore, the answer (arabic numerals) is
LLM
375.

Sources: Wei *et al.* (2022) and Kojima *et al.* (2022). Chain of Thought (CoT) Prompt Engineering Approaches: a) A CoT approach using a single inference (Wei *et al.*, 2022). b) The Zero-shot-CoT pipeline requires two prompted inferences. The first inference prompts the model to "think step by step" to obtain a complete reasoning path from the large pretrained language model. We ask for a synthetic conclusion in the second inference that prompts the model with "Therefore, the answer is:". These simple prompt engineering procedures significantly increase the model performance in the analyzed reasoning task (Kojima *et al.*, 2022).

Figure 15 - The Supervised Learning Paradigm and Unknown Distributions


Training: Seen Data drawn from
Known Distribution
Inference: Unseen Data drawn from
Known Distribution
[Generalization]
Inference: Unseen Data drawn from
Unknown Distributions
[Out-of-Distribution Detection]

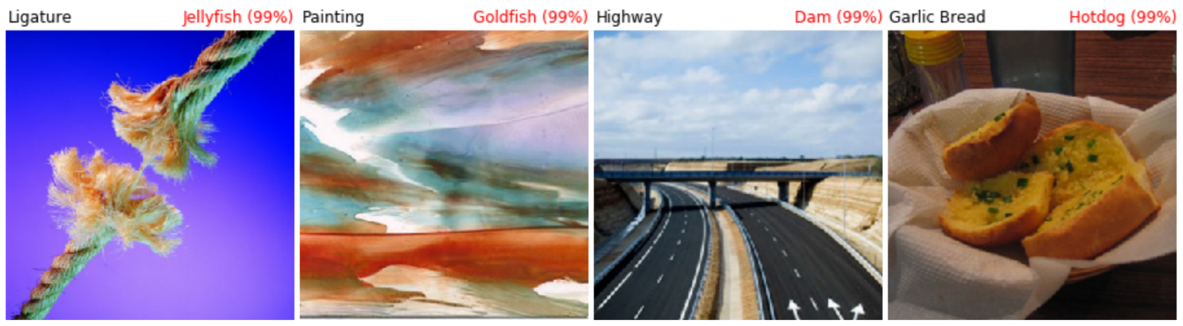
Source: The Author (2022). The Supervised Learning Paradigm and the Unknown Distributions: The machine learning paradigm is based on using data to understand the world and generalize to novel situations. However, to acquire this capability, the model is presented with data that necessarily represent a limited portion of the real-world complexity. In classification problems, during training, the model is trained with examples of known classes. During inference, the system usually satisfactorily generalizes whether unseen examples of these known classes have been presented. However, if a sample that does not represent an instance of a known class is shown to the system, *how may it know that it does not know?* Therefore, one of the significant challenges to current deep learning (and, in general, machine learning) systems is realizing when it cannot reliably perform the task it was trained to and behave appropriately.

1.2 PROBLEM: DEEP LEARNING ROBUSTNESS

Despite recent advances in self-supervision (Chen *et al.*, 2020b; Grill *et al.*, 2020), it is indisputable that the vast majority of success cases of deep learning practical applications are based on pure supervised learning or fine-tuning of pretrained models. In supervised deep learning, the classification task is the cornerstone of building more specialized tasks in computer vision (object detection (Redmon *et al.*, 2016), semantic segmentation (Chen *et al.*, 2018), instance segmentation (Ren *et al.*, 2017; He *et al.*, 2017), etc.) and natural language processing (text classification (Minaee *et al.*, 2021), named entity recognition (Li *et al.*, 2022), etc.).

However, deep learning classifiers present a significant drawback in their current form. We usually expect to present that an instance of a known class is presented to the neural network for inference. If this holds, neural networks commonly show satisfactory performance. However, in real-world applications, which are becoming even more common after the deep learning advent, this assumption may not be fulfilled.

Figure 16 - Unknown Distributions Examples



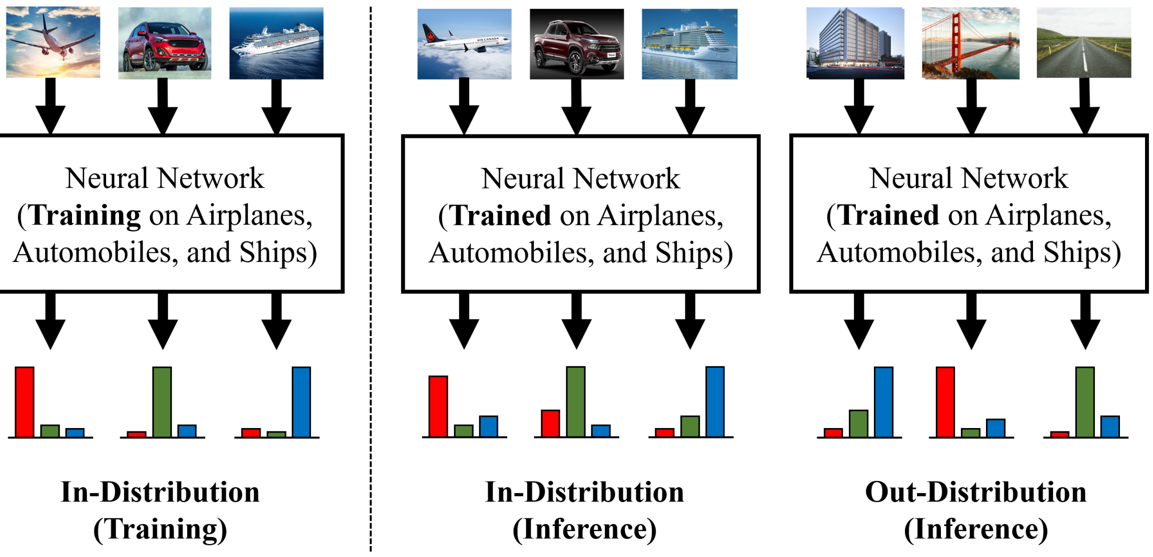
Source: Hendrycks *et al.* (2019b). Images from ImageNet-O dataset, which was created for the OOD detection task. These examples do not belong to ImageNet classes. For each image, the black text represents the correct ImageNet-O class not presented in ImageNet. The red text is the confidence of the ResNet-50 prediction that the image belongs to an ImageNet class. Images of unknown distributions are wrongly assigned highly confident predictions by the model trained on ImageNet.

Indeed, it is unrealistic to expect training examples to thoroughly consider the complexity to which the system will be submitted when in the field. We emphasize that this is not a problem exclusively for deep learning, but rather for machine learning classification systems in general. The possibility of being presented during test an instance that does not belong to any training class is a limitation of the learning paradigm itself in its current form (Fig. 15). The relevance of this problem incentivizes researchers to propose alternatives to the principle of Empirical Risk Minimization (ERM) (Vapnik, 1991). For example, Krueger *et al.* (2020) propose the principle of Risk Extrapolation (REx) that incorporates the unknown distribution aspect of the problem into the learning theory.

The ability to detect whether an input applied to a neural network does not represent an example of trained classes is essential to building robust applications in medicine, finance, agriculture, engineering, fraud detection, and many others. In such situations, it is better to have a system that can recognize that the sample should not be classified. Instead, current systems classify unknown class instances and usually present very high confidence for them (Fig. 16).

Indeed, the rapid adoption of neural networks in modern real-world applications makes the development of systems that can detect when dealing with examples that belong to unknown distributions a primary necessity from a practical point of view. Besides interpretability, casual inference, reasoning, common sense, privacy, fairness, security, and other robustness aspects, this constitutes one of the primary challenges to construct more reliable deep learning systems.

Figure 17 - Out-of-Distribution Detection Problem Statement

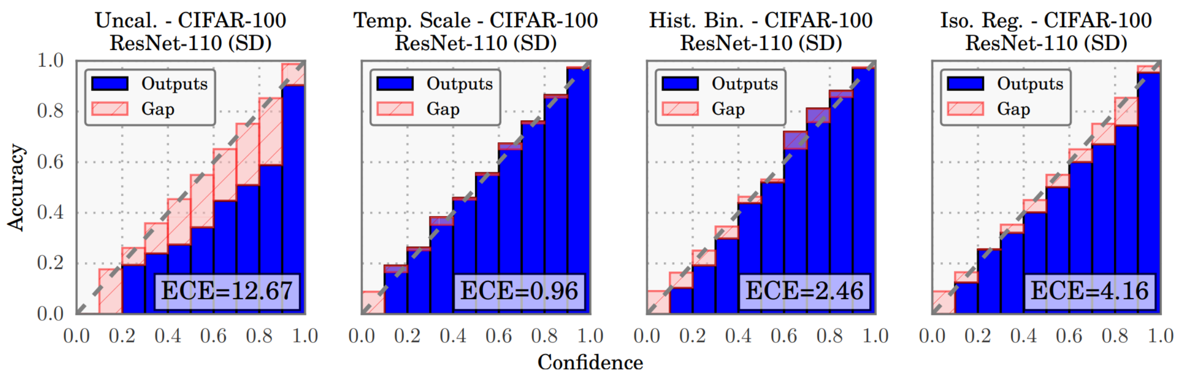

Neural Network
(**Training** on Airplanes, Automobiles, and Ships)
Neural Network
(**Trained** on Airplanes, Automobiles, and Ships)
Neural Network
(**Trained** on Airplanes, Automobiles, and Ships)
In-Distribution (Training)
In-Distribution (Inference)
Out-Distribution (Inference)

Source: The Author (2022). Out-of-distribution detection: Deep models are trained on a limited set of a priori known classes, learning a combined in-distribution (left). During the inference, if an instance of a trained class (an example sampled from the in-distribution) is presented to the considered system, the performance is usually satisfactory (middle). However, in real-world applications, the system is commonly subjected to examples of unknown classes (instances sampled from unknown distributions, collectively called out-distributions). In such situations, current models usually provide high-confidence predictions. It is a problem that affects any classification machine learning system, rather than only deep learning models. Out-of-distribution detection is the ability to detect such situations and consequently avoid nonsense classification.

This problem has been studied under many similar or related nomenclatures, such as spurious patterns (Vasconcelos *et al.*, 1995), open set recognition (Vareto *et al.*, 2017; Scheirer *et al.*, 2014) and open-world recognition (Bendale & Boult, 2015; Rudd *et al.*, 2018). Recently, to quantify the advance in our ability to construct a more reliable deep learning, Hendrycks & Gimpel (2017) defined out-of-distribution (OOD) detection as the task of evaluating whether a sample does not come from the In-Distribution (ID) on which a neural network was trained (Fig. 17).

OOD detection is closely related to anomaly and novelty detection (Pang *et al.*, 2021). However, in OOD detection, we have multiple (usually many more than two) classes. From an anomaly detection perspective, examples that belong to any of these classes are considered “normal”. Additionally, we have labels that individually identify examples from each of these “normal” classes, which collectively represent what we call the in-distribution. There are no training examples of the “abnormal” class, which are called out-of-distribution examples in the context of OOD detection. We have to decide whether we have an in-distribution (“normal”) example or an out-of-distribution (“abnormal”) example during inference. In the first case, we also have to predict the correct class.

Figure 18 - Reliability Diagrams and Expected Calibration Errors



Source: Guo *et al.* (2017). Reliability diagrams and Expected Calibration Error (ECE) for CIFAR-100: Before calibration (far left), the ECE is higher than after using calibration techniques (middle left, middle right, far right).

The relevance of building deep learning systems that support out-of-distribution detection can not be overestimated. Indeed, in his Turing Award Lecture[16] and in the invited talk at NeurIPS 2019[17], Yoshua Bengio stated that out-of-distribution detection is currently one of the major challenges to move AI forward.

There are many real-world cases in which OOD detection support is important for the solution to work properly. For example, Andrej Karpathy, the AI Tesla Director, explains how a self-driving car system trained on a set of standard stop signs usually has a hard time identifying "weird" stop signs[18]. In bacteria detection approaches, it is crucial to know when a new bacterium may exist[19].

Out-of-distribution capabilities are particularly important in many medical applications (Zadorozhny *et al.*, 2021). For example, it is critical when dealing with applications dealing with Electronic Health Records. Food classifiers should be able to reject other kind of images (Yang *et al.*, 2021). Therefore, in OOD detection capable approaches, the system presents an auxiliary task that evaluates whether it is indeed able to perform the primary classification task reliably.

Hendrycks & Gimpel (2017) introduced benchmark datasets and metrics for OOD detection. Additionally, they established the baseline performance by proposing an OOD detection approach that uses the maximum predicted probability as the score to easily detect OOD examples. Despite being a fundamental task to build reliable AI systems, most current OOD detection approaches are based on ad hoc techniques that produce unwanted side effects and add troublesome requirements to the solution.

[16]https://www.youtube.com/watch?v=llGG62fNN64
[17]https://slideslive.com/38922304/from-system-1-deep-learning-to-system-2-deep-learning
[18]https://www.youtube.com/watch?v=hx7BXih7zx8
[19]https://ai.googleblog.com/2019/12/improving-out-of-distribution-detection.html

In addition to OOD detection, uncertain estimation is relevant to producing more robust deep learning systems. Indeed, Guo *et al.* (2017) showed that neural networks are usually uncalibrated because their predicting probabilities are not representative of the actual correctness likelihood, which may be a significant drawback in many applications. It is reasonable to want the predicted probability to represent somehow the chance of the classification to be correct.

Besides showing that deep networks are usually uncalibrated (Fig. 18), Guo *et al.* (2017) used the Expected Calibration Error (ECE) to measure this. Finally, they also proposed many approaches to calibrate them.

On the one hand, OOD detection deals with rejecting examples that should not even be classified because they are *not* instances of trained classes. On the other hand, uncertainty estimation tackles the problem of assigning realistic probabilities to *in-distribution* samples. We have observed in the literature that this two tasks are usually study simultaneously[20].

[20]https://sites.google.com/view/udlworkshop2021/home

1.3 MOTIVATION

The explanations above show the potential of deep learning to be used even more broadly in practice. It certainly has the potential to improve the quality of our lives. Since the groundbreaking results obtained in 2012 with the ImageNet competition, we have noticed that deep learning has been expanding fast in the last ten years.

Indeed, from a technology used mainly in computer vision a decade ago, deep learning has currently been delivering remarkable state-of-the-art results also in natural language processing, speech, and audio processing. The adoption of deep learning approaches is not restricted to unstructured data, as methods for tabular data and time series have been proposed. We have also witnessed promising approaches to tackle robotics using pretrained language models.

We have recently seen deep learning approaches being used to promote relevant scientific discoveries and even to tackle reasoning-like tasks using large pretrained language models. It is somewhat surprising that neural networks, historically associated with perception-like tasks, are being used successfully for reasoning tasks.

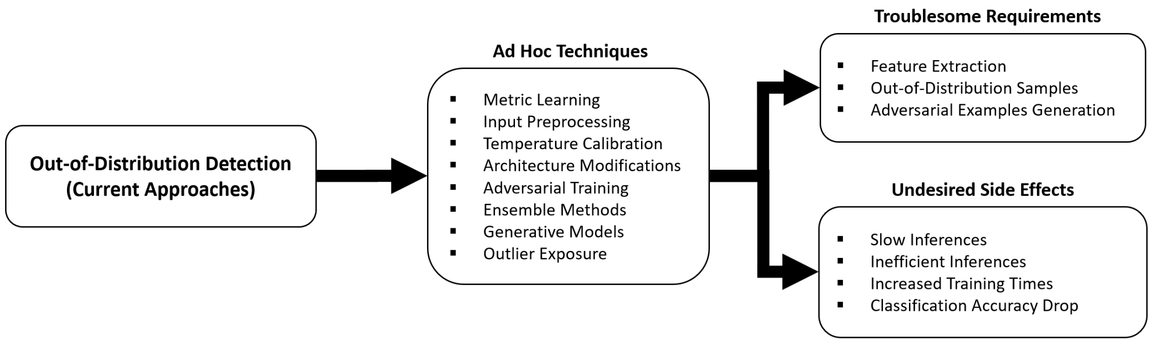
Novel technologies are making it possible to train and use deep learning models with a significantly reduced amount of labeled data (few-shot and one-shot approaches) or even no labeled data at all (self-supervision and contrastive learning). Recent techniques are allowing deploying deep models into resource-constrained embedded devices. Novel promising technologies show that it is possible to dramatically reduce the carbon footprint and energy consumption to train and perform inferences with such systems.

However, for some applications, it is crucial to reject no-sense predictions or mismatch something known with something unknown (e.g., self-driving cars). In other cases, it is critical to produce predictions with probabilities that reflect the real chance that the system is correct in its predictions. For example, for cancer diagnoses, the system must inform a probability that reflects accurate chances that it is indeed correct.

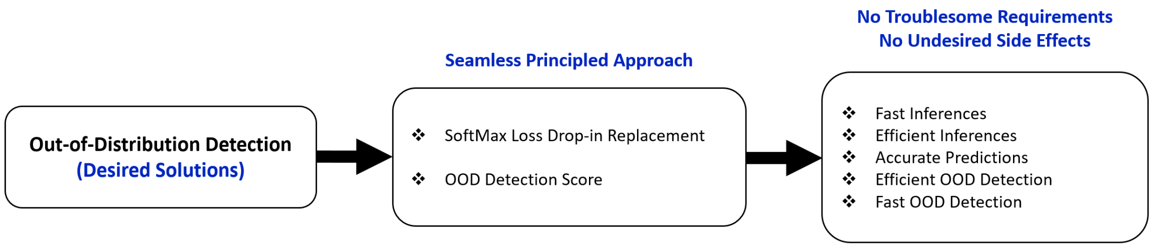
Therefore, contributing to the incorporation of those capabilities into AI-based systems is the primary motivation of this work. We believe constructing more robust deep learning systems is essential to improving people's lives.

Figure 19 - Current Approaches vs. Desired Solutions

(a)



(b)


Out-of-Distribution Detection (Desired Solutions)
Seamless Principled Approach
SoftMax Loss Drop-in Replacement
OOD Detection Score
No Troublesome Requirements
No Undesired Side Effects
Fast Inferences
Efficient Inferences
Accurate Predictions
Efficient OOD Detection
Fast OOD Detection

Source: The Author (2022). (a) Current approaches typically use a combination of conventional ad hoc techniques, such as input preprocessing, temperature calibration, adversarial training, and metric learning. However, these techniques add troublesome requirements and undesired side effects to the overall solution. Unrealistic access to OOD examples in design time is a common undesired requirement. Slow and inefficient inferences compared to those performed by neural networks trained using standard procedures) and Classification Accuracy (ACC) drop are usually unwanted side effects. (b) Unlike current approaches, in this work, our goal is to design solutions that completely avoid conventional ad hoc techniques and associated side effects and requirements. Consequently, models trained using our loss should produce accurate predictions (no classification accuracy drop) and fast and efficient inferences. Unlike many current OOD detection methods, feature extraction and metric learning after neural network training should also be avoided. The solutions have to be based on theoretical motivations. In this figure, desired solutions represent the characteristics of the solutions we aim to achieve.

1.4 OBJECTIVE

The main objective of this work is to design an OOD detection approach that, unlike current ones, allows out-of-distribution detection without relying on ad hoc techniques that add extra requirements or unwanted side effects to the overall solution (Fig. 19a).

Throughout this work, we argue that the unsatisfactory OOD detection performance of modern neural networks is mainly due to the drawbacks of the currently used loss rather than architecture limitations. Indeed, the SoftMax loss[21] (i.e., the combination of the output linear layer, the SoftMax activation, and the cross-entropy loss) anisotropy caused by the linear layer does not induce the concentration of high-level representations in the feature space (Wen *et al.*, 2016; Hein *et al.*, 2019), which makes OOD detection difficult (Hein *et al.*, 2019). Furthermore, the SoftMax loss generates extremely low entropy (high confidence) posterior probability distributions (Guo *et al.*, 2017) in conflict with the fundamental principle of maximum entropy.

Therefore, we aim to contemplate novel losses that are able to work as drop-in replacements to the one currently used to train neural networks, therefore avoiding the need to change the model. No changes to the training procedure should be required. Hence, to enforce isotropy, we aim to design an exclusively distance-based loss able to train deep neural networks end-to-end in a straightforward way. Furthermore, to perform inference time OOD detection, we intend to design novel scores[22] that need to be fast and computationally efficient[23] (Fig. 19b).

To construct the desired solutions, our scores need to be defined *a priori* rather than be trainable. In addition, we aim to base the proposed approaches on solid theoretical foundations, most likely based on the fundamentals of information theory. For all these reasons, we call the overall solutions to be designed seamless and principled OOD detection methods.

To achieve these objectives, we investigate the following fundamental research questions:

RQ1. Is it possible to perform OOD by changing only the losses? How should we calculate the score in such cases?

RQ2. Is there any theoretic motivation (e.g., a fundamental principle) that sustains such OOD detection methods?

RQ3. Which characteristics would such solutions present?

In summary, we intend to study whether it is possible to design OOD detection approaches competitive to (or that outperform) current practices, avoiding their troublesome requirements and unwanted side effects. To answer **RQ1**, **RQ2**, and **RQ3**, we intend to investigate current approaches limitations and propose novel OOD detection solutions by focusing exclusively on neural networks and information theory foundations.

[21] We follow the "SoftMax loss" expression as defined in Liu *et al.* (2016).

[22] A score is a scalar that measures the likelihood of an instance belonging to the in or the out-distribution.

[23] In this work, we consider that an approach does not present classification accuracy drop if it never presents a classification accuracy more than one percent lower than the correspondent SoftMax loss trained model.

1.5 PUBLICATIONS

Here, we present the list of our *deep learning* publications:[24]:

Books

Foundations

- David Macêdo. **Enhancing Deep Learning Performance Using Displaced Rectifier Linear Unit.** *Editora Dialética*, ISBN Physical book: 978-65-252-3056-6, ISBN E-book: 978-65-252-3075-7, 2022.

Related Papers

Foundations

- D Macêdo, TI Ren, C Zanchettin, ALI Oliveira, T Ludermir. **Entropic Out-of-Distribution Detection.** *2021 International Joint Conference on Neural Networks (IJCNN).*

- D Macêdo, TI Ren, C Zanchettin, ALI Oliveira, T Ludermir. **Entropic Out-of-Distribution Detection: Seamless Detection of Unknown Examples.** *IEEE Transactions on Neural Networks and Learning Systems, 2022.*

- D Macêdo, T Ludermir. **Enhanced Isotropy Maximization Loss: Seamless and High-Performance Out-of-Distribution Detection Simply Replacing the SoftMax Loss.** *arXiv preprint arXiv:2105.14399 (paper under review).*

- D Macêdo, C Zanchettin, T Ludermir. **Distinction Maximization Loss: Efficiently Improving Classification Accuracy, Uncertainty Estimation, and Out-of-Distribution Detection Simply Replacing the Loss and Calibrating.** *arXiv preprint arXiv:2205.05874 (paper under review).*

[24]For an updated version, please visit: `https://scholar.google.com/citations?user=hypWII4AAAAJ&hl=en`

Additional Papers

Foundations

- D Macêdo, C Zanchettin, ALI Oliveira, T Ludermir. **Enhancing Batch Normalized Convolutional Networks using Displaced Rectifier Linear Units: A Systematic Comparative Study.** *Expert Systems with Applications, 2019.*

- D Macêdo, P Dreyer, T Ludermir, C Zanchettin. **Training Aware Sigmoidal Optimizer.** *arXiv preprint arXiv:2102.08716.*

Computer Vision

- MEN Gomes, D Macêdo, C Zanchettin, PSG de Mattos Neto, A Oliveira **Multi-human Fall Detection and Localization in Videos.** *Computer Vision and Image Understanding, 2022.*

- A Ayala, B Fernandes, F Cruz, D Macêdo, C Zanchettin. **Convolution Optimization in Fire Classification.** *IEEE Access, 2022.*

- W Costa, D Macêdo, C Zanchettin, LS Figueiredo, V Teichrieb. **Multi-Cue Adaptive Emotion Recognition Network.** *arXiv preprint arXiv:2111.02273.*

- A Ayala, B Fernandes, F Cruz, D Macêdo, ALI Oliveira, C Zanchettin. **Kutralnet: A portable deep learning model for fire recognition.** *2020 International Joint Conference on Neural Networks (IJCNN).*

- RB das Neves, LF Verçosa, D Macêdo, BLD Bezerra, C Zanchettin. **A Fast Fully Octave Convolutional Neural Network for Document Image Segmentation.** *2020 International Joint Conference on Neural Networks (IJCNN).*

- H Felix, WM Rodrigues, D Macêdo, F Simões, ALI Oliveira, V Teichrieb, C Zanchettin. **Squeezed Deep 6DoF Object Detection Using Knowledge Distillation.** *2020 International Joint Conference on Neural Networks (IJCNN).*

- JLP Lima, D Macêdo, C Zanchettin. **Heartbeat Anomaly Detection using Adversarial Oversampling.** *2019 International Joint Conference on Neural Networks (IJCNN).*

- D Castro, D Pereira, C Zanchettin, D Macêdo, BLD Bezerra. **Towards Optimizing Convolutional Neural Networks for Robotic Surgery Skill Evaluation.** *2019 International Joint Conference on Neural Networks (IJCNN).*

- CC de Amorim, D Macêdo, C Zanchettin. **Spatial-Temporal Graph Convolutional Networks for Sign Language Recognition.** *ICANN 2019: Artificial Neural Networks and Machine Learning.*

- AG Santos, CO de Souza, C Zanchettin, D Macêdo, ALI Oliveira, T Ludermir. **Reducing SqueezeNet Storage Size with Depthwise Separable Convolutions.** *2018 International Joint Conference on Neural Networks (IJCNN).*

- LA de Oliveira Junior, HR Medeiros, D Macêdo, C Zanchettin, ALI Oliveira, T Ludermir. **SegNetRes-CRF: A Deep Convolutional Encoder-Decoder Architecture for Semantic Image Segmentation.** *2018 International Joint Conference on Neural Networks (IJCNN).*

Natural Language Processing

- J Pereira, D Macêdo, C Zanchettin, A Oliveira, R Fidalgo. **PictoBERT: Transformers for Next Pictogram Prediction.** *Expert Systems With Applications, 2022.*

- J Moreira, C Oliveira, D Macêdo, C Zanchettin, L Barbosa. **Distantly-Supervised Neural Relation Extraction with Side Information using BERT.** *2020 International Joint Conference on Neural Networks (IJCNN).*

- J Abreu, L Fred, D Macêdo, C Zanchettin. **Hierarchical Attentional Hybrid Neural Networks for Document Classification.** *ICANN 2019: Artificial Neural Networks and Machine Learning.*

- AB Duque, LLJ Santos, D Macêdo, C Zanchettin. **Squeezed Very Deep Convolutional Neural Networks for Text Classification.** *ICANN 2019: Artificial Neural Networks and Machine Learning.*

- FAO Santos, KL Ponce-Guevara, D Macêdo, C Zanchettin. **Improving Universal Language Model Fine-Tuning using Attention Mechanism.** *2019 International Joint Conference on Neural Networks (IJCNN).*

Speech Processing

- JAC Nunes, D Macêdo, C Zanchettin. **AM-MobileNet1D: A Portable Model for Speaker Recognition.** *2020 International Joint Conference on Neural Networks (IJCNN).*

- JAC Nunes, D Macêdo, C Zanchettin. **Additive Margin SincNet for Speaker Recognition.** *2019 International Joint Conference on Neural Networks (IJCNN).*

Tabular Data

- AA Silva, AS Xavier, D Macêdo, C Zanchettin, ALI Oliveira **An Adapted GRASP Approach for Hyperparameter Search on Deep Networks Applied to Tabular Data.** *2022 International Joint Conference on Neural Networks (IJCNN).*

Time Series

- PM Vasconcelos, D Macêdo, LM Almeida, RGL Neto, CA Benevides, Cleber Zanchettin, Adriano LI Oliveira. **Identification of Microorganism Colony Odor Signature using InceptionTime.** *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC).*

Information Security

- PF de Araujo-Filho, G Kaddoum, DR Campelo, AG Santos, D Macêdo, C Zanchettin. **Intrusion Detection for Cyber–Physical Systems using Generative Adversarial Networks in Fog Environment.** *IEEE Internet of Things Journal, 2020.*

Financial Services

- JCS Silva, D Macêdo, C Zanchettin, ALI Oliveira, AT de Almeida Filho. **Multi-Class Mobile Money Service Financial Fraud Detection by Integrating Supervised Learning with Adversarial Autoencoders.** *2021 International Joint Conference on Neural Networks (IJCNN).*

Power Efficiency

- PFC Barbosa, BA da Silva, D Macêdo, C Zanchettin, RM de Moraes. **Otimização do Consumo de Energia em Redes Ad Hoc Aloha Empregando Deep Learning.** *2019 Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPERFORMANCE).*

2 BACKGROUND

"Truth is the daughter of time, not of authority."
–Francis Bacon

"If I have seen further, it is by standing on the shoulders of giants."
–Isaac Newton

"I swear to tell the truth, the whole truth, and nothing but the truth."
–Witness (Sworn testimony)

"If I swear to him, then all that I am is dead already."
–William Wallace (Braveheart)

Magistrate: "The prisoner wishes to say a word."
William Wallace: "Freeeedommm!"
–Last Scene (Braveheart)

In this chapter, considering one of our aims is to build isotropic (exclusively distance-based) losses designed to be applied to OOD detection, we discuss both current OOD detection approaches and distance-based losses. Moreover, we also introduce the maximum entropy principle that will have a fundamental role in our solutions.

2.1 CURRENT APPROACHES

The relevance of the out-of-distribution detection auxiliary task is demonstrated in Fig. 20 while Table 2 presents the limitations of major current approaches. In the following paragraphs, we explain in more detail their drawbacks.

Out-of-DIstribution Detector for Neural networks (ODIN) was proposed in Liang *et al.* (2018) by combining *input preprocessing* with *temperature calibration*. Although it significantly outperforms the baseline, the input preprocessing introduced in ODIN considerably increases the inference time by requiring an initial forward pass, a backpropagation, and finally a second forward pass to perform an inference that can be used for OOD detection.

Considering that backpropagation is typically slower than a forward pass, input preprocessing makes ODIN inferences at least three times slower than normal. Additionally, input preprocessing multiplies the inference power consumption and computational cost by at least three. Those are severe limitations from an economic and environmental perspective (Schwartz *et al.*, 2019). Several subsequent OOD detection proposals incorporated input preprocessing and its drawbacks (Liang *et al.*, 2018; Lee *et al.*, 2018; Hsu *et al.*, 2020; DeVries & Taylor, 2018).

Temperature calibration consists of changing the scale of the logits of a *pretrained model*. Both input preprocessing and temperature calibration require hyperparameter tuning. When proposing novel approaches, we should prefer the ones with the smallest possible number of hyperparameters to make the solution easier to use. It is essential in deep learning because we may require increased computational resources to perform extensive hyperparameter validation.

Moreover, ODIN requires unrealistic access to OOD samples to validate hyperparameters. Even if the supposed OOD samples are available during design-time, using these examples to tune hyperparameters makes the solution overfit to detect this particular type of out-of-distribution.

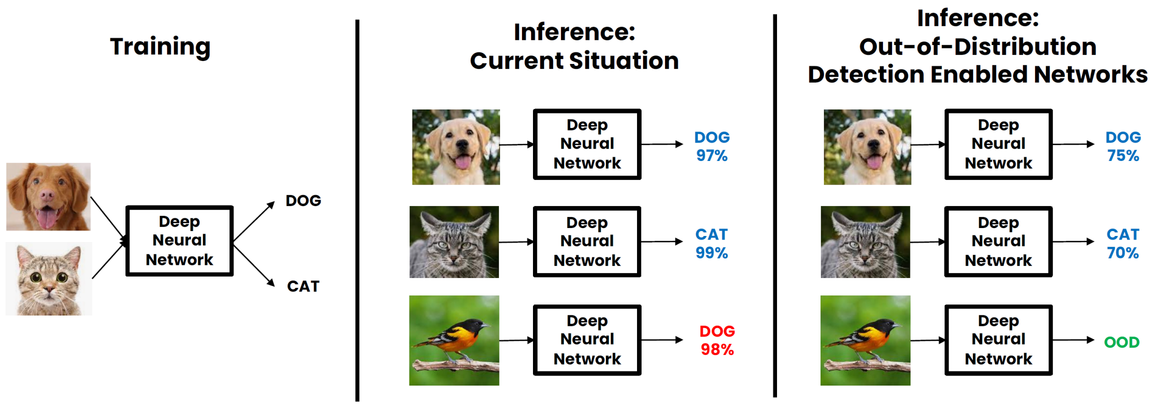
In real-world applications, the system will likely operate under different/novel/unknown out-of-distributions, and the estimated OOD detection performance could degrade significantly. Therefore, using OOD samples to validate hyperparameters may produce over-optimistic OOD detection performance estimations (Shafaei *et al.*, 2019).

The Mahalanobis distance-based method[1] (Lee *et al.*, 2018) overcomes the need for access to OOD samples by validating hyperparameters using adversarial examples, producing more realistic OOD detection performance estimates. However, the use of adversarial examples has the disadvantage of adding a cumbersome procedure to the solution. Even worse, the generation of adversarial samples itself requires hyperparameters, such as the maximum perturbations. Although adequate hyperparameters may be known for research datasets, they may be difficult to find for novel real-world data.

Moreover, as the Mahalanobis approach also requires input preprocessing, the previously mentioned drawbacks associated with this technique are still present in the Mahalanobis solution. Hence, both ODIN (Liang *et al.*, 2018) and the Mahalanobis distance-based method (Lee *et al.*,

[1]For the rest of this work, the expression “the Mahalanobis distance-based method” is replaced by “the Mahalanobis method”.

Figure 20 - Out-of-Distribution Detection Relevance



Source: The Author (2022). Out-of-Distribution Detection Relevance: In the real world, the deep neural network is subject to out-of-distribution (OOD) examples. Ideally, the system should be able to recognize such situations.

Table 2 - OOD Detection Approaches: Special Requirements and Side Effects

Approach	Special Requirement		Side Effect	
	Hyperparameter Tuning	Additional Data	Inefficient Inference or OOD Detection	Classification Accuracy Drop
ODIN[1]	Required	Not Required	Present	Not Present
Mahalanobis[2]	Required	Not Required	Present	Not Present
ACET[3]	Required	Not Required	Not Present	Present
Outlier Exposure[4]	Not Required	Required	Not Present	Not Present
GODIN[5]	Required	Not Required	Present	Present
Gram Matrices[6]	Not Required	Not Required	Present	Not Present
Scaled Cosine[7]	Not Required	Not Required	Not Present	Present
Energy-based[8]	Required	Required	Not Present	Not Present
Deep Ensemble[9]	Not Required	Not Required	Present	Not Present
DUQ[10]	Required	Not Required	Present	Not Present
SNGP[11]	Required	Not Required	Present	Not Present

Source: The Author (2022). [1]Liang *et al.* (2018), [2]Lee *et al.* (2018), [3]Hein *et al.* (2019), [4]Hendrycks *et al.* (2019a), [5]Hsu *et al.* (2020), [6]Sastry & Oore (2019), [7]Techapanurak & Okatani (2019), [8]Liu *et al.* (2020b), [9]Lakshminarayanan *et al.* (2017), [10]van Amersfoort *et al.* (2020), [11]Liu *et al.* (2020a).

2018) use *input preprocessing*, which produces remarkably slow and energy-inefficient inferences, which is undesired because it is important to make deep learning more computationally efficient (Schwartz *et al.*, 2019).

Furthermore, *feature ensemble* introduced in the Mahalanobis approach also presents limitations. In fact, feature ensembles require training of ad hoc classification and regression models on features extracted from many network layers. Finally, the Mahalanobis method involves feature extraction and metric learning. Similar to the Mahalanobis distance-based approach, Vareto *et al.* (2017); Scheirer *et al.* (2014); Bendale & Boult (2015); Rudd *et al.* (2018) require metric learning on features extracted from pretrained models.

Hyperparameter tuning is also a drawback to methods based on adversarial training, such as Adversarial Confidence Enhancing Training (ACET) (Hein *et al.*, 2019), as we need to define the appropriated adversarial perturbations. Adversarial training is known for increasing training time (Wong *et al.*, 2020), reducing classification accuracy (Raghunathan *et al.*, 2019), and presenting limited scalability when dealing with large-size images (Shafahi *et al.*, 2019). Furthermore, adversarial training can cause a drop in classification accuracy (Raghunathan *et al.*, 2019).

In some cases, OOD detection proposals require architecture modifications (Yu & Aizawa, 2019) or ensemble methods (Vyas *et al.*, 2018; Lakshminarayanan *et al.*, 2017). Despite significantly improving the OOD detection performance, loss enhancement (regularization) techniques, such as outlier exposure (Hendrycks *et al.*, 2019a; Papadopoulos *et al.*, 2019), background methods (Dhamija *et al.*, 2018), and the energy-based fine-tuning (Liu *et al.*, 2020b) require the addition of carefully chosen extra/outlier/background data and expand memory usage. Moreover, they usually add hyperparameters to the solution.

Solutions based on uncertainty (or confidence) estimation (or calibration) (Kendall & Gal, 2017; Leibig *et al.*, 2017; Malinin & Gales, 2018; Kuleshov *et al.*, 2018; Subramanya *et al.*, 2017) usually present additional complexity, slow and energy-inefficient inferences (Schwartz *et al.*, 2019), and OOD detection performance typically worse than ODIN (Shafaei *et al.*, 2019; Hsu *et al.*, 2020).

The Entropic Open-Set loss and the Objectosphere loss were proposed in Dhamija *et al.* (2018). These two losses used background samples to improve the performance of detecting unknown inputs. The Entropic Open-Set loss works like the usual SoftMax loss in the in-distribution training data, producing a low entropy for these samples. However, it forces maximum entropy in the background samples. The Objectosphere loss is the Entropic Open-Set loss with an added regularization factor that forces the feature magnitude of in-distribution samples to be near a predefined value ξ while minimizing the feature magnitude of background samples.

Methods that use data-augmentation have been proposed. Tack *et al.* (2020) improve OOD detection in a self-supervised setting. Sastry & Oore (2019) analyzed statistics of the activations of the pretrained model on training and validation data to detect OOD examples.

In 2019, on the one hand, IsoMax (Macêdo *et al.*, 2019) proposed a *non-squared Euclidean distance last layer to address out-of-distribution detection* in an *end-to-end trainable way* (i.e., no feature extraction). On the other hand, Scaled Cosine (SC) (Techapanurak & Okatani, 2019) proposed using a *cosine distance*. Although the scale factor in IsoMax is a *constant* scalar called the entropy scale, Scaled Cosine requires the addition of a *block of layers* to learn the scale factor. This is made up of an exponential function, batch normalization, and a linear layer that has the feature layer as input. Moreover, to present high performance, it is necessary to avoid applying weight decay to this *extra learning block*. We believe that this additional learning block, which adds an ad hoc linear layer in the final of the neural network, may make the solution prone to *overfitting* and explain the classification accuracy drop mentioned by the authors.

In 2020, Generalized Out-of-DIstribution Detector for Neural networks (GODIN) (Hsu *et al.*, 2020) cited and was heavily inspired by Scaled Cosine. GODIN kept the *extra learning block* to learn the scale factor and also avoided applying weight decay to it. In addition to the usual affine transformation and cosine distance from Scaled Cosine, it presents a variant that uses a Euclidean distance-based last layer, similar to IsoMax. The major contribution of GODIN was to allow using the input preprocessing introduced in ODIN without the need for out-of-distribution data. However, input preprocessing increases the inference latency (i.e., reduces the inference efficiency) approximately four times (Macêdo *et al.*, 2022).

We emphasize that both Techapanurak & Okatani (2019); Hsu *et al.* (2020) presents *classification accuracy drop* of significantly more than one percent in some situations, which is a harmful side effect, as classification is commonly the primary aim of the system (Carlini *et al.*, 2019).

Moreover, Spectral-Normalized Neural Gaussian Process (SNGP) (Liu *et al.*, 2020a) cited, followed, and improved the idea introduced by IsoMax in 2019: A *distance-based output layer* for OOD detection. In a similar direction, Deterministic Uncertainty Quantification (DUQ) (van Amersfoort *et al.*, 2020) also proposed a modified *distance-based loss to address OOD detection*. However, unlike IsoMax variants (e.g., IsoMax, IsoMax+, and DisMax), SNGP and DUQ produce inferences not as efficient as those produced by a deterministic neural network (Liu *et al.*, 2020a). In 2021, the IsoMax+ loss (Macêdo & Ludermir, 2021) introduced the *isometric* distance. Considering that OOD and uncertainty estimation are related and relevant auxiliary tasks, modern approaches such as SNGP simultaneously tackle both.

2.2 DISTANCE-BASED LOSSES

Recently, neural network distance-based losses have been proposed in the context of face recognition. For example, the contrastive (Sun *et al.*, 2014) and triplet (Schroff *et al.*, 2015) losses use the high-level feature (embeddings) *pairwise* distances. In both cases, the SoftMax *function*[2] is not present, and the *squared* Euclidean distance is used. One of the main drawbacks is the need for using Siamese neural networks, which adds complexity to the solution and expands memory requirements during training (Wen *et al.*, 2016).

Additionally, the *triplet sampling* and *pairwise training*, which implicate the recombination of the training samples with dramatic data expansion, slow convergence and produce instability (Wen *et al.*, 2016). Finally, no prototypes are learned during training. The challenge of training networks using *purely* distance-based losses while avoiding *triplet sampling* and *pairwise training* was discussed in Wen *et al.* (2016), which proposed a *squared* Euclidean distance-based *regularization* procedure.

The center loss (Wen *et al.*, 2016) has two parameters α and λ. We call a loss isotropic when its dependency on the high-level features (embeddings) is performed *exclusively* through distances, which are usually calculated to the class prototypes. Any metric may be used for such purpose. Any *valid*[3] distance may be used to construct an isometric loss. In this sense, the center loss is *not* isotropic, as it presents affine transformation in its SoftMax classification term. Thus, the center loss inherits the drawbacks of the SoftMax loss affine transformation (See Chapter 3).

In Snell *et al.* (2017), the authors proposed a solution based on *squared* Euclidean distance to address few-shot learning. However, this approach does not work as a SoftMax loss drop-in replacement, as it does not simultaneously learn high-level features (embeddings) *and prototypes* using *exclusively* stochastic gradient descent (SGD) and *end-to-end* backpropagation. Indeed, despite learning embeddings using regular SGD and backpropagation, *additional offline procedures are required to calculate the class prototypes*.

[2] We follow the "SoftMax *function*" expression as defined in Liu *et al.* (2016).

[3] https://en.wikipedia.org/wiki/Metric_(mathematics)

Figure 21 - The Principle of Maximum Entropy

We should prefer the less confident possible predictions that produce the correct class.
Claude Shannon
E.T. Jaynes

Source: The Author (2022). The Principle of Maximum Entropy: From a set of probability distributions that satisfactorily describe the prior knowledge available, the distribution that presents the maximal information entropy (i.e., the least informative option) represents the best possible choice. The maximum entropy principle produces the least biased possible probability distribution (no extra assumptions not presented in data).

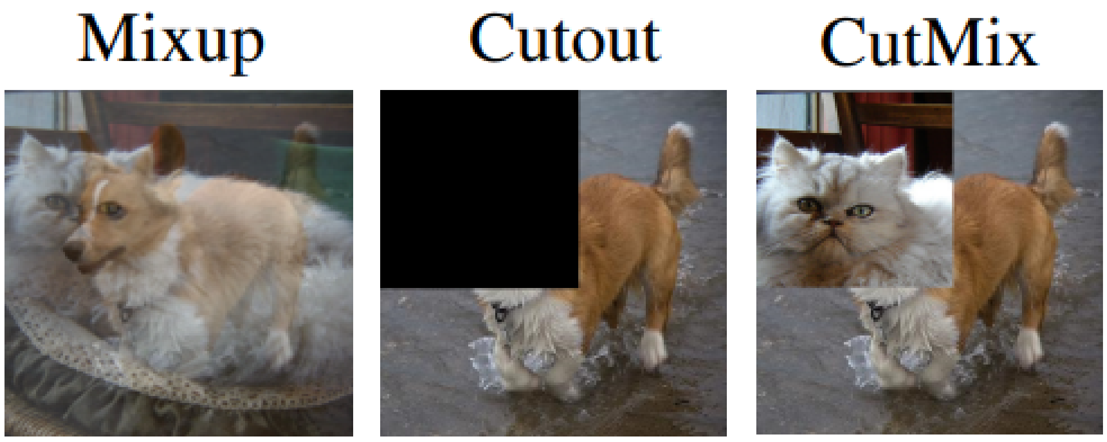
2.3 MAXIMUM ENTROPY PRINCIPLE

The principle of maximum entropy, formulated by E. T. Jaynes to unify the statistical mechanics and information theory entropy concepts (Jaynes, 1957a,b), states that when estimating probability distributions, we should choose the one that produces the maximum entropy consistent with the given constraints (Cover & Thomas, 2006). Following this principle, we avoid introducing additional assumptions or bias[4] not presented in the data.

From a set of trial probability distributions that satisfactorily describe the available prior knowledge, the distribution that presents the maximal information entropy, which is the least informative option, represents the best possible choice. In other words, we must produce posterior probability distributions as under-confident as possible as long as they match the correct predictions for accurate classification.

[4]https://mtlsites.mit.edu/Courses/6.050/2003/notes

Figure 22 - Data Augmentation Approaches



Source: Yun *et al.* (2019). Some recent data augmentation approaches are capable of increasing the OOD detection performance in some cases.

2.4 DATA AUGMENTATION

Some recent studies have shown that data augmentation may enhance OOD detection. For example, Yun *et al.* (2019) showed that composing two images improves out-of-distribution detection performance (Fig. 22). In Mixup (Zhang *et al.*, 2018), two images are added pixel by pixel using some weighted proportion. Cutout (Devries & Taylor, 2017) removed some portion of an image. CutMix replaces a piece of an image with a frame taken from another.

Building novel data augmentation methods to improve robustness is an interesting proposal because they do not impact the inference delay. However, we need to be careful to avoid increasing the training time. It is also important to avoid adding too many hyperparameters to the solution when designing novel data augmentation strategies.

3 ENTROPIC LOSSES

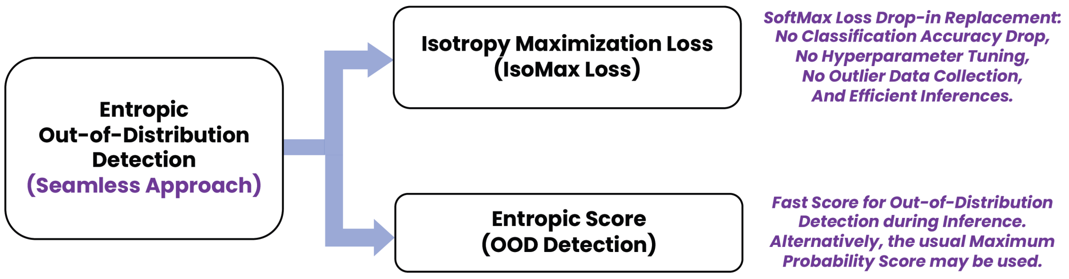
"More things should not be used than are necessary."
–William of Ockham

"Things should be made as simple as possible, but not simpler."
–Albert Einstein

"The prize is in the pleasure of finding the thing out, the kick in the discovery, the observation that other people use it [my work] – those are the real things, the honors are unreal to me."
–Richard Feynman

In this chapter, we present our solutions to the out-of-distribution and uncertainty estimation problems. We initially propose to use our isotropic distance-based loss IsoMax combined with the entropic score. Then, we evolve IsoMax to IsoMax+ by changing the initialization and performing what we call the isometrization of the distances used in IsoMax. We also propose the minimum distance score for out-of-distribution detection. Moreover, starting from IsoMax+, we add some modifications to present the state-of-the-art DisMax loss. Even more, we propose a novel score called the max-mean logit entropy score. Finally, we also built a fast way to achieve state-of-the-art uncertainty estimation by calibrating the temperature of DisMax trained models. We collectively call all proposed losses *entropic losses* because all of them are based on the principle of maximum entropy.

Figure 23 - Seamless Approach: Loss and Score


Entropic Out-of-Distribution Detection (Seamless Approach)
Isotropy Maximization Loss (IsoMax Loss)
SoftMax Loss Drop-in Replacement: No Classification Accuracy Drop, No Hyperparameter Tuning, No Outlier Data Collection, And Efficient Inferences.
Entropic Score (OOD Detection)
Fast Score for Out-of-Distribution Detection during Inference. Alternatively, the usual Maximum Probability Score may be used.

Source: The Author (2022). The OOD detection solution is composed of two parts: the IsoMax loss and the Entropic Score.

3.1 ISOTROPY MAXIMIZATION LOSS

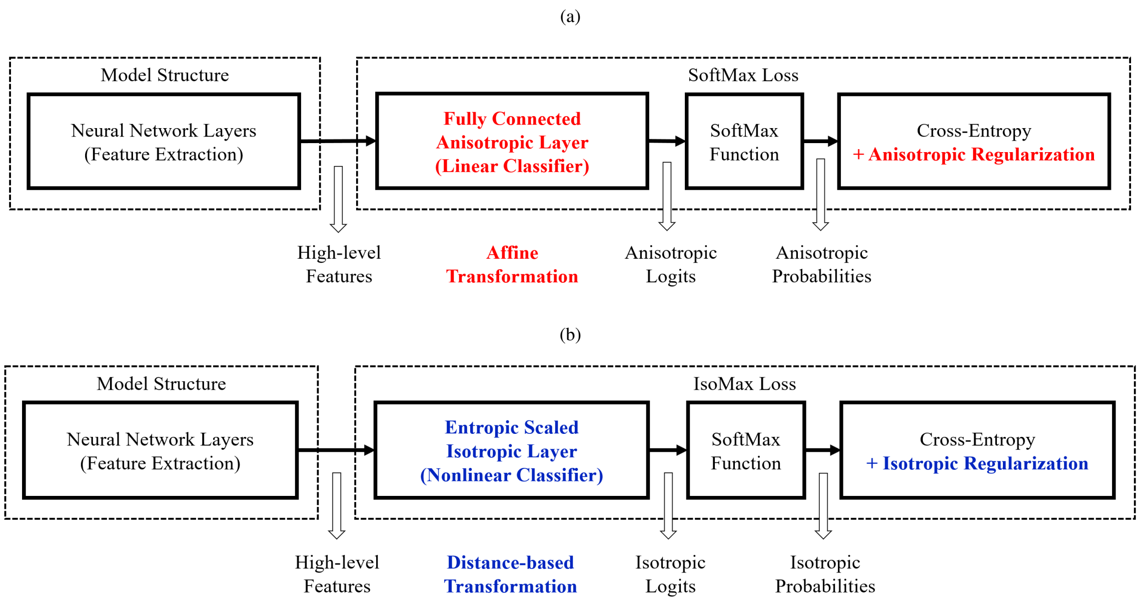
The first component of our seamless and principled approach is the Isotropy Maximization Loss (IsoMax) loss. The second is the rapid entropy score used for OOD detection. The entropic score is defined as the negative entropy of the neural network output probabilities. We chose the entropic score also based on the maximum entropy principle (Fig. 23).

As will be explained in detail in the next section, the IsoMax loss uses *exclusively distance-based logits to fix the SoftMax loss anisotropy caused by its affine transformation*. Moreover, we introduce the entropic scale, a *constant scalar multiplicative factor* applied to the logits *throughout training* that is, nevertheless, *removed before inference* to achieve high entropy posterior probability distributions in agreement with the maximum entropy principle.

The entropic scale is equivalent to the β of the SoftMax *function*. However, training with a *predefined constant* entropic scale and then *removing it before inference* is completely different from *temperature calibration*. On the one hand, in ODIN and similar methods based on *temperature calibration*, the temperature of a *pretrained* model is validated *after training*, which nevertheless was performed with a temperature equal to one. This validation usually requires unrealistic access to OOD or adversarial examples. Furthermore, over-optimistic performance estimation is commonly produced (Shafaei *et al.*, 2019). On the other hand, our approach requires neither hyperparameter validation nor access to OOD or adversarial data.

The intuitions that associate the unsatisfactory OOD detection performance of current neural networks with the SoftMax loss anisotropy and disagreement with the maximum entropy principle. The IsoMax loss trained models present accurate predictions and fast inferences that are energy- and computation-efficient. It does not require additional data. When using the entropy, rather than just one output, all network outputs are considered. In applications where the requirements and side effects of current OOD techniques are not a concern, future work may combine current approaches with our loss to achieve even higher OOD detection performance.

Figure 24 - Structural Blocks: SoftMax Loss vs. IsoMax Loss



Source: The Author (2022). Loss structural blocks: (a) SoftMax loss. Adapted from Liu *et al.* (2016). (b) IsoMax loss (ours). In contrast to the SoftMax loss affine transformation, the IsoMax loss nonlinear isotropic layer incentivizes the concentration of high-level representations around learnable prototypes, facilitating OOD detection while avoiding the need for feature extraction and metric learning post-processing phases. Moreover, the exclusively distance-based nonlinearity increases the neural network representation power. During training, the isotropic layer is multiplied by a constant value called the entropic scale. For inference, the entropic scale is removed to produce high entropy posterior probability distributions, as recommended by the maximum entropy principle. Finally, the OOD detection uses the entropic score, which is defined as the negative entropy of neural network output probabilities.

The swap of the SoftMax loss with the IsoMax loss requires changes in neither the model's architecture[1] nor training procedures or parameters. Fig. 24 synthesizes the differences between the SoftMax loss and the IsoMax loss.

3.1.1 Initial Considerations

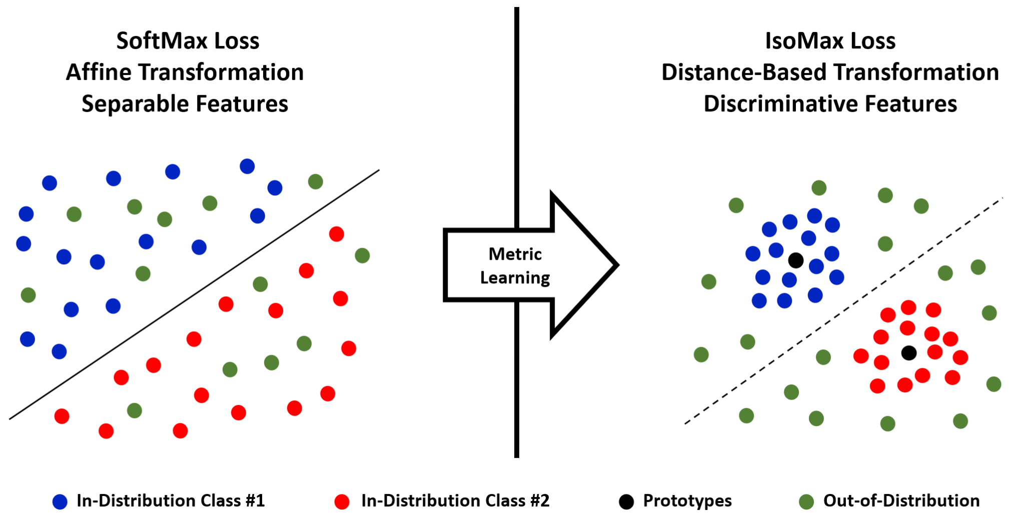
Let $\boldsymbol{x}$ represent the input applied to a neural network and $\boldsymbol{f_\theta}(\boldsymbol{x})$ represent the high-level feature vector produced by it. For this work, the underlying structure of the network does not matter. Considering k to be the correct class for a particular training example $\boldsymbol{x}$, we can write the SoftMax loss associated with this specific training sample as:

$$\mathscr{L}_S(\hat{y}^{(k)}|\boldsymbol{x}) = -\log\left(\frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{f_\theta}(\boldsymbol{x})+b_k)}{\sum\limits_j \exp(\boldsymbol{w}_j^\top \boldsymbol{f_\theta}(\boldsymbol{x})+b_j)}\right) \tag{3.1}$$

In Equation (3.1), $\boldsymbol{w}_j$ and b_j represent the weights and biases associated with class j, respectively. From a geometric perspective, the term $\boldsymbol{w}_j^\top \boldsymbol{f_\theta}(\boldsymbol{x})+b_j$ represents a hyperplane in the high-level feature space. It divides the feature space into two subspaces called positive

[1]Following a modern encoder-decoder and self-supervision terminology, we do not consider the last layer part of the architecture.

Figure 25 - Separable Features vs. Discriminative Features



Source: Adapted from Wen *et al.* (2016). SoftMax loss produces separable features (Wen *et al.*, 2016). Post-processing metric learning on features extracted from SoftMax loss trained networks may convert from the situation on the left to the situation on the right (Lee *et al.*, 2018; Mensink *et al.*, 2013; Vareto *et al.*, 2017; Scheirer *et al.*, 2014; Bendale & Boult, 2015; Rudd *et al.*, 2018). The IsoMax loss, which is an exclusively distance-based (isotropic) loss, tends to generate more discriminative features (Wen *et al.*, 2016). No feature extraction and subsequent metric learning are required when the IsoMax loss is used for training.

and negative subspaces. The deeper inside the positive subspace the feature vector $\boldsymbol{f_\theta}(\boldsymbol{x})$ of an example is located, the more likely the example is believed to belong to the considered class.

Therefore, training neural networks using SoftMax loss does not lead to agglomeration of representations of examples associated with a particular class into a limited region of the hyperspace, as it produces *separable features* rather than *discriminative features* (Wen *et al.*, 2016) (Fig. 25, left).

The immediate consequence is the propensity of neural networks trained with SoftMax loss to make high confidence predictions on examples that stay in regions far away from the training examples, which explains their unsatisfactory OOD detection performance (Hein *et al.*, 2019). Indeed, the SoftMax loss is based on affine transformations, which are essentially internal products. Consequently, the last layer representations of such networks tend to align in the direction of the weight vector, producing locally preferential directions in space and subsequently anisotropy.

The SoftMax loss anisotropy is usually corrected by using metric learning on neural network pretrained features (Lee *et al.*, 2018; Mensink *et al.*, 2013; Vareto *et al.*, 2017; Scheirer *et al.*, 2014; Bendale & Boult, 2015; Rudd *et al.*, 2018). For example, the high OOD detection performance of the Mahalanobis approach (Lee *et al.*, 2018) indicates that deploying locally isotropic spaces around class prototypes improves the OOD detection performance. In such solutions, a mapping from the extracted features to a novel embedding space is constructed and class prototypes are produced. The distance may be predefined (e.g., Euclidean distance) or learned (e.g., Mahalanobis distance).

However, approaches based on feature extraction and metric learning present drawbacks (Musgrave *et al.*, 2020). First, ad hoc procedures are required after neural network training. Additionally, they usually present hyperparameters to tune, usually requiring unrealistic access to design-time OOD or adversarial samples.

Therefore, a possible option to build a seamless approach to the OOD detection is to design an isotropic (*exclusively* distance-based) loss that works as a SoftMax loss drop-in replacement. Hence, we obtain prototypes based classification while avoiding metric learning post-processing (Fig. 25, right).

3.1.2 Seamless Isotropic Loss

Let $\boldsymbol{f_\theta}(\boldsymbol{x})$ represent the high-level feature (embedding) associated with $\boldsymbol{x}$, $\boldsymbol{p}_{\boldsymbol{\phi}}^{j}$ represent the prototype associated with the class j, and $d()$ represent a distance function. To construct an isotropic loss, we need to avoid *direct* dependency on $\boldsymbol{f_\theta}(\boldsymbol{x})$ or $\boldsymbol{p}_{\boldsymbol{\phi}}^{j}$. Therefore, the loss has to be a function that *exclusively* depends on the embedding-prototype distances given by $d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^{j})$. Therefore, we can write:

$$\mathscr{L}_I = g(d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^{j})) \tag{3.2}$$

In the previous equation, $g()$ represents a scalar function. The expression $d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^{j})$ represents the isotropic layer, where its weights are given by the learnable prototypes $\boldsymbol{p}_{\boldsymbol{\phi}}^{j}$. We decided to normalize the embedding-prototype distances using the SoftMax *function* to allow interpretation in terms of probabilities. Therefore, the embedding-prototype distances represent the logits of the SoftMax function and correspond to the output of the isotropic layer. We also decided to use the cross-entropy for efficient optimization. Therefore, we can write the following:

$$\mathscr{L}_I(\hat{y}^{(k)}|\boldsymbol{x}) = -\log\left(\frac{\exp(-d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^{k}))}{\sum\limits_j \exp(-d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^{j}))}\right) \tag{3.3}$$

In the above equation, k represents the correct class, while the negative logarithm represents the cross-entropy. The negative terms before the distances are necessary to indicate the negative correlation between distances and probabilities. The expression between the outermost parentheses applied to the term $-d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^{j})$ represents the SoftMax function.

We need to choose a distance that allows IsoMax to work as a SoftMax drop-in replacement. Hence, the loss needs to learn *both* high-level features *and prototypes* using *exclusively* SGD and *end-to-end* backpropagation, as, based on the goal we defined for this work, *no additional offline procedures are allowed*. We emphasize that the prototypes are learned using backpropagation just like other weights. We also require the training using IsoMax loss to be as consistent and stable as the typical SoftMax loss neural network training.

Figure 26 - SoftMax Loss Drawbacks and IsoMax Loss Advantages

(a)

$$\mathcal{L}_S(\hat{y}^{(k)}|\boldsymbol{x}) = -\log\left(\frac{\exp(\overbrace{\boldsymbol{w}_k^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})+b_k}^{\text{Affine Transformation}})}{\sum_j \exp(\boldsymbol{w}_j^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})+b_j)}\right)$$

The inner product present in the SoftMax Loss generates separable features in the features space.

(b)

$$\mathcal{L}_I(\hat{y}^{(k)}|\boldsymbol{x}) = -\log\left(\frac{\exp(\overbrace{-d(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^k)}^{\text{Distance-Based Transformation}})}{\sum_j \exp(-d(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^j))}\right)$$

Rather than an affine transformation, the IsoMax Loss uses a distance-based transformation to impose isotropy around prototypes.

Source: The Author (2022). SoftMax loss produces separable features (above), while distance-based losses tend to generate discriminative features (below). Out-of-distribution examples are more discernible when in-distribution examples are concentrated around prototypes.

The covariance matrix makes it hard to use the Mahalanobis distance to train a neural network directly, as it contains components that are not differentiable. Therefore, we decide to use *Euclidean* distance. We have reasons to prefer the *nonsquared* Euclidean distance rather than the *squared* Euclidean distance. First, the nonsquared Euclidean distance is a real metric that obeys the Cauchy–Schwarz inequality while the squared Euclidean distance is not. Using a metric that follows the Cauchy–Schwarz inequality is essential because of our previous geometric considerations such as features separability and isotropy. Additionally, using the squared Euclidean distance is actually equivalent to using a linear model with a particular parameterization (Snell *et al.*, 2017; Mensink *et al.*, 2013), which we prefer to avoid increasing the representative power of the proposed loss.

Moreover, we experimentally observed that training neural networks using exclusively SGD and end-to-end backpropagation is more stable and consistent when using nonsquared Euclidean distance rather than squared Euclidean distance. Indeed, squared Euclidean distance-based logits are harder to *seamlessly* train than nonsquared Euclidean distance-based logits because numeric calculus instabilities are much more likely to occur when performing calculations and derivations with values of the order of $\mathscr{O}(e^{-d^2})$ than $\mathscr{O}(e^{-d})$.

Finally, even in the cases in which we were eventually able to seamlessly train neural networks using the squared Euclidean distance, we observed lower OOD detection performance than using the nonsquared Euclidean distance-based alternative. Therefore, we choose the nonsquared Euclidean distance.

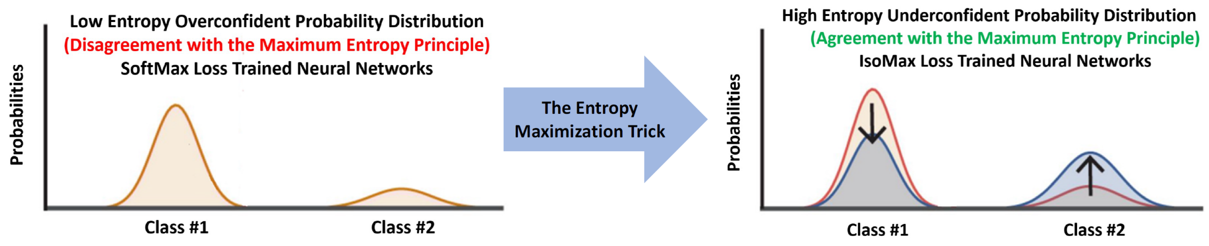
Unlike metric learning-based OOD detection approaches, rather than learning a metric from a preexisting feature space (metric learning on features extracted from a pretrained model), when using the IsoMax loss, we learn a feature space that is, from the start, consistent with the chosen metric. Indeed, the minimization of Equation 3.3 is achieved by making the expression inside the outer parentheses go to one. It is only possible by reducing the distances between the high-level features (embeddings) and the associated class prototypes, while simultaneously keeping high distances among class prototypes. Hence, the main aim of metric learning, which is to reduce intraclass distances while increasing interclass distances, is performed naturally during the neural network training, avoiding the need for feature extraction and metric learning post-processing phases (Fig. 26).

3.1.3 Principle of Maximum Entropy and Entropic Scale

Isotropy increases OOD detection performance (See Chapter 4). However, for further gains, we need to circumvent the cross-entropy extreme propensity to produce significantly low entropy posterior probability distributions. Additionally, as established in our goal, we need to achieve this without losing IsoMax seamlessness.

Figure 27 - The Entropic Maximization Trick

(a)


Low Entropy Overconfident Probability Distribution
(Disagreement with the Maximum Entropy Principle)
SoftMax Loss Trained Neural Networks
Probabilities
Class #1
Class #2
The Entropy Maximization Trick
High Entropy Underconfident Probability Distribution
(Agreement with the Maximum Entropy Principle)
IsoMax Loss Trained Neural Networks
Probabilities
Class #1
Class #2

(b)

IsoMax Loss

$$\mathscr{L}_I(\hat{y}^{(k)}|\boldsymbol{x}) = -\log\left(\frac{\exp(-E_s\left\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^k\right\|)}{\sum_j \exp(-E_s\left\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j\right\|)}\right)$$
$$= -\log\left(\frac{\exp(-E_s\sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^k)\cdot(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^k)})}{\sum_j \exp(-E_s\sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j)\cdot(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j)})}\right)$$

The entropic scale is present during training.

IsoMax Probabilities

$$p_I(y^{(i)}|\boldsymbol{x}) = \frac{\exp(-\left\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^i\right\|)}{\sum_j \exp(-\left\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j\right\|)}$$
$$= \frac{\exp(-\sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^k)\cdot(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^k)})}{\sum_j \exp(-\sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j)\cdot(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j)})}$$

The entropic scale is removed before inference.

The entropy maximization trick conciliates loss minimization and entropy maximization by multiplying the logits by a constant entropic scale present during training but removed before inference.

Source: The Author (2022). High entropy distributions (IsoMax Loss) may produce the same predictions and, consequently, classification accuracies of low entropy distributions (SoftMax Loss). However, the former provide a much higher OOD detection performance than the latter, regardless of using maximum probability or entropy as the score. SoftMax loss trained neural networks produce overconfident low-entropy probability distributions in disagreement with the maximum entropy principle. Our *entropy maximization trick*, which consists in training using logits multiplied by a constant factor called the entropic scale *that is nevertheless removed before inference*, enables IsoMax to generate underconfident high-entropy (almost maximum entropy) probability distributions in agreement with the principle of maximum entropy.

The principle of maximum entropy has been studied as a *regularization factor* (Dubey *et al.*, 2018; Pereyra *et al.*, 2017). In some cases, it has also been used as a direct optimization procedure without connection to cross-entropy minimization or backpropagation. For example, in Miller *et al.* (1995); Berger *et al.* (1996); Shawe-Taylor & Hardoon (2009), the maximization of the entropy subject to a constraint on the expected classification error was shown to be equivalent to solving an unconstrained Lagrangian. Although theoretically well grounded (Pearl, 1989; Williamson, 2005, 2009, 2013), *direct* entropy maximization presents high computational complexity (Pearl, 1989; Williamson, 2009).

Alternatively, modern neural networks are trained using computationally efficient cross-entropy. However, this procedure does not prioritize high entropy (low confident) posterior probability distributions. *Actually, the opposite is true*. Indeed, the minimization of cross-entropy has the undesired side effect of producing low entropy (overconfident) posterior probability distributions (Guo *et al.*, 2017). Unlike the previously mentioned works, we use the principle of maximum entropy neither to motivate the construction of regularization mechanisms, such as label smoothing or the confidence penalty (Dubey *et al.*, 2018; Pereyra *et al.*, 2017), nor to perform *direct* maximum entropy optimization (Miller *et al.*, 1995; Berger *et al.*, 1996; Shawe-Taylor & Hardoon, 2009). Indeed, the entropy is not even calculated during training.

In the opposite direction, we use the principle of maximum entropy as *motivation* to construct high-entropy (low-confident) posterior probabilities, still relying on computationally efficient cross-entropy minimization. Since our approach does *not directly* maximize the entropy, we cannot state that IsoMax produces the maximum entropy posterior probability distribution. However, the entropies of the probability distributions produced by IsoMax are high enough to improve the OOD detection performance significantly.

$$\mathscr{L}_{\text{SoftMax}} = -\log\left(\frac{\exp(L_k)}{\sum\limits_j \exp(L_j)}\right) \rightarrow 0 \tag{3.4}$$

$$\implies \mathscr{P}(y|\boldsymbol{x}) \rightarrow 1 \tag{3.5}$$

$$\implies \mathscr{H}_{\text{SoftMax}} \rightarrow 0 \tag{3.6}$$

Equation (3.4) describes the behavior of cross-entropy and entropy for the SoftMax loss. L_j represents the logits associated with class j, and L_k represents the logits associated with the correct class k. When minimizing the loss (Equation (3.4)), high probabilities are generated (Equation (3.5)). Consequently, significantly low entropy posterior probability distributions are produced (Equation (3.6)). Hence, the usual cross-entropy minimization tends to generate unrealistic overconfident (low entropy) probability distributions. Therefore, we have an opposition between cross-entropy minimization and the principle of maximum entropy.

$$\mathscr{L}_{\text{IsoMax}} = -\log\left(\frac{\exp(-E_s \times D_k)}{\sum\limits_j \exp(-E_s \times D_j)}\right) \rightarrow 0 \qquad (3.7)$$

$$\not\Rightarrow \mathscr{P}(y|\boldsymbol{x}) \rightarrow 1 \qquad (3.8)$$

$$\not\Rightarrow \mathscr{H}_{\text{IsoMax}} \rightarrow 0 \qquad (3.9)$$

The IsoMax loss conciliates these contradictory objectives (loss minimization and entropy maximization) by multiplying the logits by a constant positive scalar E_s, which is presented during training but removed before inference. Equation (3.7) demonstrates how the entropic scale (presented at training time but removed at inference time) allows the production of high entropy posterior distributions despite using cross-entropy minimization. D_j represents the distances associated with class j, and D_k represents the distances associated with the correct class k. The E_s present during training allows the term $-E_s \times D_k$ to become high enough (less negative compared to $-E_s \times D_j$) to produce a low loss (Equation (3.7)) *without* producing high probabilities for the correct classes, as they are calculated with the E_s removed (Equation (3.8)). Thus, it is possible to build posterior probability distributions with high entropy (Equation (3.9)) in agreement with the fundamental principle of maximum entropy despite using cross-entropy to minimize the loss (Fig. 27).

Therefore, returning to Equation (3.3), multiplying the embedding-prototype distances by E_s, and making $d()$ equal to the nonsquared Euclidean distance, we can write the definitive IsoMax loss as:

$$\begin{aligned}\mathscr{L}_I(\hat{y}^{(k)}|\boldsymbol{x}) &= -\log^{\dagger}\left(\frac{\exp(-E_s \left\|\boldsymbol{f_\theta}(\boldsymbol{x}) - \boldsymbol{p}^k_{\boldsymbol{\phi}}\right\|)}{\sum\limits_j \exp(-E_s \left\|\boldsymbol{f_\theta}(\boldsymbol{x}) - \boldsymbol{p}^j_{\boldsymbol{\phi}}\right\|)}\right)\\ &= -\log^{\dagger}\left(\frac{\exp(-E_s \sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x}) - \boldsymbol{p}^k_{\boldsymbol{\phi}})\cdot(\boldsymbol{f_\theta}(\boldsymbol{x}) - \boldsymbol{p}^k_{\boldsymbol{\phi}})})}{\sum\limits_j \exp(-E_s \sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x}) - \boldsymbol{p}^j_{\boldsymbol{\phi}})\cdot(\boldsymbol{f_\theta}(\boldsymbol{x}) - \boldsymbol{p}^j_{\boldsymbol{\phi}})})}\right)\end{aligned} \qquad (3.10)$$

[†]The probability (i.e., the expression between the outermost parentheses) and logarithm operations are computed sequentially and separately for higher OOD detection performance (see the source code).

We emphasize that removing the entropic scale after the training does not affect the ability of the solution to represent the prior knowledge available, as it does not change the predictions. Therefore, the expression for the probabilities with the entropic scale removed is preferable, as it remarkably increases the entropy of the posterior distributions in agreement with the principle of maximum entropy. Hence, inference probabilities for the IsoMax loss are defined as follows:

$$\begin{aligned} p_I(y^{(i)}|\boldsymbol{x}) &= \frac{\exp(-\left\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^i\right\|)}{\sum\limits_j \exp(-\left\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j\right\|)} \\ &= \frac{\exp(-\sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^k)\cdot(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^k)})}{\sum\limits_j \exp(-\sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j)\cdot(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j)})} \end{aligned} \tag{3.11}$$

3.1.4 Initialization and Implementation Details

We experimentally observed that using the common Xavier (Glorot & Bengio, 2010) and Kaiming (He *et al.*, 2015) initializations for prototypes makes the OOD detection performance oscillate. Sometimes it improves, sometimes it decreases. Hence, we decided to always initialize all prototypes to zero vector. It indeed makes sense, as this is the most natural value for untrained embeddings. Weight decay is applied to prototypes, as they are regular trainable parameters in our solution.

To calculate losses based on cross-entropy, deep learning libraries usually combine the logarithm and probability calculations into a single computation. However, we experimentally observed that sequentially computing these calculations as stand-alone operations improves IsoMax performance because it is more effective in keeping the initial output entropy high.

The class prototypes have the same dimension as the neural network last layer representations. Naturally, the number of prototypes is equal to the number of classes. Therefore, the IsoMax loss has fewer parameters than the SoftMax loss, as it has no bias to be learned. We remember that the prototypes are updated during the regular backpropagation procedure, just like any other parameters.

Finally, we verified classification accuracy drop and low or oscillating OOD detection performance when trying to integrate E_s with cosine similarity (Liu *et al.*, 2016; Wang *et al.*, 2018; Deng *et al.*, 2019) or the affine transformations used in SoftMax loss. The above trick (i.e., to initialize prototypes to zero vector) cannot be performed in cosine similarity and SoftMax loss cases. Therefore, our only option is to confirm that we need to use distance. Moreover, the *nonsquared* Euclidean distance as the best option to build the IsoMax loss.

3.1.5 Entropic Score

Out-of-distribution detection approaches typically define a score to be used after inference to evaluate whether an example should be considered OOD. In a seminal work, Shannon (1948a,b) demonstrated that entropy represents the optimum measure of the randomness of a source of symbols. More broadly, we currently understand entropy as a measure of the uncertainty. Therefore, considering that the uncertainty in classifying a specific sample should be an optimum metric to evaluate whether a particular example is OOD, we define our score to perform OOD detection, called the entropic score, as the *negative entropy* of the output probabilities:

$$\mathscr{E}\mathscr{S} = -\sum_{i=1}^{N} p(y^{(i)}|\boldsymbol{x}) \log p(y^{(i)}|\boldsymbol{x}) \tag{3.12}$$

Using the negative entropy as a score to evaluate whether a particular sample is OOD, we consider the information provided by *all* available network outputs rather than just one. For instance, ODIN and ACET only use the maximum probability, while the Mahalanobis method only uses the distance to the nearest prototype.

During IsoMax training, the embedding-prototype distances are affected. On the one hand, the distances from embeddings to the correct class prototype are reduced to increase classification accuracy. On the other hand, the distances from embeddings to the wrong class prototypes are increased. Consequently, based on the Equation (3.11), the probabilities of in-distribution examples increase. Therefore, it is reasonable to expect that samples with lower entropy more likely belong to the in-distribution. Additionally, using this predefined mathematical expression as score avoids the need to train an ad hoc additional regression model to detect OOD samples that is otherwise required, for example, in the Mahalanobis approach.

Even more important, since no regression model needs to be trained, there is no need for unrealistic access to OOD or adversarial samples for hyperparameter validation[2]. Since the entropic score is a predefined mathematical expression rather than a trainable model, it is available as soon as the neural network training finishes, avoiding additional training of ad hoc models in a post-processing phase.

In practice, we may set the threshold in two ways. First, we may collect random out-of-distribution samples and also use the train or validation set to define the threshold. This way, we may have an estimation of the performance we may expect in the field.

The second option is only using the train or validation set without access to a random out-of-distribution. In this case, we may set a threshold by making, for example, 95% of your in-distribution data to be considered in-distribution using the chosen threshold. In this case, however, we will not have an estimation of the expected performance. In this work, we followed the literature and often used threshold independent detection metrics to ensure the robustness of the score used.

[2]We call hyperparameter validation (or tuning) the procedure of selecting the model for test evaluation (i.e., model section) based on the performance achieved on the validation partition of a set of models trained with different hyperparameters.

3.2 ENHANCED ISOTROPY MAXIMIZATION LOSS

In this section, we present the enhanced IsoMax for training and the minimum distance score for inference. By combining those methods, we develop a seamless, scalable, and high-performance out-of-distribution detection approach.

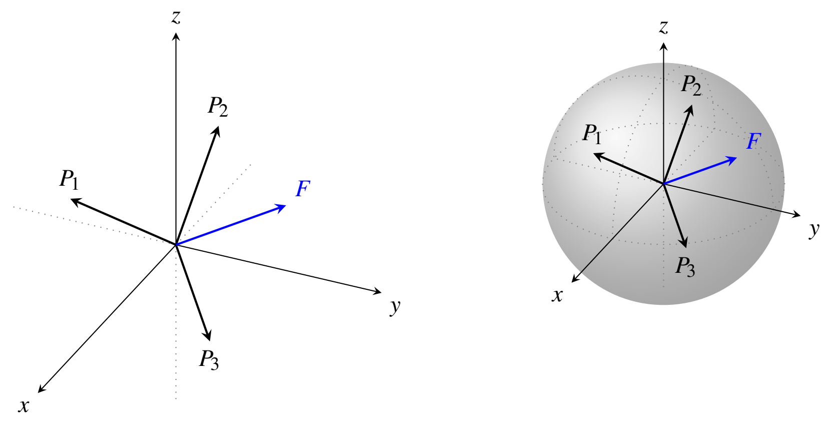
We started from the IsoMax loss. First, we normalize both the prototypes and the features. Second, we change the initialization of the prototypes. Third, we add the *distance scale*, which is a *learnable scalar parameter* that multiplies the *feature-prototype distances*. We call the combination of these three modifications the *isometrization* of the *feature-prototype distances*. We call the proposed modified version of IsoMax the Enhanced Isotropy Maximization Loss (IsoMax+). Finally, we use the *minimum feature-prototype distance* as a score to perform OOD detection. Considering that the minimum feature-prototype distance is calculated to perform the classification, *the OOD detection task presents essentially zero computational cost* because we simply reuse this value as a score to perform OOD detection.

IsoMax+ keeps the solution seamless (i.e., avoids the previously mentioned special requirements and side effects) while significantly increasing the OOD detection performance. Similar to IsoMax loss, IsoMax+ works as a *SoftMax loss drop-in replacement* because no procedures other than regular neural network training are required.

Figure 28 - Isometric Distances

(a) *IsoMax n-dimensional Euclidean space*

(b) *(n-1)-sphere of radius one in the IsoMax+ n-dimensional Euclidean space*



Source: The Authors (2022). The illustration presents the advent of the isometric distances in IsoMax+. P_1, P_2, and P_3 represent prototypes of classes 1, 2, and 3, respectively. F denotes the feature associated with a given image. IsoMax does not restrict prototypes and features to the (n-1)-sphere. In contrast, IsoMax+ does precisely this by rescaling prototypes and features.

3.2.1 Isometric Distance

We consider an input $\boldsymbol{x}$ applied to a neural network that performs a parametrized transformation $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$. We also consider $\boldsymbol{p}^j_{\boldsymbol{\phi}}$ to be the *learnable prototype* associated with class j. Additionally, let the expression $\left\|\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})-\boldsymbol{p}^j_{\boldsymbol{\phi}}\right\|$ represent the *nonsquared Euclidean distance* between $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $\boldsymbol{p}^j_{\boldsymbol{\phi}}$. Finally, we consider $\boldsymbol{p}^k_{\boldsymbol{\phi}}$ as the learnable prototype associated with the correct class for the input $\boldsymbol{x}$. Thus, we can rewrite for convenience the IsoMax loss (equation 3.10) using the equation below:

$$\mathscr{L}_{\text{IsoMax}} = -\log^{\dagger}\left(\frac{\exp(-E_s\left\|\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})-\boldsymbol{p}^k_{\boldsymbol{\phi}}\right\|)}{\sum\limits_j \exp(-E_s\left\|\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})-\boldsymbol{p}^j_{\boldsymbol{\phi}}\right\|)}\right) \tag{3.13}$$

In the above equation, E_s represents the entropic scale. From Equation 3.13, we observe that the distances from IsoMax loss are given by the expression $\mathscr{D}=\left\|\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})-\boldsymbol{p}^j_{\boldsymbol{\phi}}\right\|$. During inference, probabilities calculated based on these distances are used to produce the negative entropy, which serves as a score to perform OOD detection. However, because the features $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ are not normalized, examples with low norms are unjustifiably favored to be considered OOD examples because they tend to produce high entropy. Additionally, because the weights $\boldsymbol{p}^j_{\boldsymbol{\phi}}$ are not normalized, examples from classes that present prototypes with low norms are unjustifiably favored to be considered OOD examples for the same reason. Thus, we propose replacing

[†] The probability (i.e., the expression between the outermost parentheses) and logarithm operations are computed sequentially and separately for higher OOD detection performance (see the source code).

$\boldsymbol{f_\theta}(\boldsymbol{x})$ with its normalized version given by $\widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} = \boldsymbol{f_\theta}(\boldsymbol{x}) / \left\| \boldsymbol{f_\theta}(\boldsymbol{x}) \right\|$. Additionally, we propose replacing $\boldsymbol{p}_{\boldsymbol{\phi}}^j$ with its normalized version given by $\widehat{\boldsymbol{p}_{\boldsymbol{\phi}}^j} = \boldsymbol{p}_{\boldsymbol{\phi}}^j / \left\| \boldsymbol{p}_{\boldsymbol{\phi}}^j \right\|$. The expression $\|\boldsymbol{v}\|$ represents the 2-norm of a given vector $\boldsymbol{v}$.

However, while the distances in the original IsoMax loss may vary from zero to infinity, the distance between two normalized vectors is always equal to or lower than two. To avoid this unjustifiable and unreasonable restriction, we introduce the *distance scale* d_s, which is a *scalar learnable parameter*. Naturally, we require the distance scale to always be positive by taking its absolute value $|d_s|$.

Feature normalization makes the solution isometric regardless of the norm of the features produced by the examples. The distance scale is class independent because it is a *single* scalar value learnable during training. The weight normalization and the class independence of the distance scale make the solution isometric regarding all classes. The final distance is isometric because it produces an isometric treatment of all features, prototypes, and classes. Therefore, we can write the *isometric distances* used by the IsoMax+ loss as $\mathscr{D}_{\mathscr{I}} = |d_s| \left\| \widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}_{\boldsymbol{\phi}}^j} \right\|$. Returning to Equation (3.13), we can finally write the expression for the IsoMax+ loss as follows (see Algorithm 1 and Fig. 28):

$$\mathscr{L}_{\mathsf{IsoMax+}} = -\log^{\dagger\dagger}\left(\frac{\exp(-E_s\,|d_s| \left\| \widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}_{\boldsymbol{\phi}}^k} \right\|)}{\sum\limits_j \exp(-E_s\,|d_s| \left\| \widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}_{\boldsymbol{\phi}}^j} \right\|)}\right) \tag{3.14}$$

Applying the entropy maximization trick (i.e., the removal of the entropic scale E_s for inference) (Macêdo *et al.*, 2021), we can write the expression for the IsoMax+ loss probabilities used during inference for performing OOD detection when using the maximum probability score or the entropic score (Macêdo *et al.*, 2021):

$$\mathscr{P}_{\mathsf{IsoMax+}}(y^{(i)}|\boldsymbol{x}) = \frac{\exp(-\,|d_s| \left\| \widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}_{\boldsymbol{\phi}}^i} \right\|)}{\sum\limits_j \exp(-\,|d_s| \left\| \widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}_{\boldsymbol{\phi}}^j} \right\|)} \tag{3.15}$$

Different from IsoMax loss, where the prototypes are initialized to a zero vector, we initialized all prototypes using a normal distribution with a mean of zero and a standard deviation of one. This approach is necessary because we normalize the prototypes when using IsoMax+ loss. The distance scale is initialized to one, and we added no hyperparameters to the solution.

[††]The probability (i.e., the expression between the outermost parentheses) and logarithm operations are computed sequentially and separately for higher OOD detection performance (see Algorithm 1 and the source code).

Algorithm 1: PyTorch for the Enhanced IsoMax Loss.

```
class IsoMaxPlusLossFirstPart(nn.Module):
    """This part replaces the model classifier output layer nn.Linear()"""
    def __init__(self, num_features, num_classes):
        super(IsoMaxPlusLossFirstPart, self).__init__()
        self.num_features = num_features
        self.num_classes = num_classes
        self.prototypes = nn.Parameter(torch.Tensor(num_classes, num_features))
        nn.init.normal_(self.prototypes, mean=0.0, std=1.0)
        self.distance_scale = nn.Parameter(torch.Tensor(1))
        nn.init.constant_(self.distance_scale, 1.0)

    def forward(self, features):
        distances = torch.abs(self.distance_scale) * torch.cdist(
            F.normalize(features), F.normalize(self.prototypes),
            p=2.0, compute_mode="donot_use_mm_for_euclid_dist")
        logits = -distances
        return logits

class IsoMaxPlusLossSecondPart(nn.Module):
    """This part replaces the nn.CrossEntropyLoss()"""
    def __init__(self, entropic_scale = 10.0):
        super(IsoMaxPlusLossSecondPart, self).__init__()
        self.entropic_scale = entropic_scale

    def forward(self, logits, targets):
        """Probabilities and logarithms are calculated
        separately and sequentially"""
        """Therefore, nn.CrossEntropyLoss() must not be
        used to calculate the loss"""
        distances = -logits
        probabilities_for_training = nn.Softmax(dim=1)
        (-self.entropic_scale * distances)
        probabilities_at_targets =
        probabilities_for_training[range(distances.size(0)), targets]
        loss = -torch.log(probabilities_at_targets).mean()
        return loss
```

3.2.2 Minimum Distance Score

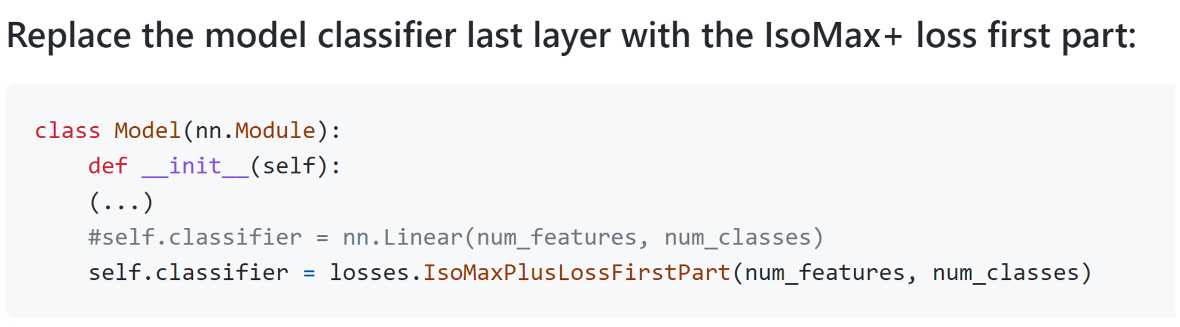
Motivated by the desired characteristics of the isometric distances used in IsoMax+, we use the minimum distance as score for performing OOD detection. Naturally, the Minimum Distance Score (MDS) for IsoMax+ is given by:

$$\text{MDS} = \min_j \left(\left\| \widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}^j_\phi} \right\| \right) \tag{3.16}$$

In this equation, $|d_s|$ was removed because, after training, it is a scale factor that does not affect the detection decision. Considering that the minimum distance is computed to perform the classification because the predicted class is the one that presents *the lowest feature-prototype distance*, when using the minimum distance score, *the OOD detection exhibits essentially zero latency and computational cost* because we simply reuse the minimum distance that was already calculated for classification purpose.

Figure 29 - IsoMax+ Loss PyTorch Code Usage

(a)



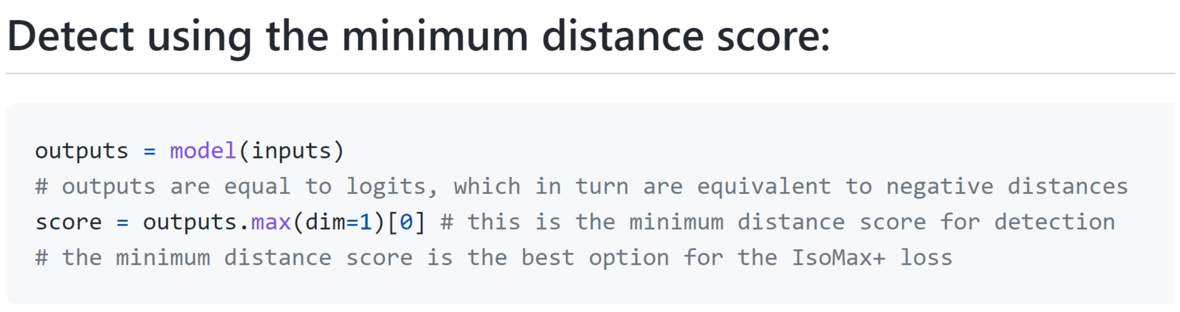
Replace the model classifier last layer with the IsoMax+ loss first part:

```
class Model(nn.Module):
    def __init__(self):
    (...)
    #self.classifier = nn.Linear(num_features, num_classes)
    self.classifier = losses.IsoMaxPlusLossFirstPart(num_features, num_classes)
```

Replace the criterion by the IsoMax+ loss second part:

```
#criterion = nn.CrossEntropyLoss()
criterion = losses.IsoMaxPlusLossSecondPart()
```

(b)

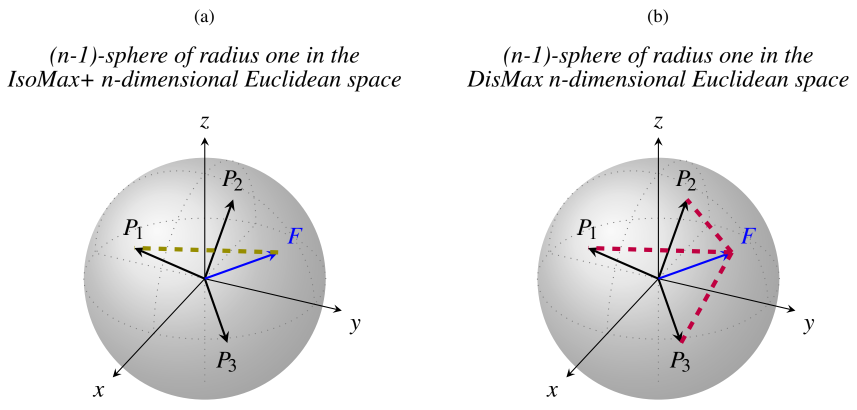


Detect using the minimum distance score:

```
outputs = model(inputs)
# outputs are equal to logits, which in turn are equivalent to negative distances
score = outputs.max(dim=1)[0] # this is the minimum distance score for detection
# the minimum distance score is the best option for the IsoMax+ loss
```

Source: The Author (2022). IsoMax+ Loss PyTorch Code Usage: a) Replace the last linear layer by the IsoMax+ loss code first part. Replace the cross-entropy loss with the IsoMax+ loss code second part. b) Detect using the minimum distance score.

Figure 30 - All-Distances-Aware Logits, Enhanced Logits, or Logits+


(a)
(n-1)-sphere of radius one in the IsoMax+ n-dimensional Euclidean space
(b)
(n-1)-sphere of radius one in the DisMax n-dimensional Euclidean space
z
P_2
P_1
F
y
P_3
x

Source: The Author (2022). The illustration presents the difference between IsoMax+ (Macêdo & Ludermir, 2021) and DisMax with respect to *logit formation*. P_1, P_2, and P_3 represent prototypes of classes 1, 2, and 3, respectively. F denotes the feature associated with a given image. Like all current losses, IsoMax+ constructs *each* logit associated with F considering its distance from a *single* prototype (olive dashed line). In contrast, DisMax loss builds *each* logit associated with F considering its distance from *all* prototypes (purple dashed lines). We use the terms *all-distances-aware* logits, *enhanced* logits, or logits+ indistinctly.

3.3 DISTINCTION MAXIMIZATION LOSS

We started from IsoMax+ loss (Macêdo & Ludermir, 2021) to construct the Distinction Maximization Loss (DisMax). First, we create the *enhanced* logits (logits+) by using *all* feature-prototype distances, rather than just the feature-prototype distance to the correct class. Second, we introduce the Fractional Probability Regularization (FPR) by minimizing the Kullback–Leibler (KL) divergence between the output probability distribution associated with a *compound* image and a target probability distribution containing *fractional* probabilities. The motivation of training on fractional probabilities is to force the neural network present outputs with higher entropies.

We call DisMax dagger (DisMax†) the variant of our loss when using FPR. Otherwise, we simply call it DisMax. Third, we construct a *composite* score for OOD detection that combines three components: the maximum logit+, the *mean* logit+, and the entropy of the network output. Fourth, we present a simple and fast temperature-scaling procedure that allows DisMax trained models to produce a high-performance uncertainty estimation. Like IsoMax+, DisMax works as a drop-in replacement for SoftMax loss. The trained models keep deterministic neural network inference efficiency.

3.3.1 All-Distances-Aware Logits

In IsoMax loss variants (e.g., IsoMax and IsoMax+), logits are formed from distances and are commonly used to calculate the score to perform OOD detection. Hence, it is essential to build logits that contain semantic information relevant to separating in-distribution (ID) from OOD during inference. IsoMax+ uses *isometric* distances (Macêdo & Ludermir, 2021).

In IsoMax+, the logits are simply the negatives of the isometric distances. We have two motivations to add the *mean* isometric distance considering *all* prototypes to the isometric distance associated with *each* class to construct what we call *all-distances-aware* logits, *enhanced* logits, or logits+.

First, considering that IsoMax+ is an isotropic loss, the pairwise distances between the prototypes and ID examples are forced to become increasingly smaller. Therefore, after training, it is reasonable to believe that ID feature-prototype distances are, on average, smaller than the distances from the prototypes to OOD samples, which were not forced to be closer to the prototypes. Hence, adding the mean distance to the logits used in IsoMax+ can help distinguish between ID and OOD more effectively. Second, taking *all* feature-prototype distances to compose the logits makes them a more stable information for OOD detection (Fig. 30).

$$\overbrace{L_+^j}^{\text{all-distances-aware logit for the j-th class}} = -\left(\underbrace{D_I^j}_{\text{isometric distance to the j-th class prototype}} + \overbrace{\frac{1}{N}\sum_{n=1}^{N} D_I^n}^{\text{mean isometric distance to all prototypes}}\right) \tag{3.17}$$

$$\overbrace{\mathscr{P}_{\text{DisMax}}(y^{(i)}|\boldsymbol{x})}^{\text{predicted probability distribution}} = \frac{\exp(E_s \overbrace{L_+^i}^{\text{all-distances-aware logit for the i-th class}} /T)}{\sum_{j=1}^{N} \exp(E_s \underbrace{L_+^j}_{\text{all-distances-aware logit for the j-th class}} /T)} \tag{3.18}$$

Therefore, we consider an input $\boldsymbol{x}$ and a network that performs a transformation $\boldsymbol{f_\theta}(\boldsymbol{x})$. We also consider $\boldsymbol{p}_{\boldsymbol{\phi}}^j$ to be the learnable prototype associated with class j. Moreover, considering that $\|\boldsymbol{v}\|$ represents the 2-norm of a vector $\boldsymbol{v}$, and $\widehat{\boldsymbol{v}}$ represents the 2-norm normalization of $\boldsymbol{v}$, we can write the *isometric distance* relative to class j as $D_I^j = |d_s| \left\| \widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}_{\boldsymbol{\phi}}^j} \right\|$, where $|d_s|$ represents the absolute value of the *learnable* scalar called distance scale (Macêdo & Ludermir, 2021). Finally, we can write the proposed *enhanced* logit for class j using the equation (3.17). N is the number of classes. Probabilities are given by equation (3.18), where T is the temperature. E_s is the entropic scale, *which is removed after training* (Macêdo *et al.*, 2021, 2022; Macêdo & Ludermir, 2021), *but before calibration*. For the rest of this section, distance means *isometric* distance.

3.3.2 Fractional Probability Regularization

We often train neural networks using *unitary* probabilities. Indeed, the usual cross-entropy loss forces a *probability equal to one* on a given training example. Consequently, we commonly train neural networks by providing a tiny proportion of points in the learning manifold. Hence, we propose what we call *fractional* probability regularization (FPR). The idea is to force the network to learn more diverse points in the learning manifold. Consequently, we confront target and predicted probability distributions also on *fractional* probability values rather than only *unitary* probability manifold points.

$$\overbrace{\mathcal{Q}_{\mathsf{Target}}(y^{(i)}|\tilde{\boldsymbol{x}})}^{\text{desired probability distribution for the compound image}} = \underbrace{\frac{1}{4}\sum_{m=1}^{4}\delta[y^{(i)}-y^{(j^m)}]}_{\text{sum one quarter to the probability of each class in the compound image}} \tag{3.19}$$

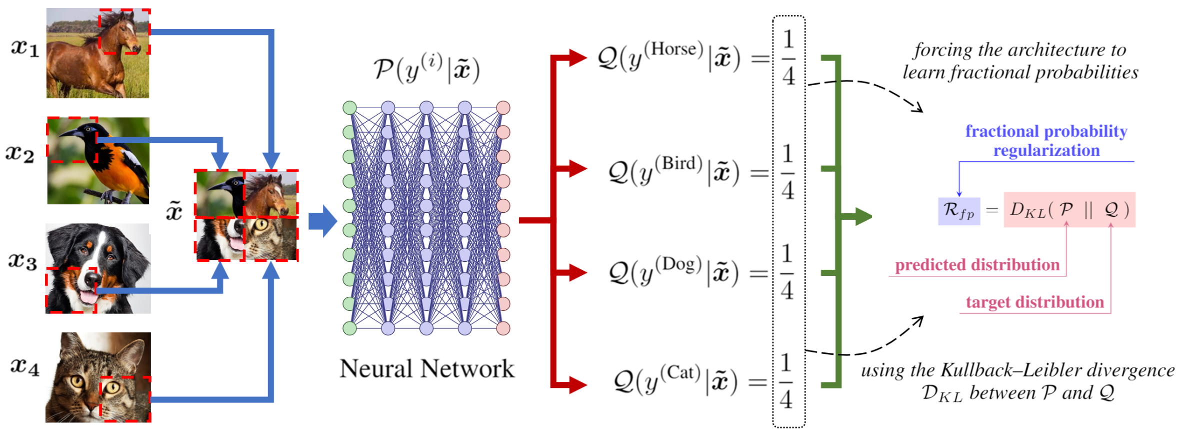
$$\overbrace{\mathcal{L}_{\mathsf{DisMax}}}^{\text{loss function}} = -\log^{*}\left(\frac{\exp(E_s\,\overbrace{L_{+}^{k}}^{\text{enhanced logit for the correct class k}})}{\sum\limits_{j}\exp(E_s\,\underbrace{L_{+}^{j}}_{\text{enhanced logit for the j-th class}})}\right) + \alpha\cdot\underbrace{D_{KL}(\mathscr{P}_{\mathsf{DisMax}}(y|\tilde{\boldsymbol{x}})\,||\,\mathcal{Q}_{\mathsf{Target}}(y|\tilde{\boldsymbol{x}}))}_{\text{fractional probability regularization}} \tag{3.20}$$

Therefore, our batch is divided into two halves. In the first half, we use the regular *unitary* probability training. For the second batch, we construct images specifically composed of patches of four others (Fig. 31). We construct our target probability distribution Q for those images by adding a quarter probability for each class corresponding to a patch of the compound image. Finally, we minimize the KL divergence regularization between our predicted and target probability distributions in the second half. These procedures do not increase training memory size requirements. Considering δ the Kronecker delta function[3], equation (3.19) represents the FPR in math terms. By combining the *enhanced* logits and the FPR, equation (3.20) presents the mathematical expression of the DisMax loss. Considering OOD is related to the uncertainty estimation, we validate α without requiring access to OOD data by choosing the pretrained model that produces the lower expected calibration error (ECE) after the temperature calibration.

[*] The probability (i.e., the expression between the outermost parentheses) and logarithm operations are computed sequentially and separately for optimal OOD detection performance.

[3] https://en.wikipedia.org/wiki/Kronecker_delta

Figure 31 - Fractional Probability Regularization

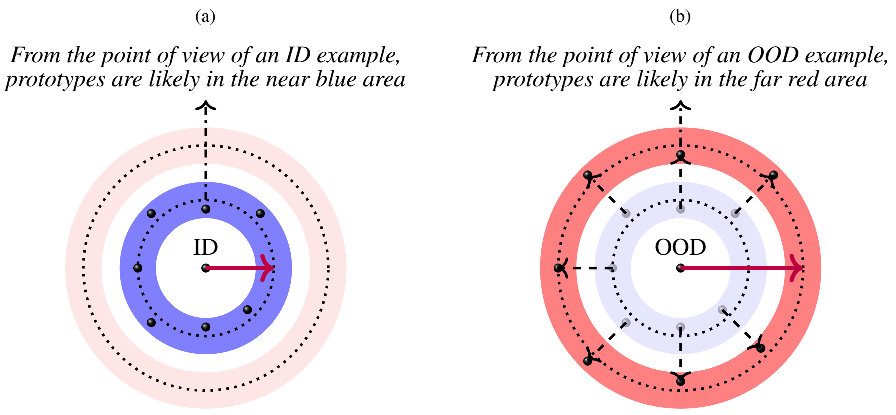

x_1
x_2
x_3
x_4
$\tilde{x}$
$\mathcal{P}(y^{(i)}|\tilde{x})$
Neural Network
$\mathcal{Q}(y^{(\text{Horse})}|\tilde{x}) = \frac{1}{4}$
$\mathcal{Q}(y^{(\text{Bird})}|\tilde{x}) = \frac{1}{4}$
$\mathcal{Q}(y^{(\text{Dog})}|\tilde{x}) = \frac{1}{4}$
$\mathcal{Q}(y^{(\text{Cat})}|\tilde{x}) = \frac{1}{4}$
forcing the architecture to learn fractional probabilities
fractional probability regularization
$\mathcal{R}_{fp} = D_{KL}(\mathcal{P} \,||\, \mathcal{Q})$
predicted distribution
target distribution
using the Kullback–Leibler divergence $\mathcal{D}_{KL}$ between $\mathcal{P}$ and $\mathcal{Q}$

Source: The Author (2022). We use images composed of patches of four randomly selected training examples. The KL divergence regularization term forces the network to predict *fractional* probabilities on compound images.

We recognize a similarity between CutMix (Yun *et al.*, 2019) and FPR: both are based on the combination of images to create compound data. However, we identify many differences. CutMix combines two images, while FPR combines four images. Moreover, the combination procedure is entirely different. In CutMix, a portion of an image is replaced by a patch of *variable* size, format, and position that comes from another image. In FPR, patches of the *same* size, format, and *predefined* positions from four different images are combined into a single one. This simplification introduced by FPR allowed us to combine four images instead of only two. Indeed, trying to extend CutMix by replacing portions of an image with patches from three others while allowing *random* sizes, shapes, and positions produces *patch superpositions*, making *the calculation of the pairwise ratio of the areas of the superposed patches extremely hard*. Therefore, this simplification made it possible to simultaneously combine four rather than only two images, in addition to avoiding the beta distribution and the α hyperparameter. In FPR, *fractional* probabilities, rather than losses, are proportional to areas.

While CutMix is applied randomly to some batches with probability p, FPR is applied to half of each batch, avoiding loss or gradient oscillations. CutMix neither creates a target distribution containing *fractional* probabilities nor forces the predicted probabilities to follow it by minimizing the KL divergence between them. Indeed, CutMix does *not* use the KL divergence at all. CutMix calculates the regular cross-entropy loss of the compound image considering the labels of the original images and takes a linear interpolation between the resulting loss values weighted by the ratio of the areas of the patch and the remaining image. While CutMix operates on losses, FPR operates directly on probabilities *before* calculating loss values. The concept of *fractional* probabilities is not even present in CutMix. Unlike CutMix, the mentioned procedure can be easily expanded to combine even more than four images. Finally, CutMix *increases the training time* and *presents hyperparameters* (Yun *et al.*, 2019).

Figure 32 - Max-Mean Logit Entropy Score


(a)
From the point of view of an ID example, prototypes are likely in the near blue area
ID
(b)
From the point of view of an OOD example, prototypes are likely in the far red area
OOD

Source: The Author (2022). In addition to the maximum logit and the negative entropy, the MMLES incorporates the *mean* logit+. We observed that the prototypes are generally closer to ID samples than OOD samples, which is true *regardless of whether the ID sample belongs to the class of the considered prototype*. Hence, incorporating this *all-distances-aware* information increases the OOD detection performance (see Chapter 4).

Even if we consider adapting the mosaic augmentation from the objection detection task (Bochkovskiy *et al.*, 2020) to classification and add to it a CutMix-like regularization, most of the previously mentioned differences between FPR and CutMix still holds to differentiate FPR from this CutMix-like classification-ported mosaic augmentation.

3.3.3 Max-Mean Logit Entropy Score

For OOD detection, we propose a score composed of three parts. The first part is the maximum logit+. The second part is the *mean* logit+. Incorporating the mean value of the logits into the score is an *independent procedure relative to the logit formation*. It can be applied regardless of the type of logit (e.g., usual or enhanced) used during training. Finally, we subtract the entropy calculated considering the probabilities of the neural network output. We call this *composite* score Maximum-Mean Logit Entropy Score (MMLES). It is given by equation (3.21). We call MMLES a *composite* score because it is formed by the sum of many other scores.

$$\overbrace{\mathscr{S}_{\text{MMLES}}}^{\text{max-mean logit entropy score}} = \underbrace{\max_j(L_+^j)}_{\text{maximum logit+}} + \underbrace{\frac{1}{N}\sum_{n=1}^{N} L_+^n}_{\text{mean logit+}} - \overbrace{\mathscr{H}(\mathscr{P}_{\text{DisMax}})}^{\text{entropy}} \tag{3.21}$$

The motivation for the MMLES score is presented in Fig. 32. The experiments presented in Chapter 4 show that, as explained in the mentioned figure, prototypes are usually closer to the in-distrbutions than to out-of-distributions.

3.3.4 Temperature Calibration

Unlike the usual SoftMax loss, the IsoMax loss variants produce *underestimated* probabilities *to obey the Maximum Entropy Principle* (Macêdo *et al.*, 2021; Macêdo *et al.*, 2019; Macêdo *et al.*, 2022; Macêdo & Ludermir, 2021). Therefore, *we need to perform a temperature calibration after training to improve the uncertainty estimation*. To find an optimal temperature, which was kept equal to one during training, we used the L-BFGS-B algorithm with approximate gradients and bounds equal to 0.001 and 100 (Byrd *et al.*, 1995; Zhu *et al.*, 1997; Morales & Nocedal, 2011) for ECE minimization on the validation set. This calibration takes only a few seconds in our experiments.

Therefore, we introduced the temperature and calibrated it to improve our proposed solutions' uncertainty estimation. We choose to tackle the uncertainty estimation problem using temperature calibration because Guo *et al.* (2017) showed that this is a simple and effective approach. Our experiments showed that mixing our modified technique for temperature calibration with our entropic losses produced state-of-the-art results.

4 EXPERIMENTS

"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong."

–Richard Feynman

"Whatever you do, love it like you loved that projection booth of the Paradiso when you were little..."

–Alfredo (Cinema Paradiso)

"Similar to humans, deep neural networks also suffer from the Dunning–Kruger effect."

–David Macêdo

"Everything I know about fighting, I know from you. What do we fight for? I ask myself, what were you fighting for? And was it worth it?"

–Arminius (Barbarians from Netflix)

In this chapter, we compare our entropic losses with the main state-of-the-art approaches. The experiments are presented in an evolutionary way. We start with IsoMax experiments. Then, we present IsoMax+ experiments. Finally, we present the DisMax experiments. The code to reproduce all experiments of this work is available online[1,2].

[1]https://github.com/dlmacedo/entropic-out-of-distribution-detection
[2]https://github.com/dlmacedo/distinction-maximization-loss

4.1 ISOTROPY MAXIMIZATION LOSS

We trained from scratch 100-layer DenseNetBCs (growth rate $k = 12$) (Huang *et al.*, 2017), 34-layer ResNets (He *et al.*, 2016), and 28-layer WideResNets (widening factor $k = 10$) (Zagoruyko & Komodakis, 2016). We trained on CIFAR10 (Krizhevsky, 2009), CIFAR100 (Krizhevsky, 2009), SVHN (Netzer & Wang, 2011), and TinyImageNet (Deng *et al.*, 2009) using SoftMax and IsoMax losses.

We used SGD with the Nesterov moment equal to 0.9 with a batch size of 64 for CIFAR10, CIFAR100, and TinyImageNet. We used an initial learning rate of 0.1. The weight decay was 0.0001, and we did not use dropout. We trained during 300 epochs for CIFAR10 and CIFAR100; and during 90 epochs for TinyImageNet. We used a learning rate decay rate equal to ten applied in epoch numbers 150, 200, and 250 for CIFAR10 and CIFAR100; and 30 and 60 for TinyImageNet. These values are commonly used in papers in general that train these models in these datasets. They were used in OOD detection papers (Liang *et al.*, 2018; Lee *et al.*, 2018).

We used resized images from the TinyImageNet (Deng *et al.*, 2009)[3] and the Large-scale Scene UNderstanding (LSUN) (Yu *et al.*, 2015)[3] as OOD data. Finally, we separately added these OOD data to the ID validation sets to construct the respective OOD detection test sets. For example, the OOD detection test set for ID CIFAR10 and OOD TinyImageNet is composed by combing the CIFAR validation set and the TinyImageNet OOD data. In some cases, CIFAR10 and SVHN work as OOD and the respective validation set is used to construct the OOD detection test set. We emphasize that our solution does not require validation sets for training or validation. In the following subsection, we provide extensive justifications for why validating the entropic scale does not produce significantly different performance. Thus, validation sets are used exclusively for building OOD detection test sets.

For text data experiments, we followed the experimental setting presented in Hendrycks *et al.* (2019a). Therefore, we used the 20 Newsgroups as in-distribution data. The 20 Newsgroups is a text classification dataset of newsgroup documents. It has 20 classes and approximately 20,000 examples that are split evenly among the classes. We used the standard 60/40 train/test split. We used the IMDB, Multi30K, and Yelp Reviews datasets as out-distribution. IMDB is a dataset of movie review sentiment classification. Multi30K is a dataset of English-German image descriptions, of which we use the English descriptions. Yelp Reviews is a dataset of restaurant reviews. We trained 2-layer GRUs (Cho *et al.*, 2014) using SoftMax and IsoMax losses.

[3]`https://github.com/facebookresearch/odin`

We evaluated the performance using three detection metrics. First, we calculated the True Negative Rate at 95% True Positive Rate (TNR@TPR95) using the adequate threshold. In addition, we evaluated the Area Under the Receiver Operating Characteristic Curve (AUROC) and the Detection Accuracy (DTACC), which corresponds to the maximum classification probability over all possible thresholds δ:

$$
\begin{aligned}
1-\min_{\delta}\Big\{&P_{\texttt{in}}\big(o(\mathbf{x})\leq\delta\big)\,P(\mathbf{x}\text{ is from }P_{\texttt{in}})\\
&+P_{\texttt{out}}\big(o(\mathbf{x})>\delta\big)\,P(\mathbf{x}\text{ is from }P_{\texttt{out}})\Big\},
\end{aligned}
\tag{4.1}
$$

where $o(\mathbf{x})$ is a given OOD detection score. It is assumed that both positive and negative samples have equal probability. This is the way this metric is defined and used in the literature. AUROC and DTACC are threshold independent.

4.1.1 Entropic Scale, High Entropy, and Out-of-Distribution Detection

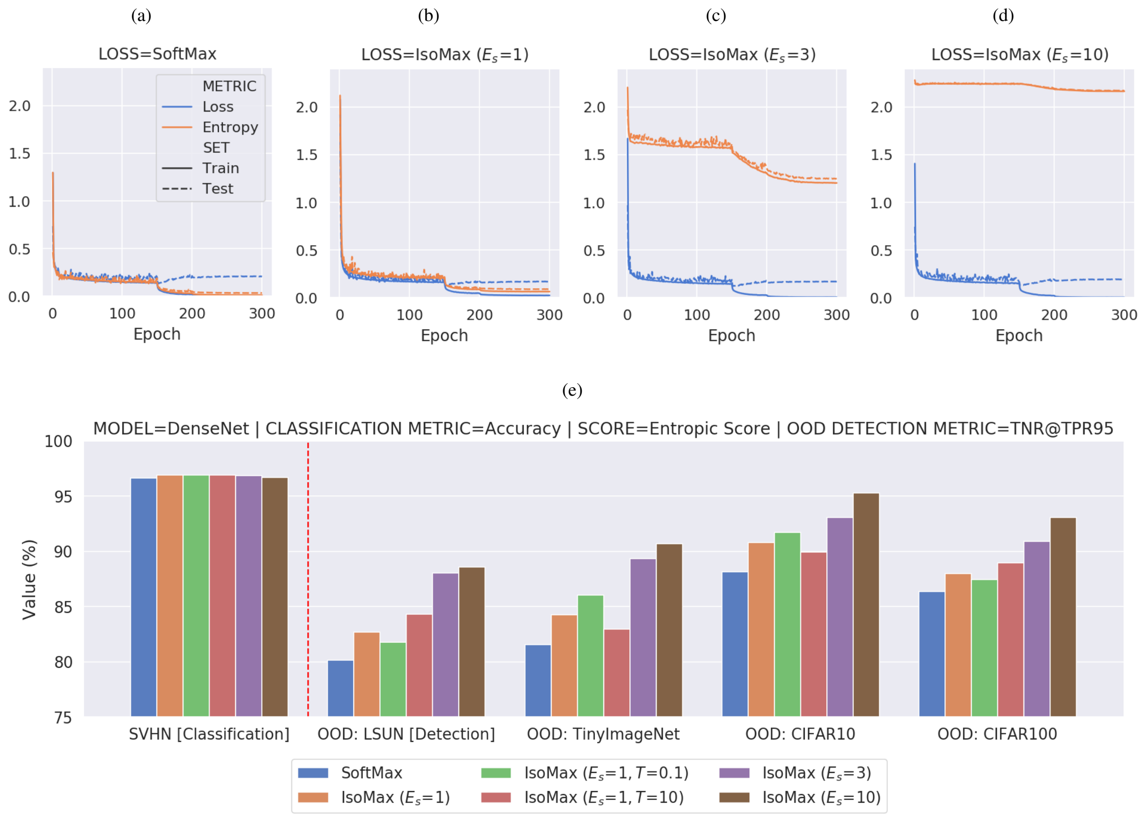
To experimentally show that higher entropic scales lead to higher entropy posterior probability distributions and improved OOD detection performance, we trained DenseNets on SVHN using the SoftMax loss and IsoMax loss with distinct entropic scale values. We used the entropic score and the TNR@TPR95 to evaluate OOD detection performance (Fig. 33).

As expected, Fig. 33a shows that the SoftMax loss generates posterior distributions with extremely low entropy. Fig. 33b illustrates that the unitary entropic scale (E_s= 1) does not increase the posterior distribution mean entropy. In other words, isotropy alone is not enough to circumvent the cross-entropy propensity to produce low entropy posterior probability distributions and the entropic scale is indeed necessary. Nevertheless, the replacement of anisotropic affine-based logits by isotropic distance-based logits is enough to produce initial OOD detection performance gains regardless of the out-distribution (Fig. 33e).

Fig. 33c shows that an intermediate entropic scale (E_s= 3) provides medium entropy probability distributions with corresponding additional OOD detection performance gains for all out-distributions (Fig. 33e). Fig. 33d illustrates that a high entropic scale (E_s= 10) produces even higher entropy probability distributions and the highest OOD detection performance regardless of the out-distribution considered (Fig. 33e). Despite the entropy scale value used, the cross-entropy is minimized and the classification accuracies produced by SoftMax and IsoMax losses are extremely similar.

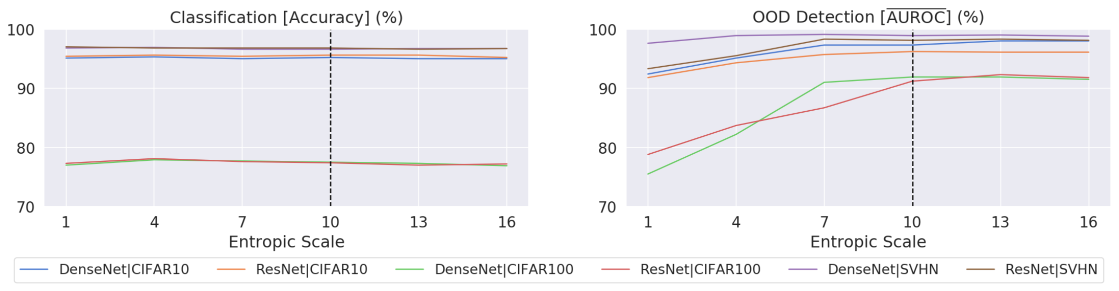
Hence, for a high entropic scale, the IsoMax loss can indeed minimize the cross-entropy while producing high entropy posterior probability distributions as recommended by the principle of maximum entropy. More importantly, higher entropy posterior probability distributions directly correlate with increased OOD detection performances despite the OOD data. An entropic scale E_s= 10 is enough to produce significantly high entropy probability distributions. Additionally, we experimentally observed no further gains for entropic scales higher than ten.

Figure 33 - IsoMax Loss Effects



Source: The Author (2022). (a) SoftMax loss simultaneously minimizes both the cross-entropy and the entropy of the posterior probabilities. (b) IsoMax loss produces low entropy posterior probabilities for a low entropic scale (E_s=1). (c) IsoMax loss produces medium mean entropy for an intermediate entropic scale (E_s=3). (d) IsoMax loss minimizes the cross-entropy while producing high mean entropies for the high entropic scale (E_s=10). *"The entropy maximization trick"* is the fundamental mechanism that allows us to migrate from low entropy posterior probability distributions (a,b) to high entropy posterior probability distributions (d). Higher entropic scale values correlate to higher mean entropies as recommended by the principle of maximum entropy. Regardless of training with a high entropic scale, if we do not remove it for inference ("the entropy maximization trick"), the IsoMax loss always produces posterior probability distributions with entropies as low as those generated by the SoftMax loss. An entropic scale equal to ten is sufficient to virtually produce the maximum possible entropy posterior probability distribution, as the highest possible value of the entropy is $\log(N)$, where N is the number of classes. (e) The left side of the dashed vertical red line presents classification accuracies. The dashed vertical red line right side shows OOD detection performance using the entropic score and the TNR@TPR95 metric. Higher mean entropies produce increased OOD detection performance regardless of the out-distribution. We emphasize that OOD examples were never used during training and no validation was required to tune hyperparameters. Additionally, isotropy enables IsoMax loss to produce higher OOD performance than SoftMax loss, even for the unitary value of the entropic scale. Training using E_s=1 and then making the temperature T=0.1 or T=10 during inference produces lower OOD detection performance than training using E_s=10 and removing it for inference, which consists in our proposal. Finally, IsoMax loss presents similar classification accuracy compared with SoftMax regardless of the entropic scale. The entropic score, which is the negative entropy of the network output probabilities, was used as the score.

Figure 34 - Accuracy and Out-of-Distribution Detection Study



Source: The Author (2022). Study of classification accuracy and OOD detection performance dependence on the entropic scale. $\overline{\text{AUROC}}$ represents the mean AUROC considering all out-of-distribution data. The classification accuracy and the mean OOD detection performance are approximately stable for $E_s = 10$ or higher regardless of the dataset and model. It also does not depend on the number of training classes N, which is probably explained by the fact that the entropic scale is inside an exponential function while the entropy increases only with the logarithm of the number of classes. Making E_s learnable did not considerably improve or decrease the OOD detection results.

Fig. 34 shows that *regardless of the combination of dataset and model*, the classification accuracy and the mean OOD detection performance are stable for $E_s = 10$ or higher, as the entropic scale is already high enough to ensure near-maximal entropy. We speculate that $E_s = 10$ worked satisfactorily regardless of the number of training classes because it is used inside an exponential function, while the entropy increases only with the logarithm of the number of training classes. Hence, an eventual validation of E_s would produce an insignificant performance increase. Actually, this is not possible because we consider access to OOD or outlier samples forbidden. Moreover, making E_s learnable did not significantly affect the OOD detection performance.

Considering that the entropic scale E_s is present in Equation 3.10, we are initially led to think that it needs to be tuned to achieve the highest possible OOD detection performance. However, we emphasize that the experiments showed that the dependence of the OOD detection performance on the entropic scale is remarkably well-behaved. Essentially, the OOD detection performance monotonically increases with the entropic scale until it reaches a saturation point near $E_s = 10$ regardless of the dataset or the number of training classes (Fig. 34). It may be explained by the fact that the entropic scale is inside an exponential function and that we experimentally observed that $\exp(-10 \times d)$ is enough to produce almost maximum entropy regardless of the dataset, model or number of training classes under consideration.

Finally, even if we consider that validating for entropic scales higher than ten would allow some minor OOD detection performance improvement, we usually cannot do this because we do not often have access to out-distribution data. Hence, the well-behaved dependence of the OOD detection performance on the entropic scale allowed us to define the E_s as a *constant scalar* equals to 10 rather than a hyperparameter that needs to be tuned for each novel dataset and model. It is the reason our approach may be used without requiring access to OOD/outlier data. Therefore, we kept the entropic scale as a constant (global value) equal to ten for all subsequent experiments.Hence, no validation of the entropic score was need for different models, datasets, or number of classes.

The entropic scale is constant regardless of the data and model considered. Hence, our approach does not require hyperparameter tuning or extra validation data. We have experimental evidence and theoretical insights to explain why this generalizes well. This constant value is used regardless of the model, in-distribution, and out-of-distribution.

The entropic scale is inside an exponential function, so the value of ten is enough to obtain almost the maximum entropy possible. Moreover, increasing this value further does not help, as the maximum possible entropy is already achieved. Therefore, we observe that gains in OOD detection performance increase fast in the beginning and saturate after some point. For example, consider the d as a distance. Hence, e^{-1d} and e^{-3d} are not as small as e^{-10d}. However, e^{-10d} and e^{-20d} are too small numbers to make any difference. This behavior is monotonic in relation to the entropic scale because we deal with a monotonic function: the exponential. A substantial amount of experiments was performed to confirm this experimentally. This behavior is the same regardless of the dataset and model.

In machine learning and deep learning, we have many examples of constant global values (i.e., they do not need to be tuned) regardless of the data and model used. For example, in the ADAM (Kingma & Ba, 2015) optimizer, we all use β_1 equals 0.9, and β_2 equals 0.99. When using SGD with momentum, we commonly use the moment equals 0.9. These are experimentally recognized as values that work well across essentially all circumstances. Like those values, many experiments have demonstrated that the entropic scale equals ten works well.

Consequently, we do not need extra validation data for tuning it, as the same constant value is used regardless of the data and models used for training. Besides the experimental evidence, this value is inside an exponential function gives theoretical insights to justify this fact.

Table 3 - Implementation Detail: Separating Logarithms from Probabilities.

Data Partition	Loss Value LogProb / Log+Prob	Mean Maximum Probability LogProb / Log+Prob	Mean Normalized Entropy LogProb / Log+Prob	Classification Accuracy (%) LogProb / Log+Prob
Train	1.697978 / 1.448191	0.308275 / 0.144888	0.796333 / 0.987172	36.25 / 48.34
Test	1.379901 / 1.487199	0.479694 / 0.167632	0.628158 / 0.979525	49.06 / 53.30

Source: The Author (2022). IsoMax results for DenseNet trained one epoch on CIFAR10. Normalized entropy means the entropy divided by the maximum entropy. LogProb means using logarithms and probabilities combined into a single operation. Log+Prob means using logarithms and probabilities as separated and sequential operations, which is the option adopted in all entropic losses. We made the code deterministic for these experiments; therefore, many executions of the same experiment produce the same results. We kept everything else the same in both LogProb and Log+Prob cases.

4.1.2 Implementation Detail: Separating Logarithms from Probabilities

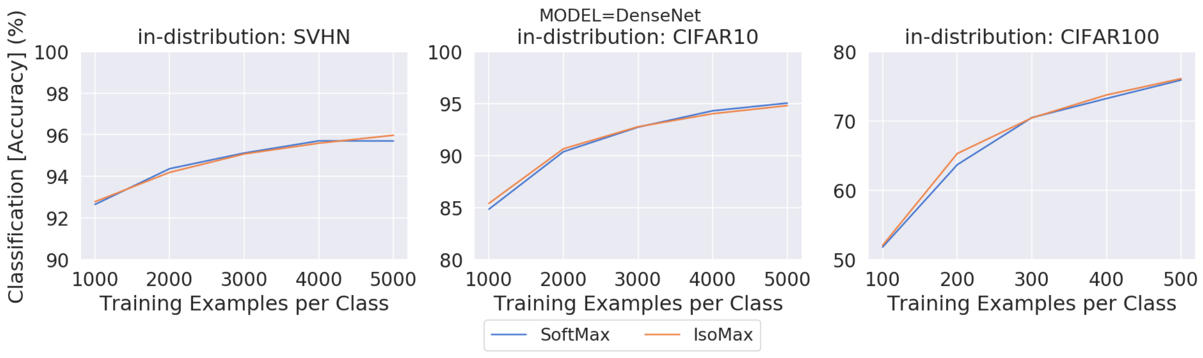
Table 3 shows that computing logarithms separated from probabilities is much more efficient in preserving the maximum normalized entropy present at the beginning of the training. Therefore, the implementations of all entropic losses calculate logarithms and probabilities as two separated and sequential operations.

4.1.3 Classification Accuracy

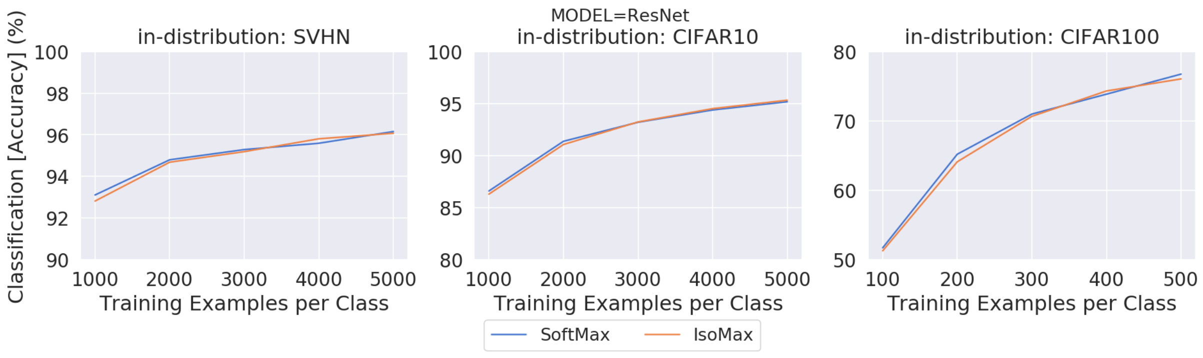
Table 4 shows that IsoMax loss consistently produces classification accuracy similar to SoftMax loss, regardless of being used as a baseline approach or combined with additional techniques to improve OOD detection performance. Fig. 35 shows that this is also true even for a varying number of training examples per class.

Figure 35 - Classification Accuracy: SoftMax vs. IsoMax

(a)



(b)



Source: The Author (2022). IsoMax loss presents test accuracy similar to SoftMax loss (no classification accuracy drop) for different numbers of training examples per class on several datasets and models. Simultaneously, IsoMax usually produces much higher OOD detection performance (Fig. 36).

4.1.4 Out-of-Distribution Detection

The IsoMax loss enhanced versions almost always outperform the OOD detection performance of the corresponding SoftMax loss enhanced version when both are using the same add-on technique. The major drawback of adding label smoothing or center loss regularization is the need to tune the hyperparameters presented by these add-on techniques. Additionally, different hyperparameters values need to be validated for each pair of in-distribution and out-distribution. As mentioned before, it is highly optimistic to assume access to OOD data, as we usually do not know what OOD data the solution we will face in the field.

Even considering this best-case scenario, SoftMax loss combined with label smoothing or center loss regularization always presented significantly lower OOD detection performance than IsoMax loss without using them and, consequently, avoiding unrealistically optimist access to OOD data and the mentioned validations. Enhancing SoftMax loss or IsoMax loss with ODIN presents the same problems from a practical perspective. In CIFAR100, SoftMax loss with outlier exposure produces lower performance than IsoMax loss without it for both DenseNet and ResNet.

4.1.4.1 Fair Scenario

Table 5 shows that models trained with the SoftMax loss using the maximum probability as the score (SoftMax+MPS) almost always present the worst OOD detection performance. In the case of models trained with SoftMax loss, replacing the maximum probability score by the entropic score (SoftMax+ES) produces small OOD detection performance gains. However, the combination of models trained using IsoMax loss with the entropic score (IsoMax+ES), which is the proposed solution, significantly improves, usually by several percentage points, the OOD detection performance across almost all datasets, models, out-distributions, and metrics. We emphasize that the entropic score only produces high OOD detection performance when the probability distributions present high entropy (IsoMax+ES). For low entropy probability distributions (SoftMax+ES), the performance increase is minimal.

The IsoMax loss is well-positioned to replace SoftMax loss as a novel baseline OOD detection approach for neural networks OOD detection, as the former does not present an accuracy drop in relation to the latter and simultaneously improves the OOD detection performance. Additional techniques (e.g., input preprocessing, adversarial training, and outlier exposure) may be added to improve OOD detection performance gains further.

Table 6 shows that the results generalize to higher resolution images. Table 7 presents results for text data. As expected, the results show that our approach is *domain-agnostic* and therefore may be applied to data other than images.

Table 4 - IsoMax: Classification Accuracy and Enhanced Versions

Model	In-Data (training)	OOD Detection Approach	Enhanced Versions of SoftMax loss and IsoMax loss. Add-on Techniques produce different Side Effects and Requirements.		
			Class. Accuracy (%) [↑] SoftMax / IsoMax	TNR@TPR95 (%) [↑] SoftMax / IsoMax	AUROC (%) [↑] SoftMax / IsoMax
DenseNet	CIFAR10	Baseline	**95.4±0.3 / 95.2±0.3**	53.3±0.4 / **84.1±0.3**	91.9±0.4 / **97.3±0.3**
		+ Label Smoothing	**95.2±0.4 / 95.0±0.3**	**70.7±0.4** / 60.3±0.3	**94.9±0.3** / 80.8±0.3
		+ Center Loss Regularization	**95.3±0.3 / 95.1±0.4**	54.7±0.6 / **87.1±0.3**	92.8±0.3 / **97.6±0.3**
		+ ODIN	**95.4±0.3 / 95.2±0.3**	91.9±0.3 / **95.3±0.4**	**98.2±0.3 / 98.4±0.4**
		+ Outlier Exposure	**95.3±0.4 / 95.6±0.4**	93.8±0.3 / **94.7±0.3**	**98.5±0.3** / **98.8±0.4**
	CIFAR100	Baseline	**77.5±0.6 / 77.5±0.4**	22.3±0.7 / **45.1±0.6**	77.4±0.8 / **91.9±0.6**
		+ Label Smoothing	**77.0±0.4 / 77.2±0.6**	31.5±0.6 / **33.0±0.6**	82.0±0.6 / **88.3±0.7**
		+ Center Loss Regularization	**77.2±0.7 / 77.0±0.6**	30.3±0.7 / **43.7±0.6**	79.5±0.8 / **92.2±0.6**
		+ ODIN	**77.5±0.6 / 77.5±0.4**	64.4±0.7 / **83.1±0.8**	92.5±0.6 / **96.9±0.7**
		+ Outlier Exposure	**77.8±0.6 / 77.5±0.7**	23.0±0.7 / **36.4±0.8**	80.5±0.8 / **89.6±0.6**
	SVHN	Baseline	**96.6±0.2 / 96.6±0.2**	90.1±0.2 / **95.9±0.1**	98.2±0.2 / **98.9±0.2**
		+ Label Smoothing	**96.6±0.2 / 96.7±0.3**	87.5±0.2 / **93.5±0.2**	97.0±0.2 / **97.8±0.3**
		+ Center Loss Regularization	**96.7±0.3 / 96.6±0.2**	88.0±0.3 / **95.9±0.2**	97.9±0.2 / **98.9±0.1**
		+ ODIN	**96.6±0.2 / 96.6±0.2**	95.5±0.2 / **96.7±0.2**	98.8±0.1 / **99.1±0.1**
		+ Outlier Exposure	**96.6±0.3 / 96.7±0.3**	**99.9±0.1** / **99.9±0.1**	**99.9±0.1** / **99.9±0.1**
ResNet	CIFAR10	Baseline	**95.4±0.3 / 95.6±0.4**	50.8±0.4 / **78.6±0.3**	91.2±0.3 / **96.1±0.4**
		+ Label Smoothing	**95.5±0.4 / 95.4±0.3**	54.0±0.4 / **63.5±0.3**	78.9±0.4 / **84.5±0.3**
		+ Center Loss Regularization	**95.6±0.3 / 95.4±0.4**	53.5±0.4 / **80.6±0.3**	90.5±0.3 / **96.6±0.2**
		+ ODIN	**95.4±0.3 / 95.6±0.4**	73.7±0.3 / **85.6±0.3**	93.4±0.2 / **97.2±0.3**
		+ Outlier Exposure	**95.5±0.3 / 95.6±0.3**	91.1±0.3 / **94.2±0.4**	97.7±0.2 / **98.6±0.3**
	CIFAR100	Baseline	**77.3±0.6 / 77.4±0.7**	22.4±0.6 / **41.8±0.7**	80.5±0.6 / **90.1±0.7**
		+ Label Smoothing	**77.7±0.5 / 77.3±0.5**	21.0±0.7 / **33.4±0.6**	81.6±0.6 / **85.9±0.6**
		+ Center Loss Regularization	**77.9±0.5 / 77.4±0.5**	29.5±0.9 / **48.2±0.7**	80.3±0.6 / **90.6±0.6**
		+ ODIN	**77.3±0.6 / 77.4±0.7**	64.0±0.6 / **77.9±0.6**	92.7±0.7 / **95.8±0.6**
		+ Outlier Exposure	**77.3±0.5 / 77.0±0.5**	37.8±0.9 / **41.7±0.8**	86.6±0.6 / **88.6±0.7**
	SVHN	Baseline	**96.8±0.2 / 96.7±0.2**	71.7±0.3 / **92.4±0.2**	94.7±0.3 / **98.0±0.2**
		+ Label Smoothing	**96.9±0.2 / 96.9±0.3**	**86.3±0.2 / 86.4±0.2**	**97.2±0.3** / 94.9±0.2
		+ Center Loss Regularization	**96.9±0.2 / 96.7±0.2**	77.2±0.2 / **82.2±0.2**	94.5±0.2 / **96.1±0.3**
		+ ODIN	**96.8±0.2 / 96.7±0.2**	79.9±0.3 / **93.5±0.3**	95.1±0.2 / **98.2±0.3**
		+ Outlier Exposure	**96.9±0.3 / 96.8±0.2**	**99.9±0.1** / **99.9±0.1**	**99.9±0.1** / **99.9±0.1**

Source: The Author (2022). Comparison of Enhanced Versions of SoftMax Loss and IsoMax Loss: Adding label smoothing, center loss regularization, ODIN, and outlier exposure to SoftMax loss and IsoMax loss. The side effects and requirements added to the solution depend on the add-on technique. Regardless of using SoftMax loss or IsoMax loss, adding label smoothing (LS) requires validation of the hyperparameter ε (Szegedy *et al.*, 2016). The values searched for ε were 0.1 (Szegedy *et al.*, 2016; Lee & Cheon, 2020) and 0.01 (Lee & Cheon, 2020). Adding center loss regularization (CLR) to SoftMax loss or IsoMax loss requires validation of the hyperparameter λ (Wen *et al.*, 2016). The values searched for λ were 0.01 and 0.003 (Wen *et al.*, 2016). We emphasize that the center loss is not used as a stand-alone loss, but rather combined as a regularization term with a preexisting baseline loss. In the case of the original paper, the baseline loss was the SoftMax loss. In this paper, we added the mentioned regularization term (Wen *et al.*, 2016, Equation (5)) also to IsoMax loss to construct the center loss enhanced version of our loss. Regardless of using SoftMax loss or IsoMax loss, adding ODIN (Liang *et al.*, 2018) requires validation of the hyperparameters ε and T. The values searched for these hyperparameters were the same used in the original paper (Liang *et al.*, 2018). Adding ODIN implies using input preprocessing, which makes inferences much slower, and energy- and cost-inefficient. We used OOD data to validate the LS, CLR, and ODIN hyperparameters. Adding outlier exposure (OE) (Hendrycks *et al.*, 2019a) to SoftMax loss or IsoMax loss requires collecting outlier data. We used the same outlier data used in Hendrycks *et al.* (2019a). The add-on techniques were not applied to the SoftMax and IsoMax losses combined, but rather individually. The values of the performance metrics TNR@TPR95 and AUROC were averaged over all out-of-distribution data. All results used the entropic score, as it always overcame the maximum probability score. The results represent the mean and standard deviation of five executions. The best values are bold when they overcome the competing approach value outside the margin of error given by the standard deviations. For OOD detection, the variants of SoftMax and IsoMax losses that presented the best performance for each combination of architecture and dataset are blue.

Table 5 - IsoMax: OOD Detection Performance

Model	In-Data (training)	Out-Data (unseen)	Baseline Out-of-Distribution Detection Approaches Comparison. Fast and Energy-Efficient Inferences. No outlier data used.		
			TNR@TPR95 (%) [↑]	AUROC (%) [↑]	DTACC (%) [↑]
			SoftMax+MPS / SoftMax+ES / IsoMax+ES (ours)		
DenseNet	CIFAR10	SVHN	32.1±0.3 / 33.1±0.4 / **77.1±0.3**	86.5±0.4 / 86.8±0.3 / **96.7±0.4**	79.8±0.3 / 79.8±0.3 / **91.8±0.3**
		TinyImageNet	55.7±0.3 / 59.7±0.4 / **88.1±0.4**	93.5±0.4 / 94.1±0.4 / **97.9±0.3**	87.5±0.2 / 87.9±0.3 / **93.3±0.3**
		LSUN	64.8±0.4 / 69.6±0.3 / **94.6±0.2**	95.1±0.3 / 95.8±0.2 / **98.9±0.3**	89.8±0.3 / 90.1±0.3 / **95.0±0.4**
	CIFAR100	SVHN	20.5±0.6 / **24.8±0.8** / **23.6±0.9**	80.1±0.7 / 81.8±0.7 / **88.8±0.6**	73.8±0.6 / 74.4±0.7 / **83.9±0.6**
		TinyImageNet	19.3±0.9 / 23.8±0.8 / **49.0±0.6**	77.1±0.6 / 78.7±0.7 / **92.8±0.6**	70.5±0.6 / 71.3±0.7 / **86.5±0.6**
		LSUN	18.7±0.6 / 24.3±0.8 / **63.1±0.6**	75.8±0.7 / 77.8±0.6 / **94.8±0.6**	69.4±0.8 / 70.3±0.9 / **89.2±0.6**
	SVHN	CIFAR10	81.5±0.2 / 83.7±0.3 / **94.1±0.2**	96.5±0.3 / 96.9±0.2 / **98.5±0.2**	91.9±0.3 / 92.1±0.2 / **95.0±0.2**
		TinyImageNet	88.3±0.2 / 90.1±0.2 / **97.2±0.3**	97.7±0.3 / 98.1±0.2 / **99.1±0.2**	93.4±0.4 / 93.8±0.2 / **96.3±0.3**
		LSUN	86.4±0.2 / 88.4±0.4 / **96.8±0.2**	97.3±0.2 / 97.8±0.3 / **99.1±0.2**	92.8±0.2 / 93.0±0.4 / **96.0±0.2**
ResNet	CIFAR10	SVHN	43.2±0.4 / 44.5±0.3 / **83.6±0.4**	91.6±0.3 / 92.0±0.3 / **97.1±0.4**	86.5±0.2 / 86.4±0.4 / **91.9±0.3**
		TinyImageNet	46.4±0.4 / 48.0±0.3 / **70.3±0.3**	89.8±0.4 / 90.1±0.2 / **94.6±0.4**	84.0±0.4 / 84.2±0.3 / **88.3±0.4**
		LSUN	51.2±0.4 / 53.3±0.2 / **82.3±0.3**	92.2±0.4 / 92.6±0.4 / **96.9±0.3**	86.5±0.2 / 86.6±0.4 / **91.5±0.3**
	CIFAR100	SVHN	15.9±0.8 / 18.0±0.7 / **20.2±0.6**	71.3±0.6 / 72.7±0.7 / **85.3±0.6**	66.1±0.6 / 66.3±0.7 / **79.7±0.6**
		TinyImageNet	18.5±0.8 / 22.4±0.6 / **50.6±0.7**	74.7±0.6 / 76.3±0.7 / **92.0±0.7**	68.8±0.6 / 69.1±0.6 / **85.6±0.7**
		LSUN	18.3±0.8 / 22.4±0.5 / **54.9±0.6**	74.7±0.6 / 76.5±0.7 / **93.3±0.6**	69.1±0.5 / 69.4±0.7 / **87.6±0.8**
	SVHN	CIFAR10	67.3±0.2 / 67.7±0.3 / **92.3±0.2**	89.8±0.2 / 89.7±0.3 / **98.0±0.2**	87.0±0.3 / 86.9±0.3 / **94.1±0.2**
		TinyImageNet	66.8±0.3 / 67.3±0.2 / **94.6±0.2**	89.0±0.3 / 89.0±0.2 / **98.3±0.2**	86.8±0.2 / 86.6±0.4 / **94.8±0.4**
		LSUN	62.1±0.2 / 62.5±0.3 / **90.9±0.4**	86.0±0.2 / 85.8±0.2 / **97.8±0.2**	84.2±0.2 / 84.1±0.3 / **93.6±0.4**

Source: The Author (2022). OOD Detection: Fast and Energy-Efficient Inferences. No extra/outlier/background data used. The approaches do not require outlier/background/extra data. No additional procedures other than typical straightforward neural network training is required, and no classification accuracy drop is observed. All approaches present fast and energy-efficient inferences. Neither adversarial training, input preprocessing, temperature calibration, feature ensemble, nor metric learning is used. Since there is no need to tune hyperparameters, no access to OOD or adversarial examples is required. SoftMax+MPS means training with SoftMax loss and performing OOD detection using the maximum probability score, which is the approach defined in Hendrycks & Gimpel (2017). SoftMax+ES means training with SoftMax loss and performing OOD detection using the entropic score. IsoMax+ES means training with IsoMax loss and performing OOD detection using the entropic score. The results represent the mean and standard deviation of five executions. The best values are bold when they overcome the competing approach value outside the margin of error given by the standard deviations.

Table 6 - IsoMax: OOD Detection on TinyImageNet

Model	Accuracy (%) [↑] SoftMax / IsoMax	Out-Data (unseen)	Baseline Out-of-Distribution Detection Approaches Comparison. Fast and Energy-Efficient Inferences. No outlier data used.	
			TNR@TPR95 (%) [↑]	AUROC (%) [↑]
			SoftMax+MPS / IsoMax+ES (ours)	
DenseNetBC100 (small size)	61.1±0.2 / **61.6±0.3**	CIFAR10	24.3±3.3 / **38.9±3.7**	81.1±1.3 / **88.5±2.2**
		CIFAR100	23.8±2.4 / **36.3±3.1**	79.6±0.9 / **84.2±1.2**
		SVHN	40.3±3.5 / **64.4±2.7**	84.9±2.0 / **94.3±1.4**
ResNet34 (medium size)	**65.6±0.3 / 65.4±0.3**	CIFAR10	18.2±2.6 / **51.9±4.9**	78.3±1.2 / **90.9±1.0**
		CIFAR100	16.7±2.4 / **50.5±4.7**	76.3±1.4 / **88.2±1.2**
		SVHN	19.3±5.2 / **89.1±5.4**	73.8±1.7 / **98.0±0.8**
WideResNet2810 (big size)	67.0±0.2 / **67.5±0.2**	CIFAR10	30.9±5.8 / **56.0±4.3**	84.1±2.4 / **91.4±1.9**
		CIFAR100	31.5±1.8 / **54.1±3.8**	83.3±1.3 / **88.2±0.9**
		SVHN	59.7±2.3 / **89.0±4.4**	90.0±2.0 / **98.0±1.4**

The Author (2022). SoftMax+MPS means training with SoftMax loss and performing OOD detection using the maximum probability score (Hendrycks & Gimpel, 2017). IsoMax+ES means training with IsoMax loss and performing OOD detection using the entropic score. The results represent the mean and standard deviation of five executions. The best values are bold when they overcome the competing approach value outside the margin of error given by the standard deviations.

Table 7 - IsoMax: OOD Detection on Text Data

Model	In-Data (training)	Out-Data (unseen)	Baseline Out-of-Distribution Detection Approaches Comparison. Fast and Energy-Efficient Inferences. No outlier data used.	
			TNR@TPR95 (%) [↑]	AUROC (%) [↑]
			SoftMax+MPS / IsoMax+ES (ours)	
GRU2L	20 Newsgroups	IMDB	31.9±6.9 / **58.0±12.6**	86.8±2.0 / **92.2±2.0**
		Multi30K	29.7±3.1 / **47.9±13.3**	82.4±1.8 / **85.7±2.8**
		Yelp Reviews	26.7±3.8 / **46.6±5.7**	84.0±1.5 / **88.7±0.9**
	TREC	IMDB	35.8±8.4 / **54.3±5.3**	88.9±1.8 / **93.2±1.2**
		Multi30K	15.4±5.4 / **37.4±4.1**	74.8±3.8 / **82.0±2.3**
		Yelp Reviews	34.4±5.3 / **52.3±6.2**	82.5±0.4 / **90.2±1.2**

Source: The Author (2022). SoftMax+MPS means training with SoftMax loss and performing OOD detection using the maximum probability score (Hendrycks & Gimpel, 2017). IsoMax+ES means training with IsoMax loss and performing OOD detection using the entropic score. The results represent the mean and standard deviation of five executions. The best values are bold when they overcome the competing approach value outside the margin of error given by the standard deviations.

Table 8 - IsoMax: Comparison with No Seamless Solutions

Model	In-Data (training)	Out-Data (unseen)	ODIN, ACET, and Mahalanobis present troublesome requirements. ODIN, ACET, and Mahalanobis produce undesired side effects.	
			AUROC (%) [↑]	DTACC (%) [↑]
			ODIN / ACET / IsoMax+ES (ours) / Mahalanobis	
DenseNet	CIFAR10	SVHN	92.7±0.4 / NA / **96.7±0.4** / **97.5±0.6**	86.4±0.4 / NA / **91.8±0.4** / **92.4±0.5**
		TinyImageNet	97.3±0.4 / NA / **97.9±0.4** / **98.5±0.6**	92.1±0.4 / NA / **93.5±0.4** / **94.3±0.6**
		LSUN	98.4±0.4 / NA / **98.9±0.4** / **99.1±0.6**	94.3±0.3 / NA / **95.1±0.4** / **95.9±0.4**
	CIFAR100	SVHN	88.1±0.6 / NA / 88.7±0.7 / **91.7±0.6**	80.8±0.6 / NA / **83.9±0.6** / **84.3±0.7**
		TinyImageNet	85.2±0.6 / NA / 92.8±0.5 / **96.9±0.7**	77.1±0.6 / NA / 86.7±0.5 / **91.7±0.6**
		LSUN	85.8±0.6 / NA / 94.6±0.4 / **97.7±0.6**	77.3±0.6 / NA / 89.2±0.7 / **93.5±0.5**
	SVHN	CIFAR10	91.8±0.2 / NA / **98.6±0.2** / **98.7±0.3**	86.7±0.2 / NA / **95.7±0.3** / **96.1±0.3**
		TinyImageNet	94.8±0.2 / NA / **99.3±0.2** / **99.7±0.3**	90.2±0.2 / NA / 96.2±0.2 / **98.8±0.3**
		LSUN	94.0±0.2 / NA / **99.5±0.2** / **99.8±0.3**	89.1±0.2 / NA / 96.1±0.2 / **99.0±0.1**
ResNet	CIFAR10	SVHN	86.4±0.4 / **97.7±0.4** / **97.4±0.4** / 95.5±0.6	77.7±0.4 / NA / **91.8±0.4** / 89.0±0.5
		TinyImageNet	93.9±0.3 / 85.7±0.4 / 94.7±0.3 / **99.1±0.5**	86.1±0.3 / NA / 88.5±0.3 / **95.4±0.5**
		LSUN	93.4±0.4 / 85.9±0.4 / 96.8±0.3 / **99.5±0.3**	85.7±0.4 / NA / 91.3±0.3 / **97.3±0.4**
	CIFAR100	SVHN	72.1±0.6 / **91.1±0.6** / 85.2±0.6 / 84.3±0.5	67.8±0.4 / NA / **79.9±0.6** / 76.4±0.7
		TinyImageNet	83.7±0.6 / 75.3±0.5 / **92.4±0.5** / 87.7±0.6	75.7±0.5 / NA / **85.8±0.5** / 84.3±0.5
		LSUN	81.8±0.6 / 69.7±0.5 / **93.3±0.6** / 82.2±0.7	74.7±0.6 / NA / **87.6±0.5** / 79.6±0.6
	SVHN	CIFAR10	92.1±0.2 / 97.3±0.3 / **98.2±0.2** / 97.3±0.2	89.3±0.3 / NA / **94.3±0.2** / **94.5±0.3**
		TinyImageNet	92.8±0.2 / 97.6±0.2 / **98.8±0.1** / **99.0±0.3**	90.0±0.2 / NA / 94.6±0.3 / **98.7±0.2**
		LSUN	90.6±0.2 / **99.7±0.3** / 97.9±0.2 / **99.8±0.2**	88.3±0.2 / NA / 93.7±0.3 / **99.4±0.2**

Source: The Author (2022). OOD Detection: Unfair comparison of approaches with different requirements and side effects. ODIN (Liang *et al.*, 2018) uses input preprocessing, temperature calibration, and adversarial validation (hyperparameter tuning using adversarial examples). The Mahalanobis (Lee *et al.*, 2018) solution uses input preprocessing, adversarial validation, feature extraction, feature ensemble, and metric learning. Input preprocessing makes the inferences of ODIN and the Mahalanobis method at least three times slower and at least three times less energy/computationally efficient than SoftMax or IsoMax inferences. Feature ensembles may limit the Mahalanobis method scalability to deal with large-size images used in real-world applications. ACET (Hein *et al.*, 2019) uses adversarial training, which results in slower training, possibly reduced scalability for large-size images, and eventually, classification accuracy drop. The ODIN, the Mahalanobis approach, and ACET present hyperparameters that need to be validated for each combination of datasets and models presented in the table. Furthermore, considering that adversarial hyperparameters (e.g., the adversarial perturbation) used in ODIN/Mahalanobis/ACET were validated using the SVHN/CIFAR10/CIFAR100 validation sets and that these sets were reused as OOD detection test set, we conclude that the OOD detection performance reported by those papers, which we are reproducing in this table, may be overestimated. ODIN/Mahalanobis/ACET results were obtained using SoftMax loss rather than IsoMax loss as baseline. No outlier/extra/background data is used. IsoMax+ES means training with IsoMax loss and performing OOD detection using the entropic score. IsoMax+ES does not use these techniques and does not have such requirements, side effects, and hyperparameters to tune (as IsoMax+ES works as a baseline OOD detection approach, those techniques may be incorporated in future research). The results represent the mean and standard deviation of five executions. The best values are bold when they overcome the competing approach value outside the margin of error given by the standard deviations.

4.1.4.2 Unfair Scenario

Table 8 presents a perspective on how our proposed baseline OOD detection approach compares with no seamless solutions. Hence, we need to analyze the mentioned table considering that it shows an *unfair* comparison of approaches that present different requirements and side effects. ODIN and the Mahalanobis solution use input preprocessing[4]. Consequently, they present solutions with much slower and less energy-efficient inferences than models trained with IsoMax loss, which are as fast and computation-efficient as the models trained with common SoftMax loss. Moreover, they also require validation using adversarial samples.

Additionally, ODIN requires temperature calibration, while the Mahalanobis approach uses feature ensemble and metric learning, which may be implicated in limited scalability for large-size images. ACET requires adversarial training, which may also prevent its use in large-size images. ODIN, the Mahalanobis approach, and ACET have hyperparameters tuned for each combination of in-data and models in the table. ODIN, the Mahalanobis method, and ACET used SoftMax loss rather than IsoMax loss. IsoMax+ES exhibits neither the mentioned unfortunate requirements nor undesired side effects. Additionally, taking into consideration that ODIN, Mahalanobis, and ACET used adversarial hyperparameters (e.g., the adversarial perturbation) that were validated using the validation sets of SVHN/CIFAR10/CIFAR100 and noticing that these sets composed the OOD detection test sets, we conclude that some overestimation may be present in the OOD detection performance reported by these papers.

Regardless of the previous considerations, Table 8 shows that IsoMax+ES considerably outperforms ODIN in all evaluated scenarios. Therefore, in addition to avoiding hyperparameter tuning and access to OOD or adversarial samples, the results show that removing the entropic scale is much more effective in increasing the OOD detection performance than performing temperature calibration. Furthermore, IsoMax+ES usually outperforms ACET by a large margin. Moreover, in most cases, even operating under much more favorable conditions, the Mahalanobis method surpasses IsoMax+ES by less than 2%. In some scenarios, the latter overcomes the former despite avoiding hyperparameter validation, being seamless and producing much faster and more computation-efficient inferences, as no input preprocessing technique is required.

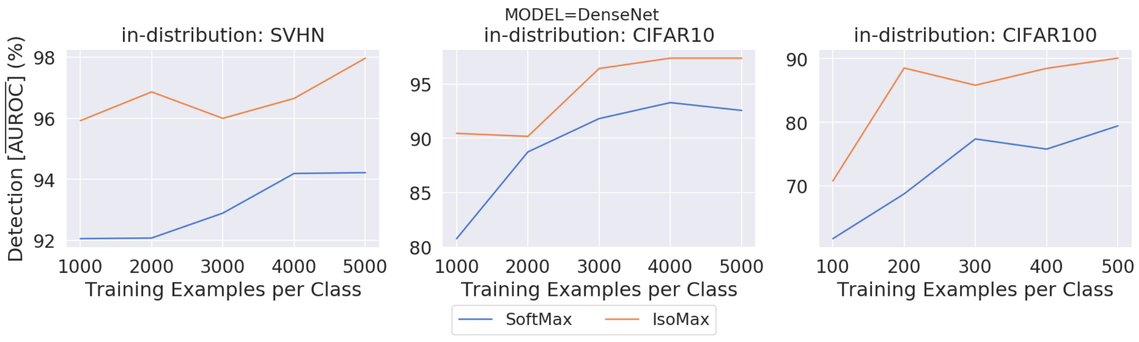
4.1.5 Training Examples per Class Variation Study

Fig. 36 presents the OOD detection performance of SoftMax and IsoMax losses in many models (DenseNet and ResNet), and datasets (SVHN, CIFAR10, and CIFAR100) for a range of training samples per class. For each in-distribution, the AUROC and TNR@TPR95 results were averaged over all possible combinations of OOD detection test set, except noise and fooling ones. The entropic score was used in all cases. It shows that IsoMax loss consistently and notably overcomes the OOD detection performance of SoftMax loss for virtually all combinations of evaluated datasets, training examples per class, metrics, and models.

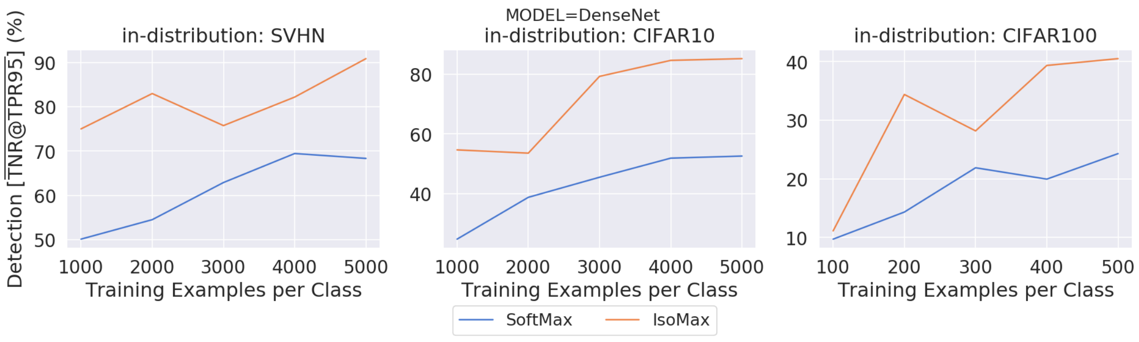
[4]To allow OOD detection, each inference requires a first neural network forward pass, a backpropagation, and a second forward pass.

Figure 36 - Training Examples per Class Variation Study: SoftMax vs. IsoMax

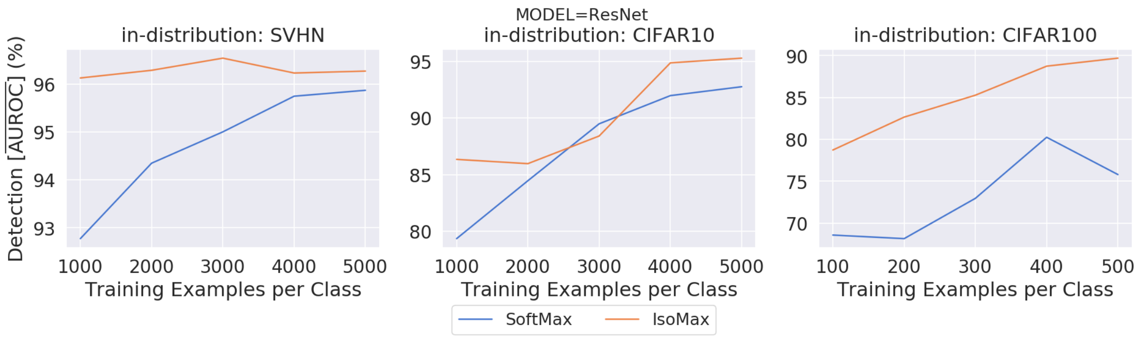
(a)



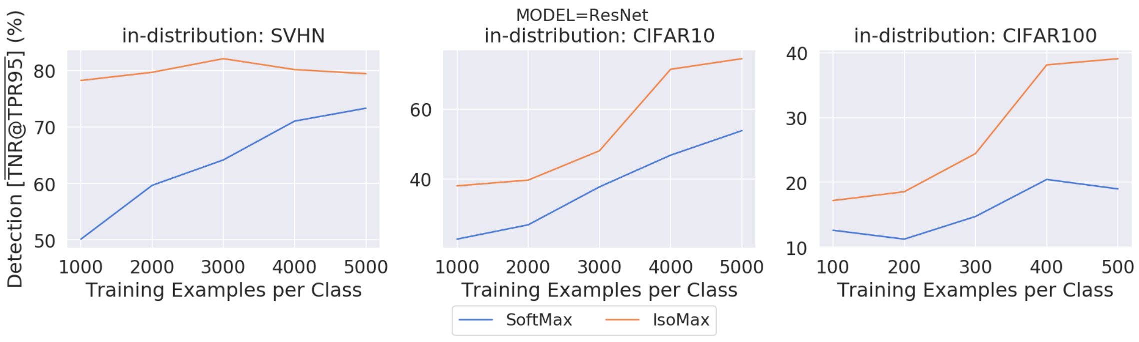
(b)



(c)



(d)



Source: The Author (2022). Training Examples per Class Variation Study: IsoMax loss consistently presents higher OOD detection performance than does SoftMax loss for different numbers of training examples per class on several datasets, models, and metrics. The entropic score was used for both SoftMax and IsoMax losses. For each in-distribution, we calculated the mean AUROC and TNR@TPR95 considering all possible OOD detection test sets. The vertical scales are automatically adjusted to better focus on the intervals the curves indeed variate.

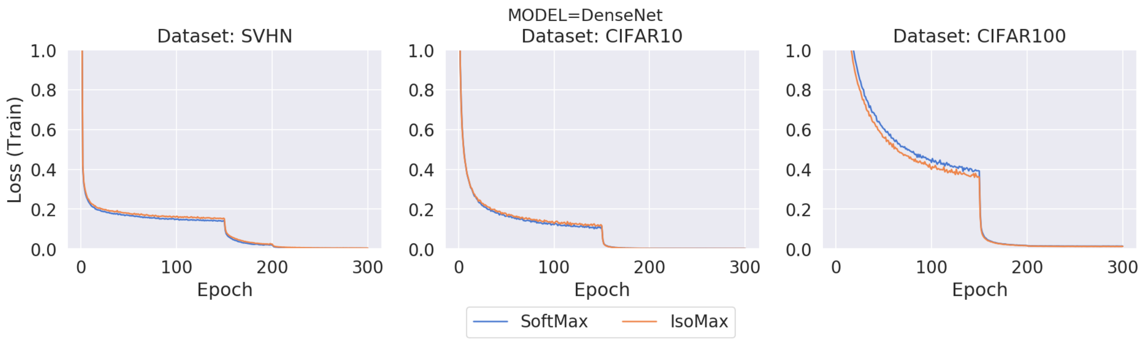
4.1.6 Logits, Probabilities, Entropies, and Training

Fig. 37 illustrates that training metrics are remarkably similar for SoftMax and IsoMax losses. Fig. 38 shows that the in-distribution *interclass* logits are more distinguishable from out-distribution logits when using IsoMax loss, which explains its increased OOD detection performance compared with the SoftMax loss because there are far more *interclass* logits than *intraclass* logits. Distances are calculated from class prototypes.

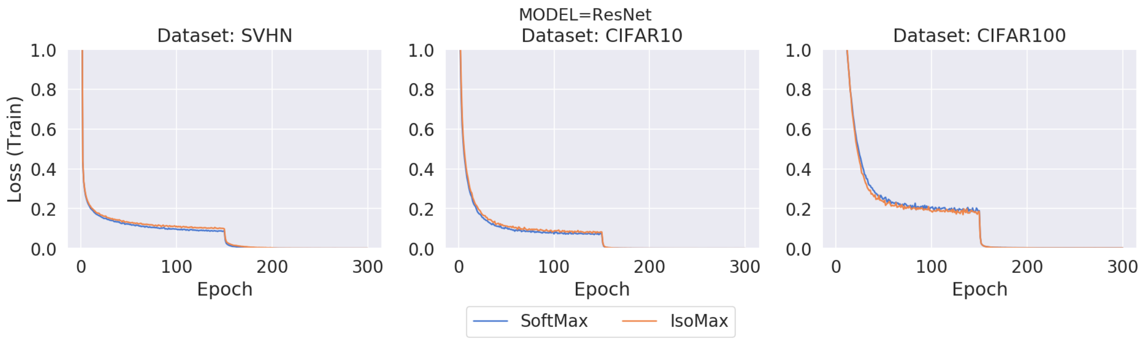
Fig. 39 shows that networks trained with SoftMax loss exhibit extremely high maximum probabilities. Sometimes this is true even for OOD samples. For networks trained with IsoMax loss, OOD samples usually present lower maximum probabilities compared with in-distribution samples. Furthermore, it also shows that the networks trained with SoftMax loss are extremely overconfident. It can be observed that the entropy works as a high-quality score to distinguish the in-distribution from the out-distribution in neural networks trained with IsoMax loss.

Figure 37 - Training Metrics: SoftMax vs. IsoMax

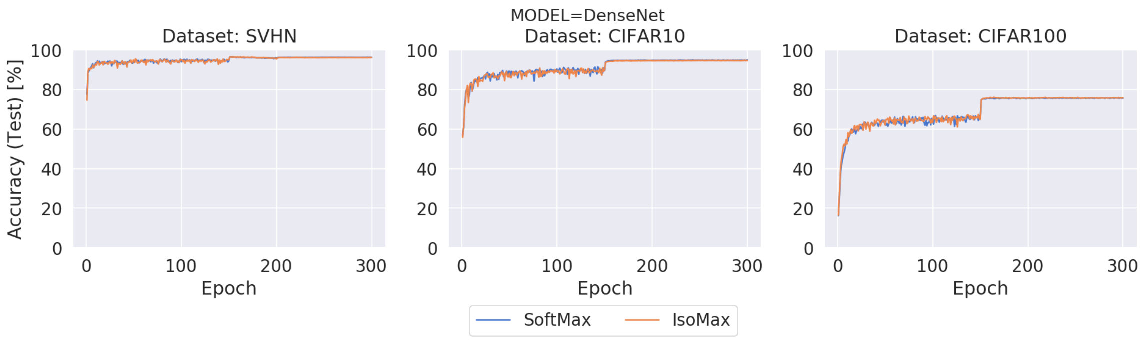
(a)



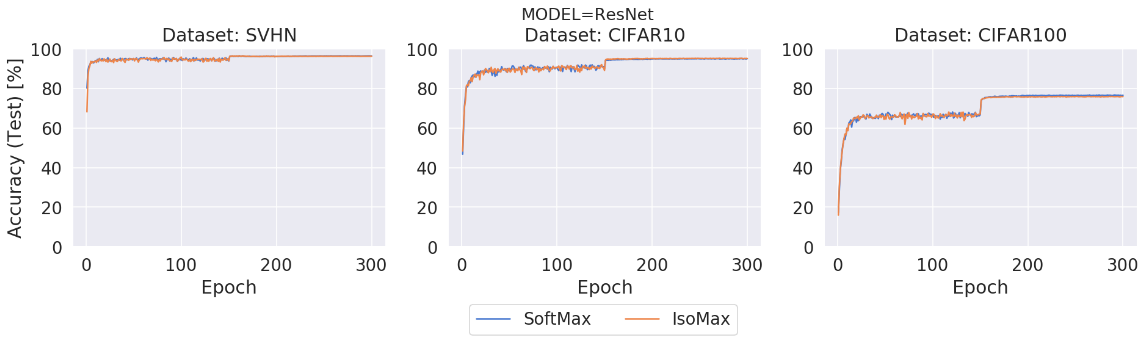
(b)



(c)



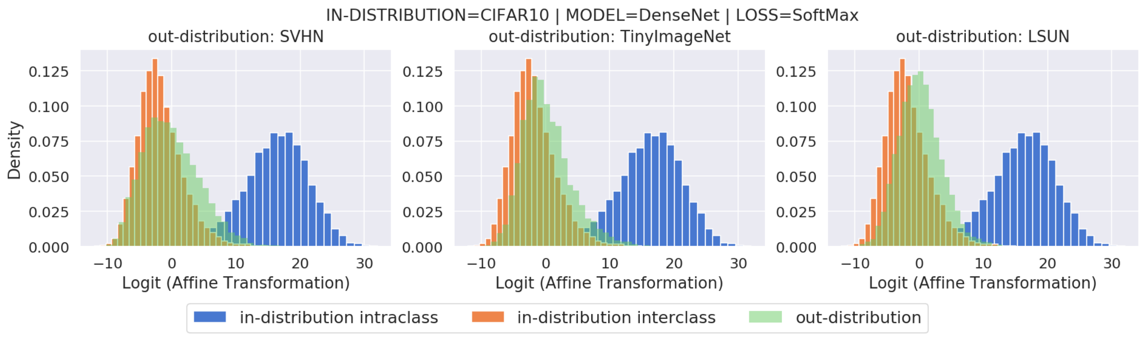
(d)



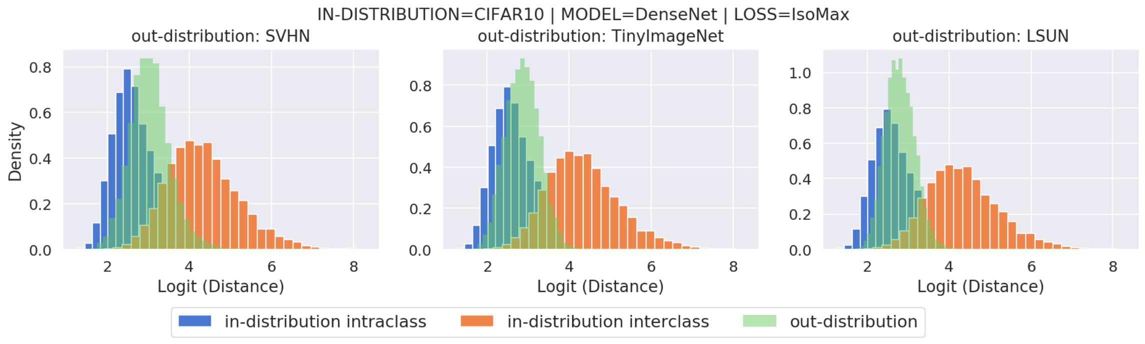
Source: The Authors (2022). Training metrics: (a) Training loss values and (b) test accuracies for a range of models, datasets, and losses. SoftMax loss and IsoMax loss present remarkably similar metrics throughout training, which confirms that IsoMax loss training is consistent and stable.

Figure 38 - Logits: SoftMax vs. IsoMax

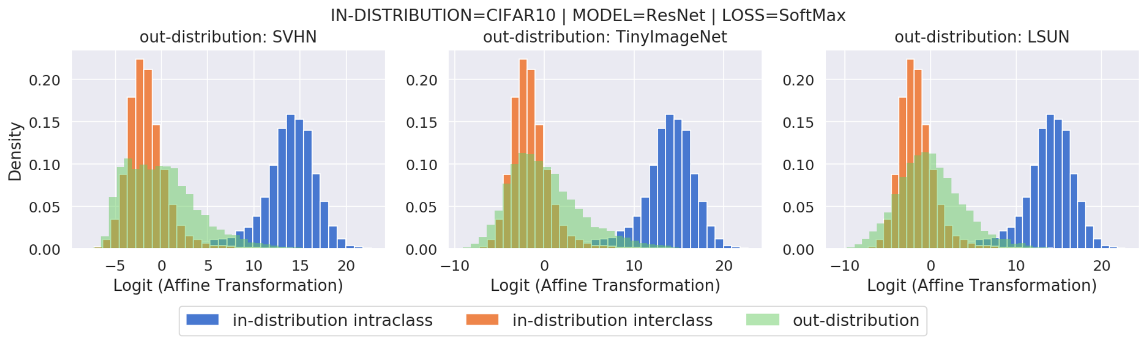
(a)



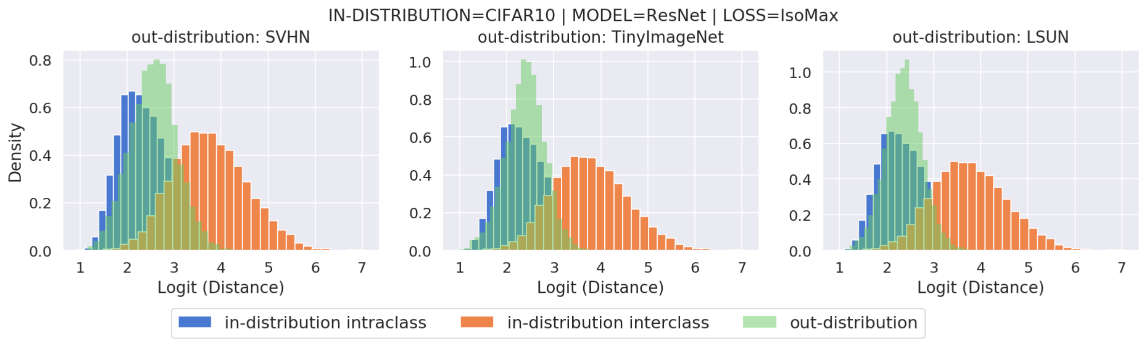
(b)



(c)



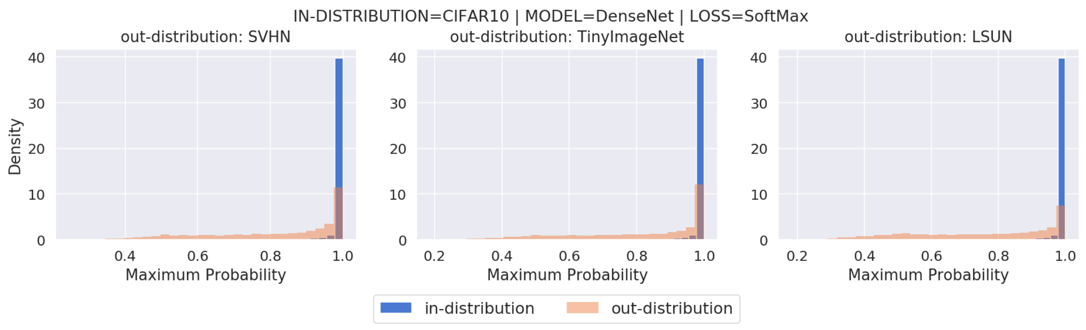
(d)



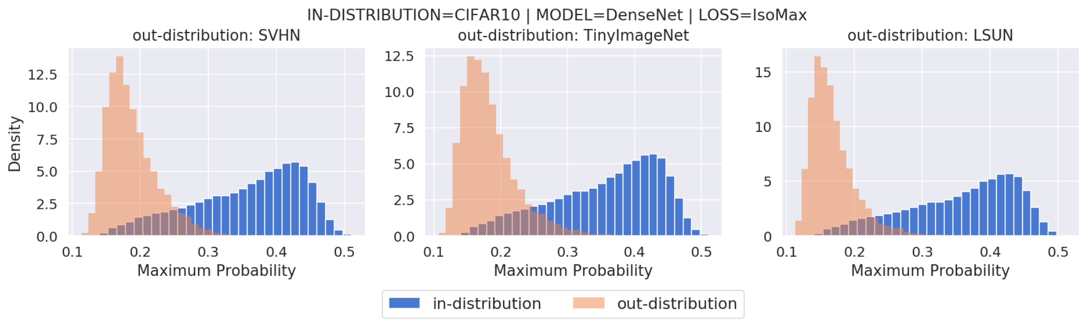
Source: The Authors (2022). Logits: (a,c) In SoftMax loss, out-distribution logits mimic in-distribution interclass logits. (b,d) In IsoMax loss, out-distribution logits mimic in-distribution intraclass logits, which facilitates OOD detection, as there are many more interclass logits than intraclass logits, and the entropic score takes into consideration the information provided by all network outputs rather than just one. Distances are calculated from class prototypes.

Figure 39 - Probabilities and Entropies: SoftMax vs. IsoMax

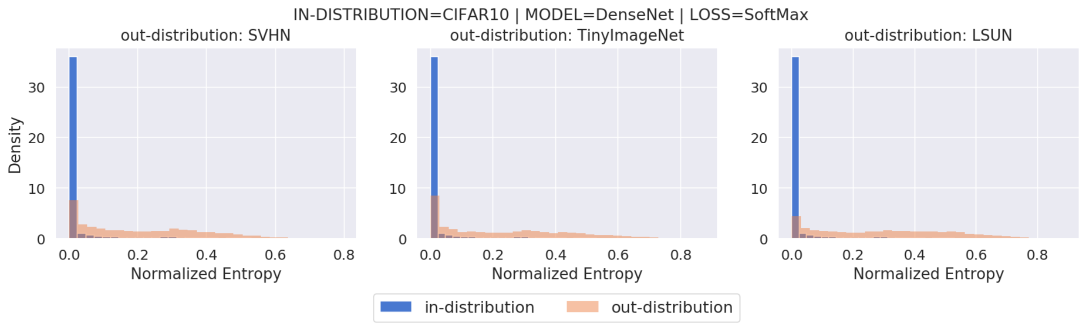
(a)



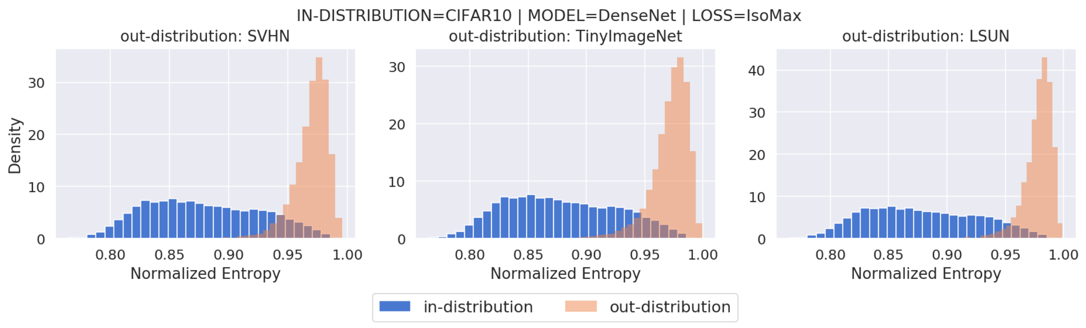
(b)



(c)



(d)



Source: The Author (2022). Probabilities and entropies: (a) SoftMax loss produces extremely high confident predictions for in-distribution samples. SoftMax loss usually provides extremely high maximum probabilities, even for OOD samples. (b) IsoMax loss produces less confident predictions for in-distribution samples than SoftMax loss. IsoMax loss commonly produces an even lower maximum probability for OOD samples. (c) SoftMax loss provides extremely low entropy (high confidence) for almost all in-distribution samples and even usually for OOD samples. (d) IsoMax loss produces high entropy for out-distributions. More precise separation between the in-distribution and out-distributions is obtained.

Table 9 - Inference Delays, Computational Cost and Energy Consumption

Model	In-Data (training)	Hardware (inference)	Out-of-Distribution Detection: Inference Delays / Presumed Computational Cost and Energy Consumption Rates.		
			SoftMax Loss (current baseline) MPS (ms) [↓] / ES (ms) [↓]	IsoMax Loss (ours) (proposed baseline) MPS (ms) [↓] / ES (ms) [↓]	Input Preprocessing: ODIN, Mahalanobis, Generalized ODIN (ms) [↓]
DenseNet	CIFAR10	CPU	18.1 / 19.4	18.0 / 19.2	242.4 (≈ **10x slower**)
		GPU	11.6 / 13.0	11.6 / 11.5	39.2 (≈ **4x slower**)
	CIFAR100	CPU	18.4 / 19.8	18.4 / 19.3	261.0 (≈ **10x slower**)
		GPU	12.9 / 11.4	11.8 / 11.5	39.6 (≈ **4x slower**)
	SVHN	CPU	18.1 / 18.6	18.3 / 18.6	241.5 (≈ **10x slower**)
		GPU	11.6 / 11.9	11.7 / 11.6	39.6 (≈ **4x slower**)
ResNet	CIFAR10	CPU	22.3 / 23.2	23.0 / 23.5	250.4 (≈ **10x slower**)
		GPU	4.5 / 3.8	4.2 / 4.1	15.4 (≈ **4x slower**)
	CIFAR100	CPU	23.3 / 23.1	23.3 / 23.8	252.6 (≈ **10x slower**)
		GPU	4.3 / 3.9	4.3 / 4.2	14.8 (≈ **4x slower**)
	SVHN	CPU	23.1 / 23.4	23.4 / 23.3	263.8 (≈ **10x slower**)
		GPU	4.2 / 4.0	4.0 / 4.0	15.7 (≈ **4x slower**)

Source: The Author (2022). MPS means maximum probability score (Hendrycks & Gimpel, 2017). ODIN is from Liang *et al.* (2018). Mahalanobis is from Lee *et al.* (2018). Generalized ODIN is from Hsu *et al.* (2020). ES means entropic score. For SoftMax and IsoMax losses (baseline OOD detection approaches), the inference delays combine both classification and detection computation. For the methods based on input preprocessing, the inference delays represent only the input preprocessing phase. All values are in milliseconds. In addition to presenting similar classification accuracy and much better OOD detection performance, IsoMax loss trained networks produce inferences as fast as SoftMax trained networks. Moreover, the entropic score is as fast as the maximum probability score. Using CPU (Intel i7-4790K, 4.00GHz, x64, octa-core) for inference (the case more relevant from a cost point of view), methods based on input preprocessing are about ten times slower than the baseline approaches. Using GPU (Nvidia GTX 1080 Ti) for inference, our approach is about four times faster than the methods based on input preprocessing. The inference delay rates presumably reflect similar computational cost and energy consumption rates. The inference delays presented are the mean value of the inference delay of each image calculated (batch size equals one) over the entire dataset. The standard deviation was below 0.3 for all cases.

4.1.7 Inference Efficiency

Table 9 presents the inference delays for SoftMax loss, IsoMax loss, and competing methods using CPU and GPU. We observe that neural networks trained using IsoMax loss produce inferences equally fast as the ones produced by networks trained using SoftMax loss, regardless of using CPU or GPU for inference. Additionally, the entropic score is as fast as the usual maximum probability score. Moreover, methods based on input preprocessing were more than ten times slower on CPU and about four times slower on GPU. Finally, those rates presumably apply to the computational cost and energy consumption as well.

At first sight, inference methods (i.e., methods that can be applied to pre-trained models) may be seen as “low cost” compared with training methods like our IsoMax loss, as we avoid training or fine-tuning the neural network. However, this conclusion may be misleading, as we have to keep in mind that inference methods (e.g., ODIN (Liang *et al.*, 2018) and Mahalanobis (Lee *et al.*, 2018)) produce inferences that are much more energy, computation, and time inefficient (Table 9).

For example, consider initially the rare practical situation where a pretrained model is available, and no fine-tuning to a custom dataset is required. In such cases, an inference method may indeed be applied without requiring any loss function. However, despite avoiding training or fine-tuning a neural network once or a few times, all the subsequent inferences, which are usually performed thousands or millions of times on the field (sometimes even by constrained devices), will be about 6 to 10 times more computational, energy, environment, and time inefficient (Table 9).

Alternatively, consider the case where a pretrained model is not available or fine-tuning to a custom dataset is needed. In this situation, which is much more likely in practice, we cannot avoid training or fine-tuning the neural network, and a training method like ours will be required anyway. In these cases, it would be recommended to train or fine-tune using IsoMax rather than SoftMax loss, as our experiments showed that both training times are the same and IsoMax loss produces considerably higher OOD detection performance.

Hence, inference methods are more inference inefficient because they coexist with a model that was trained with a loss not designed from the start with OOD detection in mind. The drawback is to produce an enormous amount of inefficient inferences on the field that usually uses constrained computational resource devices such as embedded systems.

Nevertheless, suppose the increased inference time, computation, environment damage, and energy consumption required to use an inference method is not a concern from a practical point of view. In that case, the model pretrained or fine-tuned using a training method may be subsequently subjected to the desired inference approach to increase overall OOD detection performance further. In other words, we do not claim that inference methods such as ODIN and Mahalanobis do not increase the OOD detection performance compared with Maximum Score Probability or even the Entropic Score. However, we point out that producing more energy inefficient inferences is one drawback of adopting such inference methods.

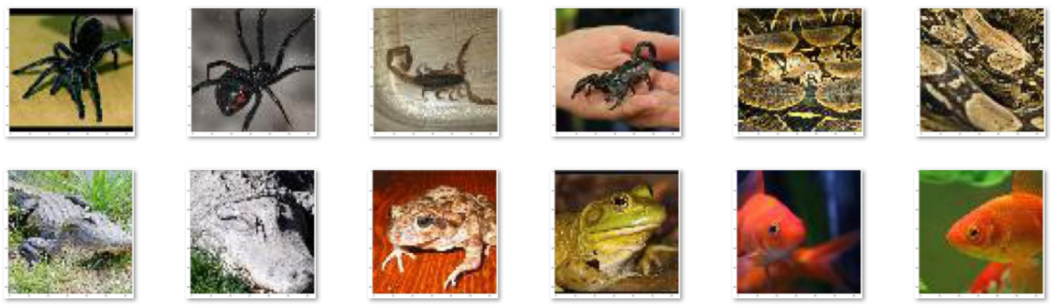
In summary, rather than concurrent, the training and inference methods are orthogonal and complementary. Moreover, we see no reason not to train the models with a loss designed to support OOD, regardless of subsequently applying an OOD inference method.

4.1.8 Qualitative Study

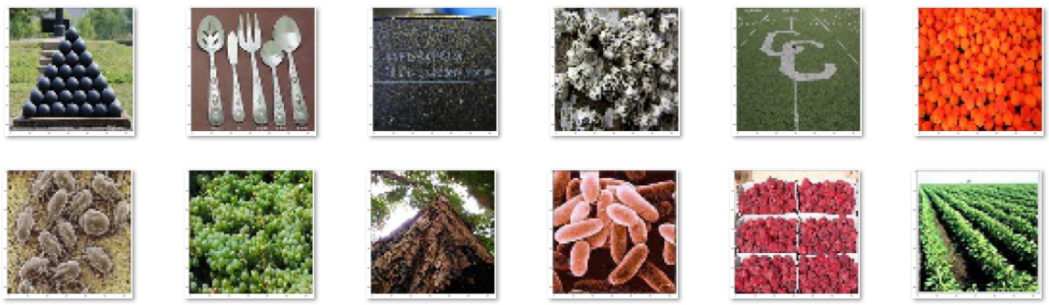
Fig. 40 presents examples of out-of-distribution detection performed by a ResNet34 trained on TinyImageNet using IsoMax. Resized examples from ImageNet-O (Hendrycks *et al.*, 2019b) were used as OOD examples. Examples in Fig 40a are from ID and were correctly detected as such. Moreover, examples from Fig. 40b are from OOD and were correctly detected as such. Examples in Fig. 40c were wrongly detected as OOD. It may be possibly explained by the fact they visually appear composed of more than one class or no class at all. are not very well-defined as belonging to a particular class. Fig. 40d present examples that were wrongly detected as ID. It may have happened because they present images with shapes and textures very different from those used in the training data.

Figure 40 - Out-of-Distribution Detection Qualitative Study

(a)

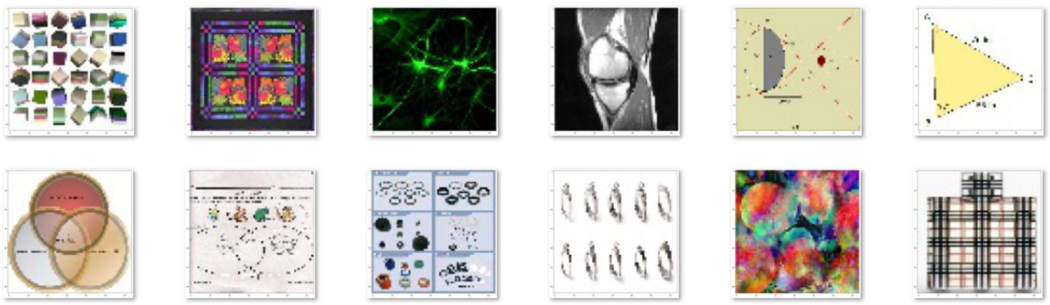


(b)



(c)

(d)



Source: The Authors (2022). Out-of-Distribution Detection Qualitative Study: a) In-distribution examples correctly detected as in-distribution. b) Out-of-distribution examples correctly detected as out-of-distribution. c) In-distribution examples wrongly detected as out-of-distribution. d) Out-of-distribution examples wrongly detected as in-distribution.

4.2 ENHANCED ISOTROPY MAXIMIZATION LOSS

To allow standardized comparison, we used the datasets, training procedures, and metrics that were established in Hendrycks & Gimpel (2017) and used in many subsequent OOD detection papers (Liang *et al.*, 2018; Lee *et al.*, 2018; Hein *et al.*, 2019). We trained many 100-layer DenseNetBCs with growth rate $k = 12$ (i.e., 0.8M parameters) Huang *et al.* (2017), 110-layer ResNets (*correct size proper implementation*) (He *et al.*, 2016)[5], and 34-layer ResNets (*overparametrized commonly used implementation*) (He *et al.*, 2016)[6] on CIFAR10 (Krizhevsky, 2009), CIFAR100 (Krizhevsky, 2009), SVHN (Netzer & Wang, 2011), and TinyImageNet (Deng *et al.*, 2009) datasets with SoftMax, IsoMax, and IsoMax+ losses using the same procedures (e.g., initial learning rate, learning rate schedule, weight decay).

We used SGD with the Nesterov moment equal to 0.9 with a batch size of 64 and an initial learning rate of 0.1. The weight decay was 0.0001, and we did not use dropout. We trained during 300 epochs for CIFAR10, CIFAR100, and SVHN. We used a learning rate decay rate equal to ten applied in epoch numbers 150, 200, and 250 for CIFAR10, CIFAR100, and SVHN.

We used resized images from the TinyImageNet (Deng *et al.*, 2009), the Large-scale Scene UNderstanding dataset (LSUN) (Yu *et al.*, 2015), CIFAR10, and SVHN (Netzer & Wang, 2011) to create out-of-distribution samples. We added these out-of-distribution images to the validation sets of in-distribution data to form the test sets and evaluate the OOD detection performance. We evaluated the OOD detection performance using the true negative rate at 95% true positive rate (TNR@TPR95), the area under the receiver operating characteristic curve (AUROC) and the detection accuracy (DTACC), which corresponds to the maximum classification probability over all possible thresholds δ:

$$1 - \min_{\delta} \Big\{ P_{\texttt{in}} \big(o(\mathbf{x}) \leq \delta \big) P(\mathbf{x} \text{ is from } P_{\texttt{in}}) \\ + P_{\texttt{out}} \big(o(\mathbf{x}) > \delta \big) P(\mathbf{x} \text{ is from } P_{\texttt{out}}) \Big\},$$

where $o(\mathbf{x})$ is the OOD detection score. It is assumed that both positive and negative samples have an equal probability of being in the test set, i.e., $P(\mathbf{x} \text{ is from } P_{\texttt{in}}) = P(\mathbf{x} \text{ is from } P_{\texttt{out}})$. This is the way this metric is defined and used in the literature. All metrics follow the calculation procedures specified in Lee *et al.* (2018).

In this section, we present the results and discussion. We initially show that the enhanced IsoMax produces classification accuracies that are comparable to those of the SoftMax loss function. We then show that the enhanced IsoMax produces much higher OOD detection performance than the IsoMax Loss and the SoftMax Loss.

[5] https://github.com/akamaster/pytorch_resnet_cifar10

[6] https://github.com/pokaxpoka/deep_Mahalanobis_detector

Table 10 - Classification accuracies using SoftMax, IsoMax, and IsoMax+ losses

Model	Data	Train Accuracy (%) [↑]	Test Accuracy (%) [↑]
		SoftMax Loss / IsoMax Loss / IsoMax+ Loss	
DenseNet100	CIFAR10	99.9±0.1 / 99.9±0.1 / 99.9±0.1	95.3±0.2 / 95.2±0.3 / 95.3±0.1
	CIFAR100	99.9±0.1 / 99.0±0.1 / 99.9±0.1	77.2±0.3 / 77.3±0.4 / 77.0±0.3
	SVHN	97.0±0.2 / 97.8±0.2 / 97.0±0.3	96.5±0.2 / 96.6±0.3 / 96.5±0.2
ResNet110	CIFAR10	99.9±0.1 / 99.9±0.1 / 99.9±0.1	94.4±0.3 / 94.5±0.3 / 94.6±0.2
	CIFAR100	99.5±0.1 / 99.9±0.1 / 99.8±0.1	72.7±0.2 / 74.1±0.4 / 73.9±0.3
	SVHN	99.8±0.1 / 99.9±0.1 / 99.5±0.1	96.6±0.3 / 96.8±0.2 / 96.9±0.3

Source: The Authors (2022). Besides preserving classification accuracy compared with SoftMax loss- and IsoMax loss-trained networks, IsoMax+ loss-trained models show higher OOD detection performance. Results are shown as means and standard deviations of five different iterations (see Table 11).

4.2.1 Classification Accuracy

Table 10 shows classification accuracies. We see that IsoMax+ loss never exhibits *classification accuracy drop* compared to SoftMax loss or IsoMax loss regardless of the dataset and model. The IsoMax loss variants achieve more than one percent better accuracy than the SoftMax loss when using ResNet110 on the CIFAR100 dataset.

4.2.2 Out-of-Distribution Detection

Table 11 summarizes the results of the *fair* OOD detection comparison. We report the results using the entropic score for SoftMax loss (SoftMax$_{\text{ES}}$) and IsoMax loss (IsoMax$_{\text{ES}}$) because this score always overcame the maximum probability score in these cases. For IsoMax+, we report the values using the minimum distance score (IsoMax+$_{\text{MDS}}$) because this method overcame the maximum probability and the entropic score in this situation.

All approaches are accurate (i.e., exhibit no *classification accuracy drop*); fast and power-efficient (i.e., inferences are performed without *input preprocessing*); and no hyperparameter tuning was performed. Additionally, no additional/outlier/background data are required. IsoMax+$_{\text{MDS}}$ always overcomes IsoMax$_{\text{ES}}$ performance, regardless of the model, dataset, and out-of-distribution data.

The minimum distance score produces high OOD detection performance when combined with IsoMax+, which shows that the isometrization of the distances works appropriately in this case. However, the same minimum distance score produced low OOD detection performance when combined with the original IsoMax loss. Fig. 41, and Table 14 provide an explanation for this important fact.

Table 12 summarizes the results of an *unfair* OOD detection comparison because the methods have different requirements and produce distinct side effects. ODIN (Liang *et al.*, 2018) and the Mahalanobis[7] (Lee *et al.*, 2018) approaches require adversarial samples to be generated to validate hyperparameters for each combination of dataset and model. These approaches also use *input preprocessing*, *which makes inferences at least four times slower and less energy-efficient*

[7]Considering that the proposed approach easily outperforms the vanilla Mahalanobis method when applied to SoftMax loss trained models (i.e., training a Mahalanobis distance-based classifier using features extracted from the neural network and using the Mahalanobis distance as score), we use the term Mahalanobis approach to refer to the full Mahalanobis approach.

(Macêdo *et al.*, 2021, 2022). Validation using adversarial examples may be cumbersome when performed from scratch on novel datasets because hyperparameters such as optimal adversarial perturbations may be unknown in such cases. IsoMax+$_{\text{MDS}}$ does not have these requirements, and does not produce the side effects.

Additionally, IsoMax+$_{\text{MDS}}$ achieves higher performance than ODIN by a large margin. In addition to the differences between the entropy maximization trick and temperature calibrations present in Macêdo *et al.* (2021, 2022), we emphasize that training with an entropic scale affects the learning of all weights, while changing the temperature during inference affects only the last layer. Thus, the fact that the proposed solution overcomes ODIN by a safe margin is evidence that the *entropy maximization trick produces much higher OOD detection performance than temperature calibration*, even when the latter is combined with input preprocessing. Moreover, the entropy maximization trick does not require access to validation data to tune the temperature.

In addition to being seamless and avoiding the drawbacks of the Mahalanobis approach, IsoMax+$_{\text{MDS}}$ typically overcomes it in terms of AUROC and produces similar performance when considering the DTACC.

Table 13 *unfairly* compares the performance of the proposed approach with the outlier exposure solution. Similar to IsoMax variants, the outlier exposure approach does not require hyperparameter tuning and produces efficient inferences. However, it does require collecting outlier data, while our approach does not. We emphasize that outlier exposure may also be combined with IsoMax loss variants to further increase the OOD detection performance (Macêdo *et al.*, 2022). In the table, we present the IsoMax loss variants *without outlier exposure* to show that the outlier exposure-enhanced SoftMax loss typically achieves worse OOD detection than IsoMax+$_{\text{MDS}}$ *even without using outlier exposure*.

The minimum distance score produces high OOD detection performance when combined with IsoMax+, which shows that the isometrization of the distances works appropriately in this case. However, the same minimum distance score produced low OOD detection performance when combined with the original IsoMax loss. Fig. 41, and Table 14 provide explanations.

Table 11 - IsoMax+: OOD Detection Performance

Model	Data (training)	OOD (unseen)	Out-of-Distribution Detection: Seamless Approaches. TNR@TPR95 (%) [↑]	AUROC (%) [↑]
			SoftMax$_{\text{ES}}$ / IsoMax$_{\text{ES}}$ / IsoMax+$_{\text{MDS}}$ (ours)	SoftMax$_{\text{ES}}$ / IsoMax$_{\text{ES}}$ / IsoMax+$_{\text{MDS}}$ (ours)
DenseNet100	CIFAR10	SVHN	40.4±5.3 / 78.6±9.0 / **97.0±0.7**	89.1±2.2 / 96.2±1.0 / **99.4±0.1**
		TinyImageNet	58.0±3.7 / 83.9±3.6 / **92.6±2.4**	94.0±0.6 / 97.1±0.4 / **98.5±0.3**
		LSUN	64.6±1.7 / 90.3±1.4 / **94.3±1.4**	95.2±0.4 / 98.0±0.3 / **99.1±0.2**
	CIFAR100	SVHN	21.9±2.8 / 29.6±3.7 / **78.2±4.1**	78.4±3.3 / 88.8±2.8 / **96.3±1.3**
		TinyImageNet	24.0±3.0 / 49.3±4.9 / **85.5±2.8**	75.5±2.1 / 90.8±2.1 / **97.5±0.8**
		LSUN	24.9±2.9 / 60.6±6.6 / **78.3±3.9**	77.2±3.2 / 93.0±1.4 / **97.2±1.3**
	SVHN	CIFAR10	85.2±3.3 / 93.4±1.1 / **95.1±0.4**	97.3±0.3 / 98.4±0.2 / **99.1±0.1**
		TinyImageNet	90.8±0.6 / 96.2±0.9 / **98.1±0.3**	98.3±0.2 / 99.0±0.2 / **99.6±0.1**
		LSUN	87.9±0.8 / 94.3±1.7 / **97.7±1.1**	97.8±0.3 / 98.7±0.2 / **99.6±0.2**
ResNet110	CIFAR10	SVHN	35.6±5.0 / 68.7±5.5 / **85.3±3.9**	88.9±0.9 / 94.8±1.5 / **98.1±0.9**
		TinyImageNet	39.0±4.1 / 65.4±5.6 / **76.2±2.7**	88.2±2.1 / 94.3±0.4 / **96.1±0.5**
		LSUN	46.9±5.7 / 80.7±2.2 / **86.9±2.5**	91.2±1.7 / 96.5±0.3 / **97.9±0.7**
	CIFAR100	SVHN	16.6±1.5 / 20.8±5.9 / **41.0±6.4**	73.6±3.7 / 85.3±1.3 / **88.9±1.2**
		TinyImageNet	15.6±2.1 / 22.8±1.9 / **44.7±5.9**	71.5±1.7 / 81.3±2.3 / **88.0±2.9**
		LSUN	16.5±3.3 / 22.9±3.5 / **46.1±4.3**	72.6±4.2 / 83.3±2.0 / **89.1±2.4**
	SVHN	CIFAR10	80.9±3.5 / 84.3±1.3 / **88.4±2.3**	95.1±0.3 / 96.5±0.2 / **97.4±0.3**
		TinyImageNet	84.0±3.2 / 87.4±2.2 / **93.4±1.8**	96.6±0.7 / 96.7±0.6 / **98.5±0.7**
		LSUN	81.4±2.3 / 84.7±2.1 / **90.2±2.2**	96.1±0.3 / 95.8±0.4 / **97.8±0.3**

Source: The Authors (2022). Fair comparison of seamless out-of-distribution detection approaches: No hyperparameter tuning, no additional/outlier/background data, no classification accuracy drop, and no slow inferences. SoftMax$_{\text{ES}}$ means training using SoftMax loss and performing OOD detection using the entropic score (ES). IsoMax$_{\text{ES}}$ means training using IsoMax loss and performing OOD detection using the entropic score (ES). IsoMax+$_{\text{MDS}}$ means training using IsoMax+ loss and performing OOD detection using minimum distance score (MDS). Results are shown as the mean and standard deviation of five iterations. The best results are shown in bold. See Table 10.

Table 12 - IsoMax+: Unfair Comparison with No Seamless Approaches

Model	Data (training)	OOD (unseen)	Comparison with approaches that use input preprocessing and adversarial validation. AUROC (%) [↑]	DTACC (%) [↑]
			ODIN / Mahalanobis / IsoMax+$_{\text{MDS}}$ (ours)	ODIN / Mahalanobis / IsoMax+$_{\text{MDS}}$ (ours)
DenseNet100	CIFAR10	SVHN	92.1±0.2 / 97.2±0.3 / **99.4±0.1**	86.1±0.3 / 91.9±0.3 / **96.3±0.4**
		TinyImageNet	97.2±0.3 / 97.7±0.2 / **98.5±0.3**	91.9±0.3 / **94.3±0.5** / **93.9±0.6**
		LSUN	98.5±0.3 / 98.6±0.2 / **99.1±0.2**	94.3±0.3 / **95.7±0.4** / **95.3±0.5**
	CIFAR100	SVHN	88.0±0.5 / 91.3±0.4 / **96.3±1.3**	80.0±0.6 / 84.3±0.4 / **90.3±0.5**
		TinyImageNet	85.6±0.5 / **96.7±0.3** / **97.5±0.8**	77.6±0.5 / **91.0±0.4** / **91.3±0.3**
		LSUN	85.7±0.6 / **97.1±1.9** / **97.2±1.3**	77.5±0.4 / **92.5±0.8** / **91.7±0.7**
ResNet34	CIFAR10	SVHN	86.0±0.3 / 95.0±0.3 / **98.0±0.4**	77.1±0.4 / 88.7±0.3 / **93.5±0.4**
		TinyImageNet	92.6±0.3 / **98.3±0.4** / 95.3±0.3	86.5±0.5 / **94.8±0.3** / 90.0±0.4
		LSUN	93.0±0.4 / **98.8±0.3** / 96.3±0.4	86.3±0.4 / **96.8±0.4** / 92.1±0.5
	CIFAR100	SVHN	71.0±0.4 / 84.0±0.6 / **88.0±0.7**	68.0±0.5 / 77.3±0.7 / **82.1±0.4**
		TinyImageNet	83.1±0.3 / 87.3±0.5 / **90.5±0.4**	76.2±0.4 / **84.0±0.4** / **84.2±0.5**
		LSUN	81.0±0.3 / 82.0±0.5 / **88.6±0.6**	75.2±0.3 / 79.3±0.5 / **82.5±0.4**

Source: The Authors (2022). Unfair comparison with approaches that use input preprocessing and produce slow/inefficient inferences in addition to requiring validation using adversarial examples. ODIN and Mahalanobis were applied to models trained using SoftMax loss. These approaches compute at least four times slower and less power efficient inferences Macêdo *et al.* (2021) because they use input preprocessing. Their hyperparameters were validated using adversarial examples. Additionally, Mahalanobis requires feature extraction for training ad-hoc models to perform OOD detection. Finally, feature ensemble was also used. IsoMax+$_{\text{MDS}}$ (ours) means training using IsoMax+ loss and performing OOD detection using minimum distance score (MDS). No hyperparameter tuning is required when using IsoMax+ loss for training and the MDS for OOD detection. The IsoMax+ loss OOD detection performance shown in this table may be increased further without relying on input preprocessing or hyperparameter tuning by replacing the minimum distance score with the Mahalanobis Lee *et al.* (2018) or the energy score Liu *et al.* (2020b). Results are the mean and standard deviation of five runs. The best results are shown in bold.

Table 13 - IsoMax+: Unfair Comparison with Outlier Exposure

Model	Data (training)	Comparison of IsoMax loss variants without using additional data with outlier exposure-enhanced SoftMax loss.	
		TNR@TPR95 (%) [↑]	AUROC (%) [↑]
		$\text{SoftMax}_{\text{ES}}^{\text{OE}}$ / $\text{IsoMax}_{\text{ES}}$ / $\text{IsoMax+}_{\text{MDS}}$ (ours)	
DenseNet100	CIFAR10	**94.4±1.4** / 84.2±4.6 / **94.6±1.5**	98.0±0.3 / 97.1±0.5 / **99.0±0.2**
	CIFAR100	36.4±9.4 / 46.5±5.0 / **80.6±3.6**	83.8±5.6 / 90.8±2.1 / **97.0±1.1**
ResNet110	CIFAR10	**82.4±2.1** / 71.6±4.4 / **82.8±3.0**	**96.8±0.7** / 95.2±0.7 / **97.3±0.7**
	CIFAR100	29.1±4.5 / 22.1±3.7 / **43.9±5.5**	80.5±2.8 / 83.3±1.8 / **88.6±2.1**

Source: The Author (2022). Unfair comparison of outlier exposure-enhanced SoftMax loss with IsoMax loss and IsoMax+ loss without using additional data. $\text{SoftMax}_{\text{ES}}^{\text{OE}}$ means training using SoftMax loss enhanced during training using outlier exposure (Hendrycks *et al.*, 2019a), which requires the collection of outlier data, and performing OOD detection using the entropic score. We used the same outlier data used in Hendrycks *et al.* (2019a). We collected the same amount of outlier data as the number of training examples present in the training set used to train $\text{SoftMax}^{\text{OE}}$. Despite being possible (Macêdo *et al.*, 2022), the IsoMax loss and IsoMax+ loss were not enhanced by outlier exposure to keep the solution seamless. $\text{IsoMax}_{\text{ES}}$ means training using IsoMax loss and performing OOD detection using the entropic score. $\text{IsoMax+}_{\text{MDS}}$ (ours) means training using IsoMax+ loss and performing OOD detection using minimum distance score (MDS). The values of the performance metrics TNR@TPR95 and AUROC were averaged over all out-of-distribution (SVHN, TinyImageNet, and LSUN). Results are shown as the mean and standard deviation of five iterations. The best results are shown in bold.

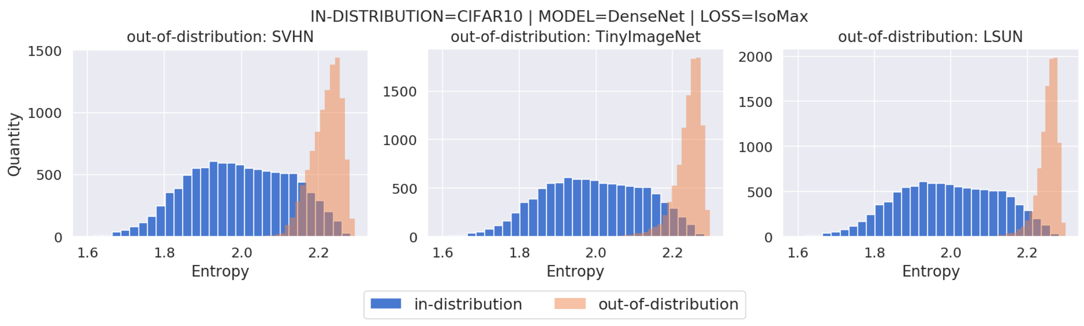
Table 14 - IsoMax Variants using Different Scores

Model	Data (training)	Comparison of IsoMax loss variants using different scores.	
		TNR@TPR95 (%) [↑]	AUROC (%) [↑]
		$\text{IsoMax}_{\text{ES}}$ / $\text{IsoMax}_{\text{MDS}}$ / $\text{IsoMax+}_{\text{ES}}$ / $\text{IsoMax+}_{\text{MDS}}$ (ours)	
DenseNet100	CIFAR10	84.2±4.6 / 0.9±0.5 / 89.3±2.3 / **94.6±1.5**	97.1±0.5 / 65.5±6.0 / 97.9±0.3 / **99.0±0.2**
	CIFAR100	46.5±5.0 / 6.2±6.1 / 63.7±8.0 / **80.6±3.6**	90.8±2.1 / 50.1±7.4 / 94.0±1.3 / **97.0±1.1**
ResNet110	CIFAR10	71.6±4.4 / 0.1±0.1 / 74.8±3.5 / **82.8±3.0**	95.2±0.7 / 83.7±4.1 / 95.2±0.6 / **97.3±0.7**
	CIFAR100	22.1±3.7 / 1.6±2.0 / 22.3±5.3 / **43.9±5.5**	83.3±1.8 / 61.9±7.2 / 84.4±0.8 / **88.6±2.1**

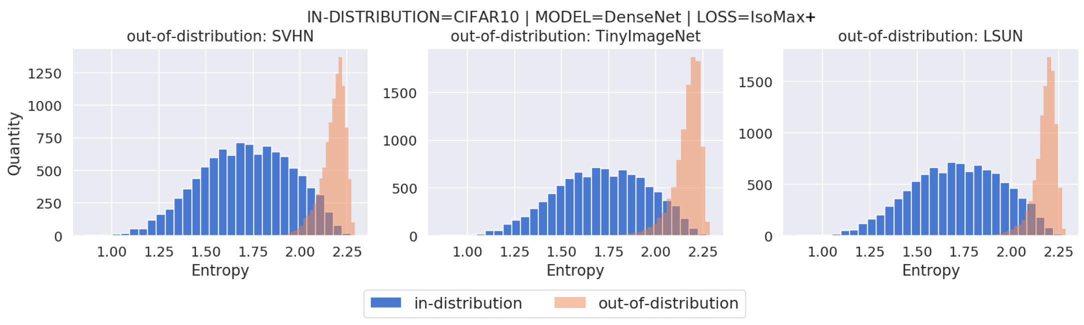
Source: The Author (2022). Comparison of IsoMax variants using different scores. $\text{IsoMax}_{\text{ES}}$ means training using IsoMax loss and performing OOD detection using the entropic score (ES). $\text{IsoMax}_{\text{MDS}}$ means training using IsoMax loss and performing OOD detection using the minimum distance score (MDS). $\text{IsoMax+}_{\text{ES}}$ means training using IsoMax+ loss and performing OOD detection using entropic score (ES). $\text{IsoMax+}_{\text{MDS}}$ (ours) means training using IsoMax+ loss and performing OOD detection using minimum distance score (MDS). The values of the performance metrics TNR@TPR95 and AUROC were averaged over all out-of-distribution (SVHN, TinyImageNet, and LSUN). Results are shown as the mean and standard deviation of five iterations. The best results are shown in bold. See Fig. 41.

Figure 41 - Minimum Distance Score Analyses

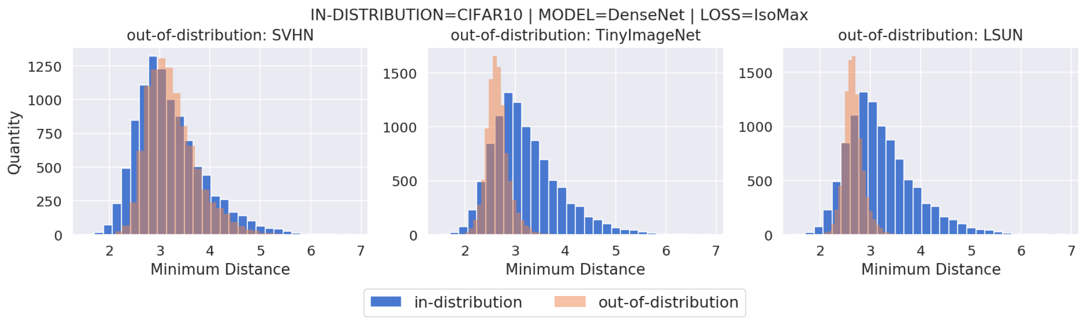
(a)



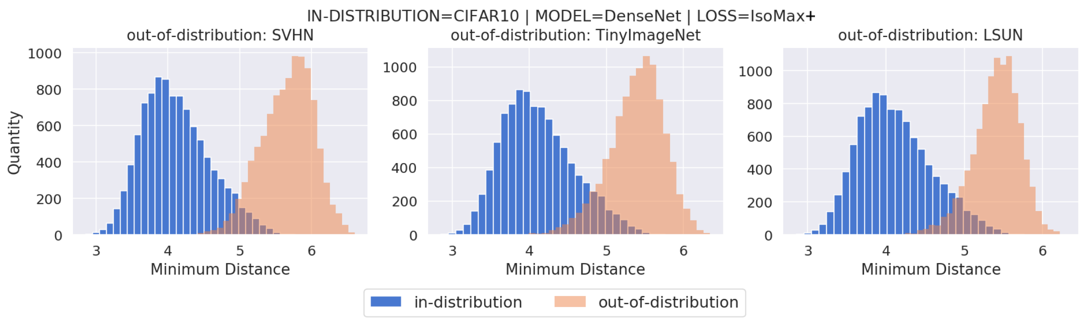
(b)



(c)



(d)



Source: The Authors (2022). In agreement with other studies (Liu *et al.*, 2020a), (a) IsoMax (b) and IsoMax+ produce higher entropy/uncertainty on out-of-distributions than in-distribution. Therefore, the entropic score produces high OOD detection performance in both cases. (c) However, IsoMax does not make the in-distribution closer to the prototypes than out-of-distributions. (d) Concurrently, by introducing *distance isometrization*, IsoMax+ gets in-distribution closer to the prototypes while pushing out-of-distribution data far away, which is what we expect based on the findings of other recent studies (Liu *et al.*, 2020a). This result also explains why the minimum distance score produces high performance when using IsoMax+, while producing low performance when using IsoMax. See also Table 14.

4.3 DISTINCTION MAXIMIZATION LOSS

To allow standardized comparison, we used the datasets, training procedures, and metrics that were established in Hendrycks & Gimpel (2017) and used in many subsequent papers (Liang *et al.*, 2018; Lee *et al.*, 2018; Hein *et al.*, 2019). We trained many 100-layer DenseNetBCs with growth rate $k = 12$ (i.e., 0.8M parameters) (Huang *et al.*, 2017), 34-layer ResNets He *et al.* (2016), and 28-layer WideResNets (widening factor $k = 10$) (Zagoruyko & Komodakis, 2016) on the CIFAR10 (Krizhevsky, 2009) and CIFAR100 (Krizhevsky, 2009) datasets with SoftMax, IsoMax+, and DisMax losses using the same procedures (e.g., initial learning rate, learning rate schedule, weight decay). We used DisMax for DenseNetBC100 trained on CIFAR10 because this is a very small model and the mentioned dataset has too many examples per class; therefore, no augmentation is needed. We used DisMax† with $\alpha = 1$ for all cases.

We used stochastic gradient descent (SGD) with the Nesterov moment equal to 0.9 with a batch size of 64 and an initial learning rate of 0.1. The weight decay was 0.0001, and we did not use dropout. We trained during 300 epochs. We used a learning rate decay rate equal to ten applied in epoch numbers 150, 200, and 250. We used resized images from TinyImageNet (Deng *et al.*, 2009), the Large-scale Scene UNderstanding dataset (LSUN) (Yu *et al.*, 2015), CIFAR10 (Krizhevsky, 2009), CIFAR100 (Krizhevsky, 2009), and SVHN (Netzer & Wang, 2011) to create out-of-distribution samples.

We added these out-of-distribution images to the validation sets of the ID data to form the test sets and evaluate the OOD detection performance. We evaluated the accuracy (ACC) to assess classification performance. We evaluated the OOD detection performance using the AUROC, the Area Under the Precision-Recall Curve (AUPR), and the true negative rate at a 95% true positive rate (TNR@TPR95). We used the Expected Calibration Error (ECE) (Naeini *et al.*, 2015; Guo *et al.*, 2017; Minderer *et al.*, 2021) for uncertainty estimation performance. The results are the mean and standard deviation of five runs. Two methods are considered to produce the same performance if their mean performance difference is less than the sum of the error margins.

4.3.1 Ablation Study

Table 15 shows that logits+ and FPR often improve the accuracy and OOD detection performance compared to IsoMax+ even when using MDS. It also shows that replacing MDS with the *composite* score MMLES often increases OOD detection. These conclusions are essentially true regardless of the model, in-distribution, and (near, far, and very far) out-of-distribution.

Finally, we performed experiments *combining IsoMax+ with CutMix*. However, adding CutMix to IsoMax+ did *not* significantly increase the OOD detection performance. Often, the performance actually *decreased*. Therefore, DisMax easily outperformed IsoMax+ even when the latter was combined with CutMix.

Table 15 - DisMax: Ablation Study

CIFAR10

Model	Method	Score	Out-of-Distribution Detection			
			Near	Far		Very Far
			CIFAR100	TinyImageNet	LSUN	SVHN
		MPS,MDS MPS/MMLES	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]
DenseNetBC100 (small size)	SoftMax (baseline)	MPS	39.5±2.1	53.1±7.8	62.1±6.2	41.2±3.6
	IsoMax+	MDS	**57.3±1.1**	86.9±0.4	91.4±0.3	96.4±0.6
	IsoMax+ with CutMix	MDS	41.4±0.8	73.8±8.5	87.4±3.2	59.7±9.3
	DisMax (ours)	MDS	**56.2±0.5**	**88.0±0.3**	**92.2±0.3**	97.5±0.3
		MPS/MMLES	54.2±1.0	**89.0±1.0**	**92.4±1.1**	**98.3±0.3**
ResNet34 (medium size)	SoftMax (baseline)	MPS	40.0±1.6	46.4±4.9	53.6±4.7	44.1±9.3
	IsoMax+	MDS	55.1±1.3	71.0±6.4	81.5±4.4	82.4±8.8
	IsoMax+ with CutMix	MDS	40.2±8.0	62.4±4.3	77.4±4.5	78.6±7.2
	DisMax† (ours)	MDS	**60.4±0.7**	**92.0±1.5**	97.2±0.4	**91.1±2.9**
		MPS/MMLES	**60.0±0.5**	**93.3±1.1**	**98.0±0.3**	**91.2±2.7**
WideResNet2810 (big size)	SoftMax (baseline)	MPS	44.9±0.7	53.4±3.3	59.2±3.6	50.1±5.2
	IsoMax+	MDS	61.5±0.4	80.2±4.2	87.4±3.0	**96.3±1.4**
	IsoMax+ with CutMix	MDS	37.4±8.0	76.3±8.2	85.8±6.3	60.0±9.8
	DisMax† (ours)	MDS	60.2±1.3	**98.4±0.4**	**99.4±0.1**	**93.8±1.7**
		MPS/MMLES	**62.9±0.5**	**98.6±0.2**	**99.5±0.1**	92.8±2.0

CIFAR100

Model	Method	Score	Out-of-Distribution Detection			
			Near	Far		Very Far
			CIFAR10	TinyImageNet	LSUN	SVHN
		MPS,MDS MPS/MMLES	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]
DenseNetBC100 (small size)	SoftMax (baseline)	MPS	17.6±1.1	18.1±1.7	18.7±2.0	19.8±2.9
	IsoMax+	MDS	17.2±0.7	71.6±6.5	66.8±9.4	**67.1±3.0**
	IsoMax+ with CutMix	MDS	**19.8±2.1**	60.8±9.9	57.6±9.8	57.9±3.5
	DisMax† (ours)	MDS	16.6±0.6	97.7±0.3	98.5±0.4	57.9±3.6
		MPS/MMLES	**22.1±1.1**	**99.0±0.5**	**99.4±0.3**	**66.6±2.6**
ResNet34 (medium size)	SoftMax (baseline)	MPS	19.4±0.5	20.6±2.4	21.3±3.4	17.1±5.0
	IsoMax+	MDS	18.0±0.7	43.3±4.3	41.5±5.7	43.6±3.5
	IsoMax+ with CutMix	MDS	18.9±1.7	46.3±9.6	46.5±9.0	35.6±3.9
	DisMax† (ours)	MDS	20.8±0.4	79.9±1.5	81.5±1.4	43.7±1.6
		MPS/MMLES	**22.0±0.5**	**85.4±1.7**	**86.4±1.3**	**48.5±2.0**
WideResNet2810 (big size)	SoftMax (baseline)	MPS	21.8±0.7	26.7±5.9	28.7±6.7	15.8±5.5
	IsoMax+	MDS	19.0±0.7	66.9±3.9	67.9±3.3	61.8±1.9
	IsoMax+ with CutMix	MDS	21.5±1.9	52.5±9.4	52.0±9.1	33.3±9.2
	DisMax† (ours)	MDS	22.4±0.2	92.3±1.3	95.2±0.4	56.8±1.8
		MPS/MMLES	**24.6±0.3**	**96.3±1.2**	**97.8±0.9**	**65.6±1.2**

Source: The Author (2022). MPS (Hendrycks & Gimpel, 2017) means Maximum Probability Score (i.e., the standard for SoftMax loss). MDS indicates Minimum Distance Score (i.e., the standard for IsoMax+ loss (Macêdo & Ludermir, 2021)). MMLES means Max-Mean Logit Entropy Score (i.e., the standard for DisMax loss for (very) far OOD detection). CutMix is from Yun *et al.* (2019). We used MPS for near OOD detection for DisMax, as this score provided the best results in this particular case. We emphasize that the MPS for DisMax is based on *logits+ rather than usual logits*. The best performances are bold. All results can be reproduced using the provided code.

Table 16 - DisMax: Classification, Efficiency, Uncertainty, and OOD Detection

CIFAR10

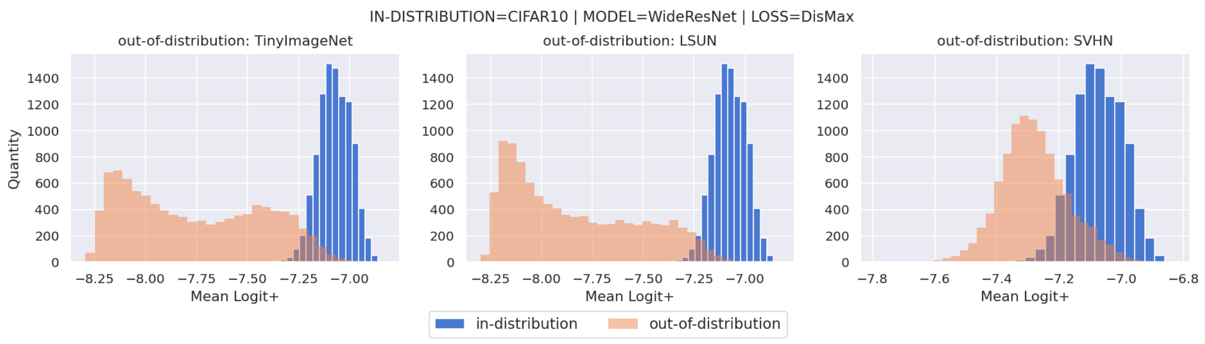
Model	Method	Classification	Inference	Uncertainty Estimation	Out-of-Distribution Detection			
					Near	Far		Very Far
					CIFAR100	TinyImageNet	LSUN	SVHN
		ACC (%) [↑]	Efficiency (%) [↑]	ECE [↓]	AUPR (%) [↑]	AUROC (%) [↑]	AUROC (%) [↑]	AUPR (%) [↑]
DenseNetBC100 (small size)	SoftMax (baseline)	**95.2±0.1**	**100.0**	**0.0043±0.0008**	86.2±0.5	92.9±1.6	94.7±0.9	93.7±3.3
	Scaled Cosine	94.9±0.1	**100.0**	-	-	**98.8±0.3**	**99.2±0.2**	-
	GODIN with preprocessing	95.0±0.1	26.0	-	-	**99.1±0.1**	**99.4±0.1**	-
	IsoMax+	**95.1±0.1**	100.0	**0.0043±0.0012**	**90.4±0.3**	97.6±0.9	98.3±0.5	99.7±0.1
	DisMax (ours)	**95.1±0.1**	**100.0**	**0.0045±0.0021**	**90.0±0.2**	98.0±0.5	98.4±0.3	**99.9±0.1**
ResNet34 (medium size)	SoftMax (baseline)	95.6±0.1	**100.0**	**0.0060±0.0013**	85.3±0.4	89.7±2.8	92.4±1.6	94.9±1.0
	GODIN	95.1±0.1	**100.0**	-	-	95.6±0.5	97.6±0.2	-
	IsoMax+	95.5±0.1	**100.0**	0.0133±0.0177	**90.1±0.3**	95.1±1.0	96.9±0.6	**98.7±0.6**
	DisMax† (ours)	**96.7±0.2**	**100.0**	**0.0058±0.0008**	**90.3±0.2**	**98.3±0.3**	**99.5±0.1**	**99.1±0.3**
WideResNet2810 (big size)	SoftMax (baseline)	96.2±0.1	**100.0**	**0.0038±0.0005**	87.5±0.3	92.6±0.9	94.0±0.7	95.3±0.9
	Deep Ensemble	96.6±0.1	10.3	0.0100±0.0010	88.8±1.0	-	-	96.4±1.0
	DUQ	94.7±0.1	45.0	0.0340±0.0020	85.4±1.0	-	-	97.3±1.0
	SNGP	95.9±0.1	62.5	0.0180±0.0010	90.5±1.0	-	-	99.0±1.0
	Scaled Cosine	95.7±0.1	**100.0**	-	-	97.7±0.7	98.6±0.3	-
	IsoMax+	96.0±0.1	**100.0**	0.0107±0.0166	**91.8±0.1**	96.6±0.6	97.7±0.4	**99.7±0.3**
	DisMax† (ours)	**97.0±0.1**	**100.0**	**0.0043±0.0008**	90.1±0.3	**99.7±0.1**	**99.9±0.1**	**99.3±0.3**

CIFAR100

Model	Method	Classification	Inference	Uncertainty Estimation	Out-of-Distribution Detection			
					Near	Far		Very Far
					CIFAR10	TinyImageNet	LSUN	SVHN
		ACC (%) [↑]	Efficiency (%) [↑]	ECE [↓]	AUPR (%) [↑]	AUROC (%) [↑]	AUROC (%) [↑]	AUPR (%) [↑]
DenseNetBC100 (small size)	SoftMax (baseline)	77.3±0.4	**100.0**	0.0155±0.0026	71.3±0.8	71.8±2.2	73.1±2.4	87.5±1.5
	Scaled Cosine	75.7±0.1	**100.0**	-	-	97.8±0.5	97.6±0.8	-
	GODIN with preprocessing	75.9±0.1	24.0	-	-	98.6±0.2	98.7±0.0	-
	IsoMax+	76.9±0.3	**100.0**	**0.0108±0.0017**	71.3±0.4	95.1±1.1	94.2±1.7	**97.4±0.6**
	DisMax† (ours)	**79.4±0.2**	**100.0**	0.0154±0.0006	**74.4±0.2**	**99.8±0.1**	**99.9±0.1**	**96.4±0.8**
ResNet34 (medium size)	SoftMax (baseline)	77.7±0.3	**100.0**	0.0268±0.0015	73.3±0.1	79.0±2.1	79.6±1.7	86.3±3.3
	GODIN	75.8±0.2	**100.0**	-	-	91.8±1.1	92.0±0.7	-
	GODIN with dropout	77.2±0.1	**100.0**	-	-	87.0±1.1	87.0±2.2	-
	IsoMax+	76.5±0.3	**100.0**	0.0190±0.0025	72.1±0.4	89.7±1.0	89.8±1.3	**94.5±0.6**
	DisMax† (ours)	**80.6±0.3**	**100.0**	**0.0116±0.0014**	**74.2±0.6**	**97.6±0.5**	**97.7±0.6**	**94.8±1.0**
WideResNet2810 (big size)	SoftMax (baseline)	79.9±0.2	**100.0**	0.0272±0.0032	75.4±0.5	81.7±2.3	82.7±2.2	86.0±2.6
	Deep Ensemble	80.2±0.1	12.3	0.0210±0.0040	78.0±1.0	-	-	88.8±1.0
	DUQ	78.5±0.1	79.9	0.1190±0.0010	73.2±1.0	-	-	87.8±1.0
	SNGP	79.9±0.1	74.9	0.0250±0.0120	**80.1±1.0**	-	-	92.3±1.0
	Scaled Cosine	78.5±0.3	**100.0**	-	-	95.8±0.7	95.2±0.8	-
	IsoMax+	79.5±0.1	**100.0**	0.0188±0.0016	73.0±0.8	94.2±2.1	94.6±2.0	**96.7±1.7**
	DisMax† (ours)	**83.0±0.1**	**100.0**	**0.0143±0.0027**	76.0±1.0	**99.4±0.2**	**99.6±0.1**	**97.0±1.5**

Source: The Author (2022). In this table, efficiency represents the inference speed (i.e., the inverse of the inference delay) calculated as a percentage of the performance of a single deterministic neural network trivially trained. For a fair comparison, we also calibrated the temperature of the SoftMax loss and IsoMax+ loss approaches using the same procedure that we defined for DisMax loss. Considering that input preprocessing can be applied indistinctly to improve the OOD detection performance of all methods compared (Hsu *et al.*, 2020) (at the cost of making their inferences approximately four times less efficient (Macêdo *et al.*, 2022)), unless explicitly mentioned otherwise, all results are presented without using input preprocessing. The results worse than the baseline or most of the other approaches are shown in red. The methods that present the best performances are bold. SoftMax (baseline) was proposed in Hendrycks & Gimpel (2017) and IsoMax+ in Macêdo & Ludermir (2021). Results for Scaled Cosine are from Scaled Cosine paper (Techapanurak & Okatani, 2019). Results for GODIN are from GODIN paper (Hsu *et al.*, 2020). Results for Deep Ensemble (Lakshminarayanan *et al.*, 2017), DUQ (van Amersfoort *et al.*, 2020), and SNGP are from SNGP paper (Liu *et al.*, 2020a).

Figure 42 - Max-Mean Logit Entropy Score Analyses



Source: The Author (2022). In the feature space, the mean distance from an in-distribution image to *all* prototypes is usually smaller than the mean distance from an out-of-distribution image to *all* prototypes. For example, consider a given class present in CIFAR10. This figure shows that even prototypes associated with classes *other than the selected class* are usually closer to images of the assumed class (in-distribution in blue) than images that do not belong to CIFAR10 at all (out-of-distributions in orange). This explains why the *mean value* of logits+ considering all prototypes contributes to the OOD detection performance. Therefore, not only the distance to the nearest prototype is used in the mentioned task.

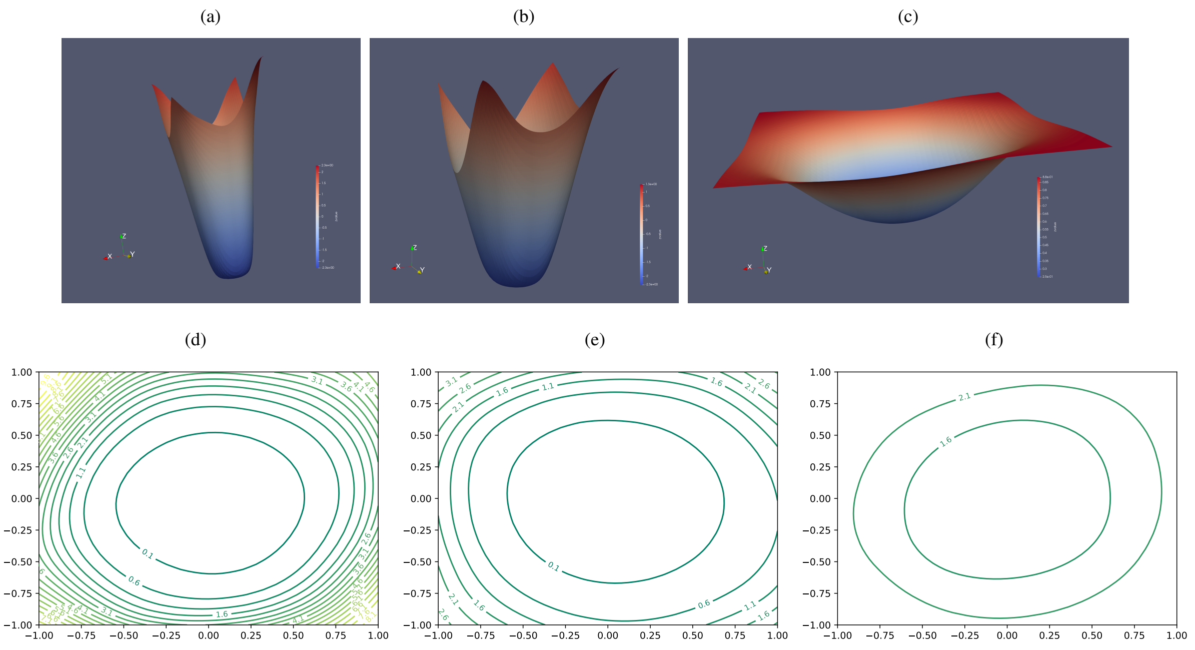
4.3.2 Classification, Efficiency, Uncertainty, and OOD Detection Results

Table 16 compares DisMax with major approaches such as Scaled Cosine (Techapanurak & Okatani, 2019), GODIN (Hsu *et al.*, 2020), Deep Ensemble (Lakshminarayanan *et al.*, 2017), DUQ (van Amersfoort *et al.*, 2020), and SNGP (Liu *et al.*, 2020a) regarding classification accuracy, inference efficiency, uncertainty estimation, and (near, far, and very far) out-of-distribution detection. Unlike other approaches, DisMax is as inference efficient as a trivially trained neural network using the usual SoftMax loss. Furthermore, DisMax often outperforms other approaches simultaneously in all evaluated metrics.

4.3.3 Max-Mean Logit Entropy Score Analyses

Fig. 41 shows the distribution of *mean* logits+ under some scenarios. We see that prototypes are, on average, usually closer to in-distribution examples than out-of-distribution examples, which explains why the *mean* enhanced logit improves OOD detection performance when combined with the maximum logit+ and the negative entropy to compose the MMLES. In other words, even prototypes *that are not associated with the class of a given in-distribution example* are usually closer to it than they are to out-of-distribution examples.

Figure 43 - Loss Surface Study



Source: The Author (2022). 3D loss surfaces and 2D loss contours as proposed in Li *et al.* (2018). Loss landscapes for ResNet34 trained on CIFAR10. (a, d) SoftMax; (b, e) IsoMax+; and (c, f) DisMax†. Considering that IsoMax+ outperforms SoftMax and DisMax† outperforms IsoMax+, a less steep 3D inclination (i.e. a lower 2D contour concentration) provides increased robustness.

4.3.4 Loss Landscape Study

Fig. 43 presents the 3D loss surfaces and 2D loss contours as studied in Li *et al.* (2018). Considering that IsoMax+ overcomes SoftMax and that DisMax† outperforms IsoMax+, we conclude that improved robustness is obtained by less steep 3D inclination (i.e. a lower 2D contour concentration).

The above-mentioned conclusion may be somewhat unexpected, as we always were conducted to believe that losses with high inclinations are “better”. It is the case probably because we were not used to thinking that we need losses that produce robust models rather than only losses that build accurate ones.

5 CONCLUSION

"There can be no triumph without loss.
No victory without suffering. No freedom without sacrifice."
–The Lord of the Rings: The Return of the King (Official Trailer)

"Become who you are!"
–Friedrich Nietzsche

"Seize the present, trust tomorrow e'en as little as you may."
–Horace (Odes)

In this work, we mentioned that AI is the subfield of computer science that handles problems not well-fitted for the classical algorithmic programming paradigm. We also showed the relevance and limitations of machine learning. We presented the revolution that deep learning represents for the machine learning community. Finally, we showed the deep learning strengths and current major limitations such as reasoning, causal inference, interpretability, and robustness.

Regarding robustness, we exemplified how current deep networks present extremely confident predictions even when they are wrong. We also gave examples of how hard it is to apprehend when the system cannot reliably predict. Subsequently, we presented the OOD detection task, which is one of the challenges that best outlines the issue of reliability in deep learning.

Finally, after identifying the SoftMax loss limitations, we present our proposals to tackle the OOD detection problem. We started from deep learning foundations such as loss function design and the principle of maximum entropy to develop solutions that do not rely on ad hoc techniques currently used to attack this problem. By doing this, we were able to achieve state-of-the-art OOD detection performance, avoiding drawbacks of previously proposed methods.

5.1 CONTRIBUTIONS

5.1.1 Isotropy Maximization Loss

Initially, we proposed IsoMax, a loss that is isotropic (exclusively distance-based) and produces high entropy posterior probability distributions in agreement with the maximum entropy principle. We additionally proposed the entropic score, which is a fast and efficient way to perform OOD detection using the information provided by all neural network output probabilities rather than just one, usually the case in current OOD detection approaches.

The proposed approach avoids the techniques, requirements, and side effects used by current methods. Networks trained using IsoMax loss produce accurate predictions, as no classification accuracy drop is observed compared to networks trained with SoftMax loss. Additionally, the models trained using IsoMax loss provide inferences that are fast and have energy and computational efficiency equivalent to models trained with SoftMax loss, making our solution viable from an economical and environmental point of view (Schwartz *et al.*, 2019).

Moreover, our approach does not require *hyperparameter tuning*, which means it does not require validation using unrealistic design-time access to OOD samples or the generation of adversarial examples. Regarding this point, we remember that *all* experiments performed in this work always used the same constant value for the Entropic Scale.

Indeed, no hyperparameter tuning is required because the entropic scale is a global constant that is kept equal to ten for all combinations of datasets and models. Even if we call the entropic scale hyperparameter, the IsoMax loss does not involve hyperparameter tuning because the same constant value of entropic scale is used in all cases. This result is possible because it was shown in Macêdo *et al.* (2021, 2022) that the OOD detection performance exhibits a *well-behaved dependence* on the entropic scale regardless of the dataset and model.

Furthermore, our solution does not require feature extraction, metric learning, or hyperparameter tuning. In other words, no extra procedures other than typical neural network training are required. Considering our approach avoids ad hoc techniques and associated troublesome requirements and undesired side effects, we say that IsoMax loss works as a SoftMax loss *drop-in replacement* and that the overall solution is seamless.

We provide theoretical foundations based on the maximum entropy principle to explain why our *seamless and principled approach* works. Substantial experimental evidence confirms our theoretical assumptions and shows that our solution presents *competitive* OOD detection performance in addition to avoiding limitations of current methods.

Additionally, considering that the replacement of the SoftMax loss by the IsoMax loss significantly improves OOD detection performance, we conclude that the general low OOD detection performance of current neural networks is due to SoftMax loss drawbacks rather than limitations of the models.

Nevertheless, when the cited requirements and side effects are not a concern for a particular real-world application, the mentioned (or other) techniques may be combined with IsoMax loss to try to achieve even higher OOD detection performance.

For example, loss enhancement techniques, such as outlier exposure or background samples, may be readily adapted to work with the IsoMax loss. Another promising approach could be using recent data augmentation techniques (Thulasidasan *et al.*, 2019; Yun *et al.*, 2019) or strategies based on pretrained models (Sastry & Oore, 2019).

5.1.2 Enhanced Isotropy Maximization Loss

We improved the IsoMax loss function by replacing its original distance with what we call the *isometric distance*. Additionally, we proposed a zero computational cost minimum distance score. Experiments showed that these modifications achieve higher OOD detection performance while maintaining the desired benefits of IsoMax loss (i.e., absence of hyperparameters to tune, no reliance on additional/outlier/background data, fast and power-efficient inference, and no classification accuracy drop).

Similar to IsoMax loss, after training using the proposed IsoMax+ loss, we may apply inference-based approaches (e.g., ODIN, Mahalanobis, Gram matrices, outlier exposure, energy-based) to the pretrained model to eventually increase the overall OOD detection performance even more. Thus, the IsoMax+ loss is a replacement for SoftMax loss but not for OOD methods that may be applied to pretrained models, which may be used to improve even more the OOD detection performance of IsoMax+ loss pretrained networks.

There is no drawback in training a model using IsoMax+ loss instead of SoftMax loss or IsoMax loss, regardless of planning to subsequently use an inference-based OOD detection approach to further increase OOD detection performance. Therefore, instead of competitors, the OOD detection approaches that may be applied to pretrained models are actually complementary to the proposed approach (Macêdo *et al.*, 2021, 2022). IsoMax+ loss achieves a better baseline performance than SoftMax loss or IsoMax loss to construct future OOD detection methods.

5.1.3 Distinction Maximization Loss

We proposed DisMax by improving the IsoMax+ with the *enhanced* logits and the *Fractional Probability Regularization*. We also presented a novel *composite* score called MMLES for OOD detection by combining the maximum logit+, the *mean* logit+, and the negative entropy of the network output. We present a *simple and fast temperature scaling procedure* performed after training that makes DisMax produce a high-performance uncertainty estimation. Our experiments showed that the proposed method commonly outperforms the current approaches simultaneously in classification accuracy, inference efficiency, uncertainty estimation, and out-of-distribution detection.

5.2 BENEFITS OF ENTROPIC LOSSES

Regarding the first research question, we showed that it is possible to perform state-of-the-art OOD replacing just the loss, and we also provided many options for scores. Concerning the second question, we showed that the maximum entropy principle offers solid theoretical motivation for the proposed approaches. Finally, to answer the third question and to summarize our achievements, here we present some features that are common to the proposed solutions (i.e., the entropic losses and the novel scores):

1. Entropic losses do not present classification accuracy drop: The experiments show that training with entropic losses does not produce a classification accuracy drop. Other training-based approaches like Generalized ODIN (G-ODIN) require losing some training data for validation, consequently producing a classification accuracy drop compared with training the SoftMax loss. Losing training data for validation is particularly harmful in the critical low labeled data regime. Classification accuracy drop is usually extremely undesired in many cases in practice.

2. Models trained using entropic losses do not produce slow and energy inefficient inferences: Approaches such as ODIN, full Mahalanobis, and G-ODIN incorporate input preprocessing, making the solution at least four times slower and energy inefficient. We should prefer environment-friendly and low-energy consumption solutions for real-world, large-scale adoption.

3. Entropic losses do not require additional data collection: Solutions like outlier exposure require extra data collection, while the proposed approach does not. If additional data is available, outlier exposure (or similar data-driven techniques) may be used to enhance the performance of our method.

4. IsoMax, IsoMax+ and DisMax do not require hyperparameter tuning: It makes the solution readily available to be fast adopted in practice and significantly increases the overall baseline OOD performance in many areas and applications.

5. Scalability: considering the solution consists of a SoftMax loss drop-in replacement, we have strong reasons to believe that the proposed approach scales well for large image datasets.

6. Domain-Agnostic: Considering that IsoMax, IsoMax+, and DisMax work at loss level and avoids data augmentation, it may potentially be applied to text.

7. Easy of use: considering that the proposed solutions work as seamless SoftMax loss drop-in replacement, using it is as simple as replacing two lines of code.

8. Compatibility with existing inference-based approaches: inference-based approaches (e.g., ODIN, vanilla or full Mahalanobis, outlier exposure, Gram matrices) may be applied to our losses pretrained model just like they are applied to SoftMax loss pretrained models to achieve even higher OOD detection performance. Unlike inference-based approaches that usually increase the computational cost of performing inferences or OOD detection, the proposed solutions produce inferences as fast and computationally efficient as pure SoftMax loss-trained models.

9. Competitive state-of-the-art performance even operation under more restrictive conditions and producing no side effects: Our results show that the proposed approaches overcome ODIN despite avoiding inefficient inference. It also usually overcomes full Mahalanobis, avoiding inefficient inferences and validations using adversarial examples. Moreover, it is competitive or overcomes outlier exposure without relying on extra data.

10. Simplicity: The simplicity and solid foundations (e.g., distance-based loss, the principle of maximum entropy, isometric distances, minimum distance score) suggest that the proposed solutions generalize well.

11. Compatibility with existing Bayesian, ensemble, and uncertainty estimation techniques: As the entropic losses work as a SoftMax loss drop-in replacement, all available Bayesian, ensemble, and uncertainty estimation techniques may be immediately combined with it to start from a much better baseline than SoftMax loss.

12. Considering that the minimum distance is computed during the classification phase and that this value is reused as the score to perform OOD detection, we say that this is a zero computational cost out-of-distribution detection. Hence, unlike other approaches (e.g., Gram matrices), no extra computation is required.

13. Do not require feature extraction or training additional models: After the neural network training, the solution is readily available. It is unnecessary to perform feature extraction or additional train models on the extracted features.

5.3 FUTURE WORKS

In future work, we intend to analyze the behavior of our solutions when dealing with natural corruptions and perturbations. Some studies have shown that the performance of deep neural networks deteriorates when submitted to corruptions and perturbations that may occur naturally (Hendrycks & Dietterich, 2019). Therefore, in some applications, it is important to design robust solutions in these cases. Robustness in the presence of distribution shifts[1] is a relevant case that we also intend to study.

Besides the natural corruptions and perturbations, we also plan to evaluate the performance of our solutions when dealing with intentionally manipulated corruptions and perturbations. In other words, we intend to analyze the behavior of our solutions when submitted to the so-called adversarial attacks (Wang *et al.*, 2021). Making the solutions robust against adversarial examples may allow its usage in critical scenarios in which this is a real concern.

We also plan to study the behavior of the proposed losses when dealing with other types of media (e.g., audio), structured data (tabular and temporal data), and tasks (e.g., object detection (Zaidi *et al.*, 2021)). We may broaden their practical applications by showing that the proposed losses also present satisfactory behavior in such cases.

Finally, we intend to verify the performance of our approach using transformer-based models, regardless of being pretrained or fine-tuned using the proposed losses and scores. Considering that vision transformers are recently becoming a reality, evaluate the performance of our proposals in such cases is becoming important.

[1] https://wilds.stanford.edu/

REFERENCES

Alam, M., Samad, M. D., Vidyaratne, L., Glandon, A., & Iftekharuddin, K. M. (2020). Survey on deep neural networks in speech and vision systems. *Neurocomputing*, 417:302–321. 19

Albelwi, S. (2022). Survey on self-supervised learning: Auxiliary pretext tasks and contrastive learning methods in imaging. *Entropy*, 24(4):551. 22

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*. 19

Bellman, R. & Collection, K. M. R. (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University Press. 15

Bellman, R. & Karush, R. (1964). *Dynamic Programming: A Bibliography of Theory and Application*. Rand Corporation. 15

Bendale, A. & Boult, T. E. (2015). Towards open world recognition. *Computer Vision and Pattern Recognition*. 30, 43, 51

Bengio, Y. (2009). Learning deep architectures for AI. *Found. Trends Mach. Learn.* 18

Bengio, Y. (2015). Deep learning: Theoretical motivations. *videolectures.net*. 16

Bengio, Y., Courville, A. C., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828. 18

Bengio, Y., Delalleau, O., & Roux, N. L. (2005). The curse of highly variable functions for local kernel machines. *Neural Information Processing Systems*. 15, 18

Bengio, Y. & Lecun, Y. (2007). *Scaling Learning Algorithms toward AI*. MIT Press. 18

Berger, A. L., Pietra, S. D., & Pietra, V. J. D. (1996). A maximum entropy approach to natural language processing. *Comput. Linguistics*, 22(1):39–71. 56

Bochkovskiy, A., Wang, C., & Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv*, abs/2004.10934. 69

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., & Kasneci, G. (2021). Deep neural networks and tabular data: A survey. *arXiv*, abs/2110.01889. 19

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners. *Neural Information Processing Systems*. 19

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208. 70

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I. J., Madry, A., & Kurakin, A. (2019). On evaluating adversarial robustness. *arXiv*, abs/1902.06705. 44

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848. 28

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. E. (2020a). A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*. 20, 22

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. E. (2020b). A simple framework for contrastive learning of visual representations. *arXiv*, abs/2002.05709. 28

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *Empirical Methods in Natural Language Processing*. 72

Cobbe, K., Kosaraju, V., Bavarian, M., Hilton, J., Nakano, R., Hesse, C., & Schulman, J. (2021). Training verifiers to solve math word problems. *arXiv*, abs/2110.14168. 25, 26

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms*. MIT Press. 25

Cover, T. M. & Thomas, J. A. (2006). *Elements of information theory*. Wiley. 46

Deiana, A. M., Tran, N., Agar, J., Blott, M., Di Guglielmo, G., Duarte, J., Harris, P., Hauck, S., Liu, M., Neubauer, M. S., Ngadiuba, J., Ogrenci-Memik, S., Pierini, M., Aarrestad, T., Bahr, S., Becker, J., Berthold, A.-S., Bonventre, R. J., Bravo, T. E. M., Diefenthaler, M., Dong, Z., Fritzsche, N., Gholami, A., Govorkova, E., Hazelwood, K. J., Herwig, C., Khan, B., Kim, S., Klijnsma, T., Liu, Y., Lo, K. H., Nguyen, T., Pezzullo, G., Rasoulinezhad, S., Rivera, R. A., Scholberg, K., Selig, J., Sen, S., Strukov, D., Tang, W., Thais, S., Unger, K. L., Vilalta, R., Krosigk, B., Warburton, T. K., Flechas, M. A., Aportela, A., Calvet, T., Cristella, L., Diaz, D., Doglioni, C., Galati, M. D., Khoda, E. E., Fahim, F., Giri, D., Hawks, B., Hoang, D., Holzman, B., Hsu, S.-C., Jindariani, S., Johnson, I., Kansal, R., Kastner, R., Katsavounidis, E., Krupa, J., Li, P., Madireddy, S., Marx, E., McCormack, P., Meza, A., Mitrevski, J., Mohammed, M. A., Mokhtar, F., Moreno, E., Nagu, S., Narayan, R., Palladino, N., Que, Z., Park, S. E., Ramamoorthy, S., Rankin, D., Rothman, S., Sharma, A., Summers, S., Vischia, P., Vlimant, J.-R., & Weng, O. (2021). Applications and techniques for fast machine learning in science. *arXiv*, abs/2110.13041. 19

Delalleau, O. & Bengio, Y. (2011). Shallow vs. deep sum-product networks. *Neural Information Processing Systems*. 18

Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Li, F. (2009). Imagenet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition*. 18, 72, 92, 98

Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. *Computer Vision and Pattern Recognition*. 58

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. *North American Chapter of the Association for Computational Linguistics*. 19, 20

Devries, T. & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv*, abs/1708.04552. 47

DeVries, T. & Taylor, G. W. (2018). Learning confidence for out-of-distribution detection in neural networks. *arXiv*, abs/1802.04865. 41

Dhamija, A. R., Günther, M., & Boult, T. E. (2018). Reducing network agnostophobia. *Neural Information Processing Systems*. 43

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*. 19, 20

Dubey, A., Gupta, O., Raskar, R., & Naik, N. (2018). Maximum-entropy fine grained classification. *Neural Information Processing Systems*. 56

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2021). A survey of quantization methods for efficient neural network inference. *arXiv*, abs/2103.13630. 26

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *International Conference on Artificial Intelligence and Statistics*, 9. 58

Goodfellow, I. J., Bengio, Y., & Courville, A. C. (2016). *Deep Learning*. MIT Press. 15, 19

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *Int. J. Comput. Vis.*, 129(6):1789–1819. 26

Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., Guo, Z., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., & Valko, M. (2020). Bootstrap your own latent: A new approach to self-supervised learning. *Neural Information Processing Systems*. 28

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. *International Conference on Machine Learning*. 31, 32, 35, 56, 70, 98

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2021). Masked autoencoders are scalable vision learners. *arXiv*, abs/2111.06377. 21, 22

He, K., Gkioxari, G., Dollár, P., & Girshick, R. B. (2017). Mask R-CNN. *International Conference on Computer Vision*. 28

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *International Conference on Computer Vision*. 58

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. *European Conference on Computer Vision*. 72, 92, 98

Hein, M., Andriushchenko, M., & Bitterwolf, J. (2019). Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *Computer Vision and Pattern Recognition*. 35, 42, 43, 51, 82, 92, 98

Hendrycks, D. & Dietterich, T. G. (2019). Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*. 108

Hendrycks, D. & Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations*. 30, 31, 81, 82, 89, 92, 98, 99, 100

Hendrycks, D., Mazeika, M., & Dietterich, T. G. (2019a). Deep anomaly detection with outlier exposure. *International Conference on Learning Representations*. 42, 43, 72, 80, 96

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., & Song, D. (2019b). Natural adversarial examples. *arXiv*, abs/1907.07174. 29, 90

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6):82–97. 19

Hsu, Y., Shen, Y., Jin, H., & Kira, Z. (2020). Generalized ODIN: detecting out-of-distribution image without learning from out-of-distribution data. *Computer Vision and Pattern Recognition*. 41, 42, 43, 44, 89, 100, 101

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Computer Vision and Pattern Recognition*. 72, 92, 98

Hughes, G. F. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Trans. Inf. Theory*, 14(1):55–63. 15

Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., & Makedon, F. (2020). A survey on contrastive self-supervised learning. *arXiv*, abs/2011.00362. 18

Jaynes, E. T. (1957a). Information theory and statistical mechanics. *Physical Review*, 106:620–630. 46

Jaynes, E. T. (1957b). Information theory and statistical mechanics. II. *Physical Review*, 108:171–190. 46

Jing, L. & Tian, Y. (2021). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4037–4058. 18

Kendall, A. & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? *Neural Information Processing Systems*. 43

Kingma, D. P. & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations*. 76

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. *arXiv*, abs/2205.11916. 26, 27

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *Science Department, University of Toronto*. 72, 92, 98

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*. 19

Krueger, D., Caballero, E., Jacobsen, J., Zhang, A., Binas, J., Priol, R. L., & Courville, A. C. (2020). Out-of-distribution generalization via risk extrapolation (rex). *arXiv*, abs/2003.00688. 29

Kuleshov, V., Fenner, N., & Ermon, S. (2018). Accurate uncertainties for deep learning using calibrated regression. *International Conference on Machine Learning*, 80. 43

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Neural Information Processing Systems*. 42, 43, 100, 101

Le-Khac, P. H., Healy, G., & Smeaton, A. F. (2020). Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934. 18

Lee, D. & Cheon, Y. (2020). Soft labeling affects out-of-distribution detection of deep neural networks. *arXiv*, abs/2007.03212. 80

Lee, K., Lee, K., Lee, H., & Shin, J. (2018). A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Neural Information Processing Systems*. 41, 42, 51, 72, 82, 89, 92, 93, 95, 98

Leibig, C., Allken, V., Ayhan, M. S., Berens, P., & Wahl, S. (2017). Leveraging uncertainty information from deep neural networks for disease detection. *Scientific Reports*, 7. 43

Lemos, P., Jeffrey, N., Cranmer, M., Ho, S., & Battaglia, P. (2022). Rediscovering orbital mechanics with machine learning. *arXiv*, abs/2202.02306. 19

Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. *Neural Information Processing Systems*. 102

Li, J., Sun, A., Han, J., & Li, C. (2022). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70. 28

Liang, S., Li, Y., & Srikant, R. (2018). Enhancing the reliability of out-of-distribution image detection in neural networks. *International Conference on Learning Representations*. 41, 42, 72, 80, 82, 89, 92, 93, 98

Liang, T., Glossner, J., Wang, L., Shi, S., & Zhang, X. (2021). Pruning and quantization for deep neural network acceleration: A survey. *arXiv*, abs/2101.09671. 26

Lim, B. & Zohren, S. (2020). Time series forecasting with deep learning: A survey. *arXiv*, abs/2004.13408. 19

Liu, J. Z., Lin, Z., Padhy, S., Tran, D., Bedrax-Weiss, T., & Lakshminarayanan, B. (2020a). Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Neural Information Processing Systems*. 42, 44, 97, 100, 101

Liu, W., Wang, X., Owens, J. D., & Li, Y. (2020b). Energy-based out-of-distribution detection. *arXiv*, abs/2010.03759. 42, 43, 95

Liu, W., Wen, Y., Yu, Z., & Yang, M. (2016). Large-margin softmax loss for convolutional neural networks. *International Conference on Machine Learning*, 48. 35, 45, 50, 58

Liu, X., Zhang, F., Hou, Z., Wang, Z., Mian, L., Zhang, J., & Tang, J. (2020c). Self-supervised learning: Generative or contrastive. *arXiv*, abs/2006.08218. 18

Macêdo, D., Ren, T. I., Zanchettin, C., Oliveira, A. L. I., & Ludermir, T. B. (2021). Entropic out-of-distribution detection. *International Joint Conference on Neural Networks*. 62, 66, 70, 94, 95, 104, 105

Macêdo, D., Ren, T. I., Zanchettin, C., Oliveira, A. L. I., & Ludermir, T. B. (2022). Entropic out-of-distribution detection: Seamless detection of unknown examples. *IEEE Transactions on Neural Networks and Learning Systems.*, 33(6):2350–2364. 44, 66, 70, 94, 96, 100, 104, 105

Macêdo, D. & Ludermir, T. (2021). Enhanced isotropy maximization loss: Seamless and high-performance out-of-distribution detection simply replacing the softmax loss. *arXiv*, abs/2105.14399. 44, 65, 66, 70, 99, 100

Macêdo, D., Ren, T. I., Zanchettin, C., Oliveira, A. L. I., & Ludermir, T. (2019). Entropic out-of-distribution detection (first version). *arXiv*, abs/1908.05569v1. 44, 70

Malinin, A. & Gales, M. J. F. (2018). Predictive uncertainty estimation via prior networks. *Neural Information Processing Systems*. 43

Mehta, S. & Rastegari, M. (2021). Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv*, abs/2110.02178. 26

Mensink, T., Verbeek, J. J., Perronnin, F., & Csurka, G. (2013). Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2624–2637. 51, 54

Miller, D., Rao, A., Rose, K., & Gersho, A. (1995). A maximum entropy approach for optimal statistical classification. *IEEE Workshop on Neural Networks for Signal Processing*. 56

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2021). Deep learning–based text classification: A comprehensive review. *ACM Comput. Surv.*, 54(3). 28

Minderer, M., Djolonga, J., Romijnders, R., Hubis, F., Zhai, X., Houlsby, N., Tran, D., & Lucic, M. (2021). Revisiting the calibration of modern neural networks. *Neural Information Processing Systems*. 98

Montúfar, G. F. & Morton, J. (2015). When does a mixture of products contain a product of mixtures? *SIAM J. Discret. Math.*, 29(1):321–347. 18

Montúfar, G. F., Pascanu, R., Cho, K., & Bengio, Y. (2014). On the number of linear regions of deep neural networks. *Neural Information Processing Systems*. 18

Morales, J. L. & Nocedal, J. (2011). Remark on "algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound constrained optimization". *ACM Trans. Math. Softw.*, 38(1):7:1–7:4. 70

Musgrave, K., Belongie, S. J., & Lim, S. (2020). A metric learning reality check. *arXiv*, abs/2003.08505. 52

Naeini, M. P., Cooper, G. F., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. *AAAI Conference on Artificial Intelligence*. 98

Netzer, Y. & Wang, T. (2011). Reading digits in natural images with unsupervised feature learning. *Neural Information Processing Systems. Workshop on Deep Learning and Unsupervised Feature Learning*. 72, 92, 98

Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. *ACM Comput. Surv.*, 54(2). 30

Papadopoulos, A.-A., Rajati, M. R., Shaikh, N., & Wang, J. (2019). Outlier exposure with confidence control for out-of-distribution detection. *arXiv:1906.03509*. 43

Pascanu, R., Montúfar, G., & Bengio, Y. (2014). On the number of inference regions of deep feed forward networks with piece-wise linear activations. *International Conference on Learning Representations*. 18

Patterson, D., Gonzalez, J., Hölzle, U., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., & Dean, J. (2022). The carbon footprint of machine learning training will plateau, then shrink. *arXiv*, abs/2204.05149. 26

Pearl, J. (1989). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann. 56

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., & Hinton, G. E. (2017). Regularizing neural networks by penalizing confident output distributions. *International Conference on Learning Representations*. 56

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. *International Conference on Machine Learning*, 139. 21, 22

Raghunathan, A., Xie, S. M., Yang, F., Duchi, J. C., & Liang, P. (2019). Adversarial training can hurt generalization. *arXiv*, abs/1906.06032. 43

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv*, abs/2204.06125. 24, 26

Redmon, J., Divvala, S. K., Girshick, R. B., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Computer Vision and Pattern Recognition*. 28

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., & de Freitas, N. (2022). A generalist agent. *arXiv*, abs/2205.06175. 22

Ren, S., He, K., Girshick, R. B., & Sun, J. (2017). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149. 28

Rudd, E. M., Jain, L. P., Scheirer, W. J., & Boult, T. E. (2018). The extreme value machine. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(3):762–768. 30, 43, 51

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., & Norouzi, M. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *arXiv*, abs/2205.11487. 23, 26

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229. 15

Samuel, A. L. (1967). Some studies in machine learning using the game of checkers. II—recent progress. *IBM Journal of Research and Development*, 11(6):601–617. 15

Sastry, C. S. & Oore, S. (2019). Detecting out-of-distribution examples with in-distribution examples and gram matrices. *arXiv*, abs/1912.12510. 42, 43, 105

Scheirer, W. J., Jain, L. P., & Boult, T. E. (2014). Probability models for open set recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(11):2317–2324. 30, 43, 51

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *Computer Vision and Pattern Recognition*. 45

Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2019). Green AI. *arXiv*, abs/1907.10597. 41, 43, 104

Shafaei, A., Schmidt, M., & Little, J. J. (2019). A less biased evaluation of out-of-distribution sample detectors. *British Machine Vision Conference*. 41, 43, 49

Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J. P., Studer, C., Davis, L. S., Taylor, G., & Goldstein, T. (2019). Adversarial training for free! *Neural Information Processing Systems*. 43

Shannon, C. E. (1948a). A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423. 59

Shannon, C. E. (1948b). A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(4):623–656. 59

Shawe-Taylor, J. & Hardoon, D. R. (2009). Pac-bayes analysis of maximum entropy classification. *International Conference on Artificial Intelligence and Statistics*, 5. 56

Snell, J., Swersky, K., & Zemel, R. S. (2017). Prototypical networks for few-shot learning. *Neural Information Processing Systems*. 45, 54

Subramanya, A., Srinivas, S., & Babu, R. V. (2017). Confidence estimation in deep neural networks via density modelling. *arXiv*, abs/1707.07013. 43

Sun, Y., Chen, Y., Wang, X., & Tang, X. (2014). Deep learning face representation by joint identification-verification. *Neural Information Processing System*. 45

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Neural Information Processing Systems*. 19

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Computer Vision and Pattern Recognition*. 80

Tack, J., Mo, S., Jeong, J., & Shin, J. (2020). CSI: novelty detection via contrastive learning on distributionally shifted instances. *Neural Information Processing Systems*. 43

Tan, M. & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*. 26

Techapanurak, E. & Okatani, T. (2019). Hyperparameter-free out-of-distribution detection using softmax of scaled cosine similarity. *arXiv*, abs/1905.10628. 42, 44, 100, 101

Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T., & Michalak, S. (2019). On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Neural Information Processing Systems*. 105

van Amersfoort, J. R., Smith, L., Teh, Y. W., & Gal, Y. (2020). Simple and scalable epistemic uncertainty estimation using a single deep deterministic neural network. *International Conference on Machine Learning*. 42, 44, 100, 101

Vapnik, V. (1991). Principles of risk minimization for learning theory. *Neural Information Processing Systems*. 29

Vareto, R. H., Silva, S., de Oliveira Costa, F., & Schwartz, W. R. (2017). Towards open-set face recognition using hashing functions. *International Joint Conference on Biometrics*. 30, 43, 51

Vasconcelos, G. C., Fairhurst, M. C., & Bisset, D. L. (1995). Investigating feedforward neural networks with respect to the rejection of spurious patterns. *Pattern Recognit. Lett.*, 16(2):207–212. 30

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Neural Information Processing Systems*. 19, 20

Veličković, P., Badia, A. P., Budden, D., Pascanu, R., Banino, A., Dashevskiy, M., Hadsell, R., & Blundell, C. (2022). The CLRS algorithmic reasoning benchmark. *arXiv*, abs/2205.15659. 25, 26

Vyas, A., Jammalamadaka, N., Zhu, X., Das, D., Kaul, B., & Willke, T. L. (2018). Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. *European Conference on Computer Vision*, 11212. 43

Wang, C., Wang, J., & Lin, Q. (2021). Adversarial attacks and defenses in deep learning: A survey. *Intelligent Computing Theories and Application*. 108

Wang, F., Cheng, J., Liu, W., & Liu, H. (2018). Additive margin softmax for face verification. *IEEE Signal Process. Lett.*, 25(7):926–930. 58

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. *arXiv*, abs/2201.11903. 26, 27

Wen, Y., Zhang, K., Li, Z., & Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. *European Conference on Computer Vision*, 9911. 35, 45, 51, 80

Williamson, J. (2005). Objective bayesian nets. *We Will Show Them! Essays in Honour of Dov Gabbay, Volume Two*, 713–730. 56

Williamson, J. (2009). Philosophies of probability. *Handbook of the Philosophy of Mathematics*, 4:493–533. 56

Williamson, J. (2013). In defence of objective bayesianism. *Oxford University Press*, 23(2):255–258. 56

Wong, E., Rice, L., & Kolter, J. Z. (2020). Fast is better than free: Revisiting adversarial training. *International Conference on Learning Representations*. 43

Wright, L. G., Onodera, T., Stein, M. M., Wang, T., Schachter, D. T., Hu, Z., & McMahon, P. L. (2022). Deep physical neural networks trained with backpropagation. *Nature*, 601:549–555. 26

Yang, J., Zhou, K., Li, Y., & Liu, Z. (2021). Generalized out-of-distribution detection: A survey. *arXiv*, abs/2110.11334. 31

Yu, F., Zhang, Y., Song, S., Seff, A., & Xiao, J. (2015). LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv*, abs/1506.03365. 72, 92, 98

Yu, Q. & Aizawa, K. (2019). Unsupervised out-of-distribution detection by maximum classifier discrepancy. *International Conference on Computer Vision*. 43

Yun, S., Han, D., Chun, S., Oh, S. J., Yoo, Y., & Choe, J. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. *International Conference on Computer Vision*. 47, 68, 99, 105

Zadorozhny, K., Thoral, P., Elbers, P. W. G., & Cinà, G. (2021). Out-of-distribution detection for medical applications: Guidelines for practical evaluation. *arXiv*, abs/2109.14885. 31

Zagoruyko, S. & Komodakis, N. (2016). Wide residual networks. *British Machine Vision Conference*. 72, 98

Zaidi, S. S. A., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M. N., & Lee, B. (2021). A survey of modern deep learning based object detection models. *arXiv*, abs/2104.11892. 108

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*. 47

Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560. 70

APPENDIX A - ENTROPIC OUT-OF-DISTRIBUTION DETECTION

Entropic Out-of-Distribution Detection

David Macêdo [1,2], Tsang Ing Ren [1], Cleber Zanchettin [1,3], Adriano L. I. Oliveira [1], and Teresa Ludermir [1]
[1]Centro de Informática, Universidade Federal de Pernambuco, Recife, Brasil
[2]Montreal Institute for Learning Algorithms, University of Montreal, Quebec, Canada
[3]Department of Chemical and Biological Engineering, Northwestern University, Evanston, United States of America
Emails: {dlm, tir, cz, alio, tbl}@cin.ufpe.br

***Abstract*—Out-of-distribution (OOD) detection approaches usually present special requirements (e.g., hyperparameter validation, collection of outlier data) and produce side effects (e.g., classification accuracy drop, slower energy-inefficient inferences). We argue that these issues are a consequence of the SoftMax loss anisotropy and disagreement with the maximum entropy principle. Thus, we propose the IsoMax loss and the entropic score. The seamless drop-in replacement of the SoftMax loss by IsoMax loss requires neither additional data collection nor hyperparameter validation. The trained models do not exhibit classification accuracy drop and produce fast energy-efficient inferences. Moreover, our experiments show that training neural networks with IsoMax loss significantly improves their OOD detection performance. The IsoMax loss exhibits state-of-the-art performance under the mentioned conditions (fast energy-efficient inference, no classification accuracy drop, no collection of outlier data, and no hyperparameter validation), which we call the seamless OOD detection task. In future work, current OOD detection methods may replace the SoftMax loss with the IsoMax loss to improve their performance on the commonly studied non-seamless OOD detection problem.**

I. Introduction

Out-of-distribution (OOD) detection approaches usually use *special requirements* such as input preprocessing [8], [9], feature extraction combined with metric learning [2], adversarial training [10], hyperparameter validation [11], and collection of additional data [12], [13], [14], [15]. Moreover, current OOD methods commonly show *side effects* such as classification accuracy drop [16], [9], and slow energy-inefficient inferences [11], [2]. Solutions based on uncertainty (or confidence) estimation (or calibration) present complexity and lead to slow computationally inefficient inferences [17], [18], [19], [20], [21].

We define the *seamless* OOD detection task, which consists of performing OOD detection under the following restrictions. First, no classification accuracy drop is allowed. Second, the resulting models should produce inferences with the same speed and energy efficiency as those produced by the regularly trained neural networks. Third, no OOD/outlier/additional/extra data may be used. Finally, no hyperparameter validation is required. Improving the performance of neural networks in the *seamless* OOD detection problem is important from a practical perspective. Additionally, such approaches may be combined in future work with current and novel OOD detection techniques to further improve the performance on the *non-seamless* OOD detection task.

We argue that the unsatisfactory OOD detection performance of modern neural networks is mainly due to the drawbacks of the SoftMax loss (we follow the "SoftMax loss" expression as defined in [22]). First, the SoftMax loss anisotropy does not incentivize the concentration of high-level representations in the feature space [1], [10], making OOD detection difficult [10] (Fig. 1a). Second, SoftMax loss produces overconfident low-entropy posterior probability distributions [23], which is in disagreement with the maximum entropy principle [24], [25], [26] (Fig. 1b). Therefore, we propose the isotropy maximization loss (IsoMax loss). To fix the SoftMax loss anisotropy, we made IsoMax an isotropic, i.e., exclusively distance-based, loss. To tackle the SoftMax loss overconfidence, we developed the *entropy maximization trick*, which consists of training with logits multiplied by a high constant that is *removed* for inference. This technique allows IsoMax loss to produce high-entropy (almost maximum) posterior probability distributions in agreement with the principle of maximum entropy.

We propose to train neural networks replacing the SoftMax loss with the IsoMax loss. The swap of the SoftMax loss with the IsoMax loss requires changes in neither the architecture of the model nor training procedures or parameters. For OOD detection, we use the negative entropy of the neural network output probabilities, which we call the entropic score (ES). Since our solution presents neither *special requirements* nor *side effects*, it qualifies as a *seamless* OOD detection approach as previously defined.

Our contributions are the following. First, we associate the unsatisfactory OOD detection performance of neural networks with the SoftMax loss anisotropy and disagreement with the maximum entropy principle. Second, we propose the IsoMax loss that acts as a *SoftMax loss drop-in replacement* and may be used as a *baseline for building improved OOD detection approaches* in future work. We show that the ES produces high performance combined with IsoMax loss. Third, we present the theoretical insight that associates the improved OOD detection performance of the networks trained with IsoMax loss with the principle of maximum entropy. Fourth, we show that our solution produces *state-of-the-art* performance for the *seamless* (fast energy-efficient inferences, no classification accuracy drop, no hyperparameter tuning, and no collection of outlier data) OOD detection task. Fifth, despite being unfair since the approaches present different *special requirements* and *side effects*, we compare our *seamless* solution with *non-seamless* OOD detection methods.

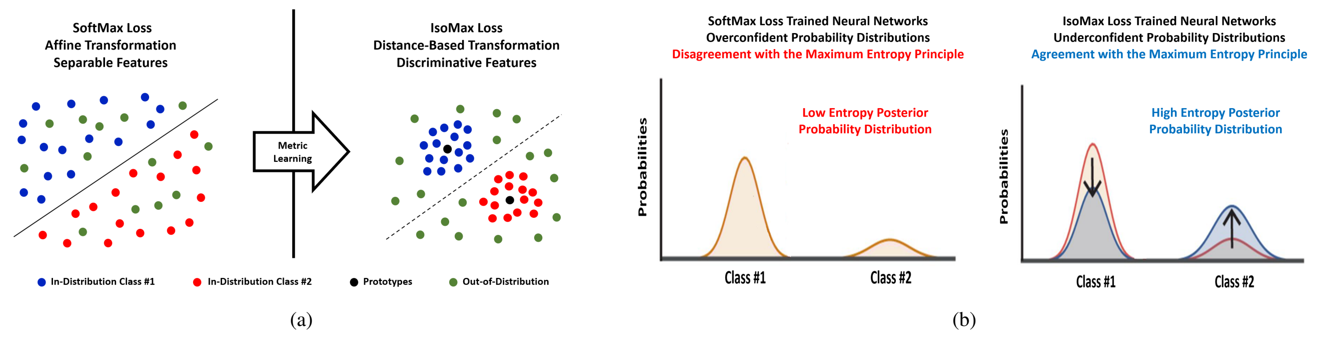

SoftMax Loss
Affine Transformation
Separable Features
Metric Learning
IsoMax Loss
Distance-Based Transformation
Discriminative Features
In-Distribution Class #1
In-Distribution Class #2
Prototypes
Out-of-Distribution
(a)
SoftMax Loss Trained Neural Networks
Overconfident Probability Distributions
Disagreement with the Maximum Entropy Principle
Low Entropy Posterior Probability Distribution
Probabilities
Class #1
Class #2
IsoMax Loss Trained Neural Networks
Underconfident Probability Distributions
Agreement with the Maximum Entropy Principle
High Entropy Posterior Probability Distribution
Probabilities
Class #1
Class #2
(b)

Fig. 1. SoftMax loss drawbacks and IsoMax loss benefits: (a) Adapted from [1, Fig. 1]. SoftMax loss produces separable features [1]. Postprocessing metric learning on features extracted from SoftMax loss-trained networks may convert from the situation on the left to the situation on the right [2], [3], [4], [5], [6], [7]. The IsoMax loss, which is an exclusively distance-based (isotropic) loss, tends to generate more discriminative features [1]. No feature extraction and subsequent metric learning are required when the IsoMax loss is used for training. (b) SoftMax loss trained neural networks produce overconfident low-entropy posterior probability distributions in disagreement with the maximum entropy principle. Our *entropy maximization trick*, which consists in training using logits multiplied by a constant factor called the entropic scale *that is nevertheless removed before inference*, enables IsoMax to generate underconfident high-entropy (almost maximum entropy) posterior probability distributions in agreement with the principle of maximum entropy.

II. Related Works

Liang et al. [11] proposed the Out-of-DIstribution detector for Neural networks (ODIN) by combining input preprocessing and temperature calibration. The authors used OOD examples to validate the hyperparameters. Lee at al. [2] proposed the Mahalanobis distance-based approach by adding feature extraction, feature ensembles, and metric learning to input preprocessing. The authors proposed using adversarial examples rather than the OOD samples to tune hyperparameters. Since this procedure produces more realistic estimations, in this work, we only consider validation on adversarial samples for *non-seamless* OOD detection methods. Hein at al. [10] proposed adversarial confidence enhancing training (ACET), which uses adversarial training.

Hsu et al. [9] proposed to use the *in-distribution* validation set for hyperparameter tuning. The CIFAR10/100 validation sets were used both for hyperparameter validation and for constructing the OOD detection test sets, which may have produced overestimated results. We believe that the in-distribution validation data used for defining hyperparameters should have been removed from the training set, which would presumably lead to an even stronger classification accuracy drop and, consequently, a decrease in OOD detection performance. The solution used input preprocessing and presented classification accuracy drop of a few percentage points in some cases.

Techapanurak et al. [16] used cosine similarity and a learnable block composed of batch normalization, an exponential, and a linear layer. The trained models presented classification accuracy drop of a few percentage points in some cases. Thus, the authors suggested using two models: one for classification and the other for OOD detection.

Recent approaches [12], [13], [14], [15] increased the OOD performance by training/fine-tuning using outlier data and hyperparameter validation. Liu et al. [15] proposed the energy score. Methods based on uncertainty/confidence estimation/calibration [17], [19], [18], [20], [21] have been proposed to tackle the OOD detection problem.

III. Entropic Out-of-Distribution Detection

A. *Isotropy.*

To fix the SoftMax loss anisotropy caused by its affine transformation, we forced the logits of the IsoMax loss to depend exclusively on the distances from the high-level features to the class prototypes.

Let $\boldsymbol{f_\theta}(\boldsymbol{x})$ represent the high-level feature (embedding) associated with $\boldsymbol{x}$, $\boldsymbol{p}^j_{\boldsymbol{\phi}}$ represent the learnable prototype associated with class j, and $d()$ represent the *nonsquared* distance. Additionally, let $\hat{y}^{(k)}$ represent the label of the correct class. Therefore, we construct an isotropic loss by writing:

$$\mathcal{L}_I(\hat{y}^{(k)}|\boldsymbol{x}) = -\log\left(\frac{\exp(-d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}^k_{\boldsymbol{\phi}}))}{\sum\limits_j \exp(-d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}^j_{\boldsymbol{\phi}}))}\right) \quad (1)$$

Unlike metric learning-based OOD detection approaches, rather than learning a metric from a preexisting feature space, in our solution, we learn a feature space that is from the start consistent with the chosen metric, avoiding the need for feature extraction and metric learning postprocessing phases after the neural network training.

B. *Entropy Maximization.*

Isotropy improves the OOD detection performance. However, for further performance gains, we need to circumvent the SoftMax loss propensity to produce low-entropy posterior probability distributions. To achieve high-entropy (almost maximum entropy) distributions in agreement with the maximum entropy principle, we introduce the entropic scale, which consists of a constant scalar factor applied to the logits *throughout the training* that is nevertheless *removed prior to inference*. We call this procedure the *entropy maximization trick*.

The entropic scale is equivalent to the *inverse of the temperature* of the SoftMax *function* (we follow the SoftMax *function* expression as defined in [22]). However, training with a *predefined constant* entropic scale and then *removing it before* *inference* is different from temperature calibration. On the one hand, the temperature of a *pretrained* model is validated *after training* and requires access to the OOD or adversarial examples. Furthermore, overoptimistic performance estimation is commonly produced [27]. On the other hand, our approach requires neither hyperparameter validation nor access to the OOD or adversarial data. Rather than be applied to *pretrained* models, our approach is used to train neural networks.

The presence of the entropic scale during training does not prevent the loss from approaching zero as required. However, when we remove it prior to the inference, the SoftMax *function* naturally makes the entropy of the output probabilities increase to almost the maximum value possible if we use a high enough entropic score during training. Thus, returning to Equation (1), multiplying the embedding-prototype distances by an entropic scale E_s, and representing the 2-norm of a vector by $\|.\|$, we write the definition of the IsoMax loss as:

$$\mathcal{L}_I(\hat{y}^{(k)}|\boldsymbol{x}) = -\log\left(\frac{\exp(-E_s\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^k\|)}{\sum\limits_j \exp(-E_s\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j\|)}\right) \tag{2}$$

By applying the *entropy maximization trick*, the inference probabilities for the IsoMax loss may be written as follows:

$$p_I(y^{(i)}|\boldsymbol{x}) = \frac{\exp(-\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^i\|)}{\sum\limits_j \exp(-\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}_{\boldsymbol{\phi}}^j\|)} \tag{3}$$

C. Prototype Initialization.

We observed that using the Xavier [28] or Kaiming [29] initializations for the prototypes leads to oscillations in performance. *Hence, we decided to initialize all prototypes to the zero vector. Weight decay is applied to the prototypes because they are trainable parameters.*

D. Entropic Score.

The entropy has been studied for OOD detection [30]. We show that the output probabilities negative entropy, which we call the entropic score, produces high-performance results when combined with IsoMax loss. Indeed, in such cases, the solution may consider the information provided by all network outputs rather than merely one output. For instance, ODIN and ACET only use the maximum probability.

E. Implementation Details.

To calculate the losses based on cross-entropy, deep learning libraries usually combine the logarithm and probability into a single computation. *However, we experimentally observed that sequentially computing these calculations as standalone operations improves the IsoMax performance*. The class prototypes have the same dimension as the neural network last-layer representations. The number of prototypes is equal to the number of classes. The IsoMax loss has fewer parameters than the SoftMax loss because it has no bias to learn.

TABLE I
CLASSIFICATION ACCURACY.

Model	Data	Test Accuracy (%) [↑]	
		SoftMax Loss	IsoMax Loss
DenseNet	SVHN	96.6	96.6
	CIFAR10	95.4	95.2
	CIFAR100	77.5	77.5
ResNet	SVHN	96.8	96.8
	CIFAR10	95.5	95.6
	CIFAR100	77.4	77.3

In addition to avoiding classification accuracy drop compared with the SoftMax loss trained networks, IsoMax loss trained models show higher OOD detection performance (see Table II).

We observed classification accuracy drop and low/oscillating performance when trying to integrate the entropic scale into the SoftMax loss or with cosine similarity [22], [31], [32]. In such cases, the above strategy, i.e., initialization of the loss weights with the zero vector, cannot be performed. *The Mahalanobis distance cannot be used because the covariance matrix is not differentiable*. Hence, the *nonsquared Euclidean distance* is the optimal choice for integration with the entropic scale.

IV. EXPERIMENTS

All datasets, models, and evaluation metrics used the baseline established in [35] that was followed in the major OOD detection papers [11], [2], [10]. We trained from the scratch 100-layer DenseNet-BC [36] (growth rate $k = 12$, 0.8M parameters) and 34-layer ResNets [37] on CIFAR10 [38], CIFAR100 [38] and SVHN [39] using the SoftMax and IsoMax losses.

For both the SoftMax loss and IsoMax loss, we used SGD with the Nesterov moment equal to 0.9, 300 epochs with a batch size of 64, and an initial learning rate of 0.1, with a learning rate decay rate equal to ten applied in epochs 150, 200, and 250. We used a dropout of zero. The weight decay was 0.0001.

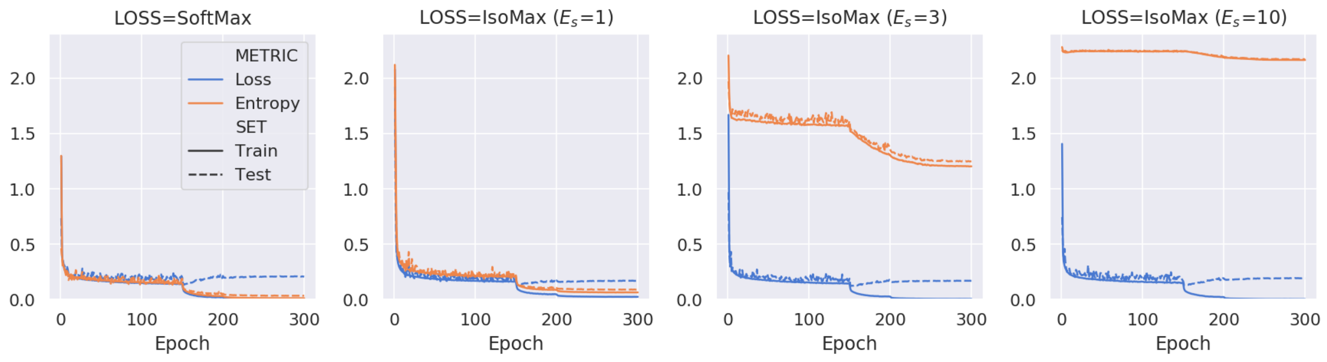
We only compared approaches that did not present classification accuracy drop because this facilitates increasing OOD detection performance [40]; moreover, it is particularly undesired from a practical perspective [41]. It is well known that using OOD/outlier/background/additional data improves the OOD detection performance. Therefore, considering that data-based regularization techniques may benefit both SoftMax loss and IsoMax loss, we perform all experiments without outlier exposure [13], [14], background samples [12], or energy-based fine-tuning [15]. The source code is available online[1].

V. RESULTS AND DISCUSSIONS

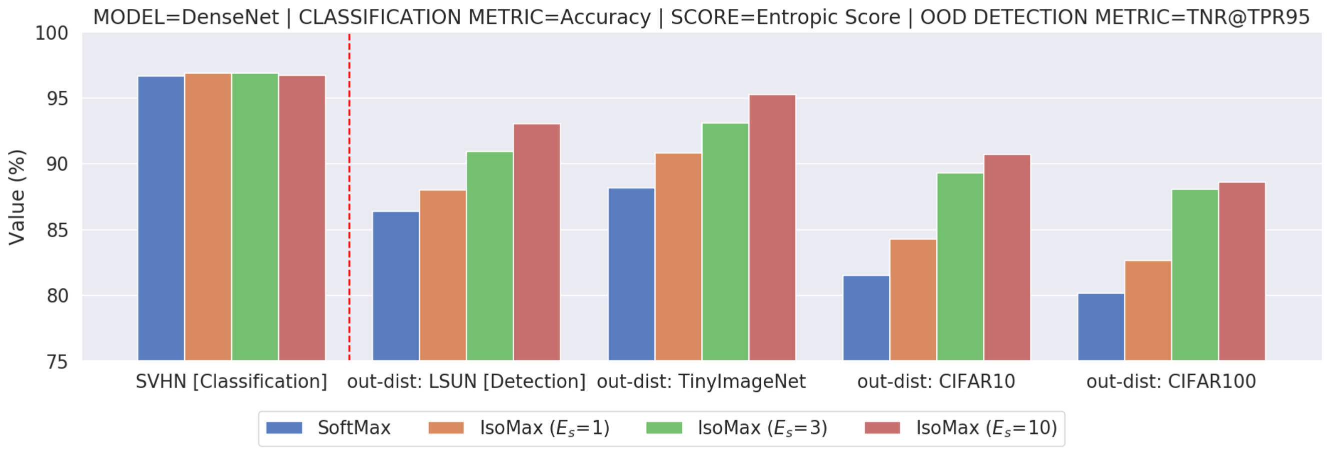
A. IsoMax Loss Properties, Ablation Study, and Entropic Scale Value Definition.

To experimentally show that higher entropic scales lead to higher mean entropy probability distributions and consequently

[1]https://github.com/dlmacedo/entropic-out-of-distribution-detection



(a) (b) (c) (d)



(e)

Fig. 2. (a) SoftMax loss minimizes both the cross-entropy and the mean entropy of the posterior probabilities. (b) IsoMax loss produces low mean entropy posterior probabilities for a low entropic scale (E_s=1). (c) IsoMax loss produces medium mean entropy for an intermediate entropic scale (E_s=3). (d) IsoMax loss produces high mean entropy for a high entropic scale (E_s=10). Therefore, higher entropic scale values are correlated with higher mean entropies as recommended by the maximum entropy principle. Notice that the orange line is almost flat in (d), so the IsoMax loss almost retains the maximum entropy present at the beginning of the training for a high entropic scale. Hence, an entropic scale equal to ten is enough to produce posterior probability distributions with virtually the maximum possible mean entropy $\log(N)$, where N is the number of classes. Consequently, there is no need to increase E_s further. Therefore, we decided to use E_s=10 for IsoMax loss (see also Fig. 3). (e) The left side of the dashed vertical red line presents the classification accuracies. The right side of the dashed vertical red line shows the OOD detection performance using the entropic score and the TNR@TPR95 (true negative rate at 95% true positive rate) metric. We observe that a higher mean entropy produces increased OOD detection performance regardless of the out-of-distribution (out-dist). Isotropy by itself enables the IsoMax loss to exhibit higher performance than the SoftMax loss (E_s=1). IsoMax loss trained models exhibit classification accuracies similar to the classification accuracies presented by SoftMax loss trained networks regardless of the entropic scale.

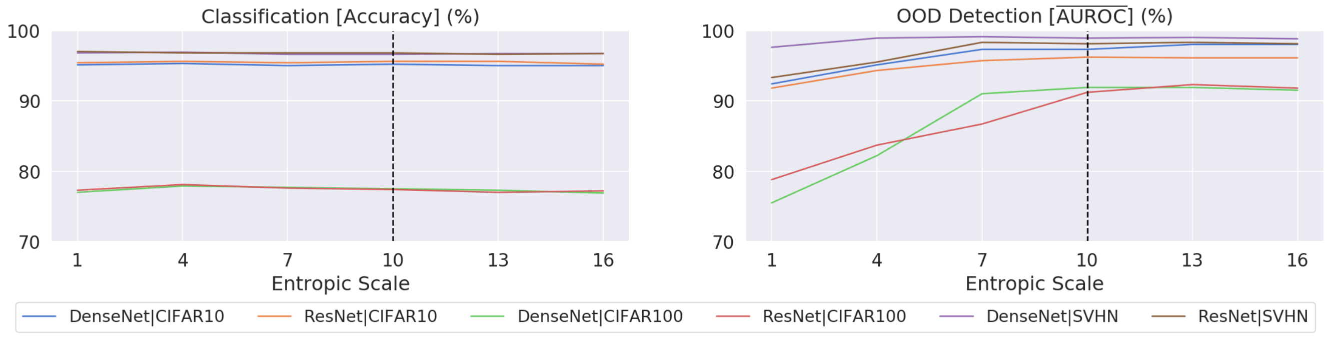


Fig. 3. $\overline{\text{AUROC}}$ represents the mean AUROC considering all out-of-distribution data. The classification accuracy and the mean OOD detection performance are approximately stable for E_s=10 or higher regardless of the dataset and model. E_s validation cannot significantly improve the OOD detection performance. In fact, this is not even possible because access to the OOD or outlier samples is not allowed in *seamless* OOD detection. Making E_s learnable did not considerably improve or decrease the OOD detection results.

TABLE II
SEAMLESS OUT-OF-DISTRIBUTION DETECTION: NO HYPERPARAMETER TUNING. FAST AND ENERGY-EFFICIENT INFERENCES. NO CLASSIFICATION ACCURACY DROP. NO OUTLIER/BACKGROUND DATA.

Model	Data (training)	OOD (unseen)	Seamless OOD Detection: No Classification Accuracy Drop. No Outlier Data. Fast and Energy-Efficient Inferences. No Hyperparameter Tuning.		
			TNR@TPR95[1] (%) [↑]	AUROC[2] (%) [↑]	DTACC[3] (%) [↑]
			SoftMax+MPS[4] / SoftMax+ES[5] / IsoMax+MPS[6] / IsoMax+ES[7] (ours)		
DenseNet	CIFAR10	SVHN	32.2 / 33.2 / 64.5 / **77.0**	86.6 / 86.9 / 94.6 / **96.6**	79.9 / 79.9 / 88.1 / **91.6**
		TinyImageNet [33]	55.8 / 59.8 / 81.1 / **88.0**	93.5 / 94.2 / 96.8 / **97.8**	87.6 / 87.8 / 90.8 / **93.2**
		LSUN [34]	64.9 / 69.5 / 88.5 / **94.5**	95.2 / 95.9 / 97.9 / **98.8**	89.9 / 90.0 / 93.1 / **94.9**
	CIFAR100	SVHN	20.6 / 24.9 / **27.5** / 23.4	80.1 / 81.9 / 86.3 / **88.6**	73.9 / 74.3 / 79.9 / **83.7**
		TinyImageNet	19.4 / 23.7 / 42.4 / **49.1**	77.0 / 78.8 / 90.2 / **92.6**	70.6 / 71.1 / 83.6 / **86.6**
		LSUN	18.8 / 24.4 / 48.9 / **63.0**	75.9 / 77.9 / 91.3 / **94.7**	69.5 / 70.2 / 84.2 / **89.1**
	SVHN	CIFAR10	81.5 / 83.7 / 91.6 / **94.1**	96.5 / 96.9 / 98.2 / **98.5**	91.9 / 92.1 / 94.1 / **95.0**
		TinyImageNet	88.2 / 90.0 / 95.3 / **97.0**	97.7 / 98.1 / 98.9 / **99.1**	93.5 / 93.7 / 95.4 / **96.1**
		LSUN	86.4 / 88.4 / 94.7 / **96.8**	97.3 / 97.8 / 98.7 / **99.1**	92.8 / 93.0 / 95.0 / **95.9**
ResNet	CIFAR10	SVHN	43.1 / 44.5 / 81.7 / **83.6**	91.7 / 92.0 / 96.8 / **97.1**	86.5 / 86.5 / 91.2 / **91.9**
		TinyImageNet	46.3 / 48.0 / 66.0 / **70.2**	89.8 / 90.0 / 93.9 / **94.6**	84.0 / 84.1 / 87.1 / **88.3**
		LSUN	51.2 / 53.3 / 76.6 / **82.3**	92.2 / 92.6 / 96.2 / **96.9**	86.5 / 86.6 / 90.1 / **91.5**
	CIFAR100	SVHN	15.9 / 18.0 / **22.5** / 20.2	71.3 / 72.7 / 83.9 / **85.3**	66.1 / 66.3 / 77.8 / **79.7**
		TinyImageNet	18.5 / 22.4 / 38.9 / **50.6**	74.7 / 76.3 / 89.2 / **92.0**	68.8 / 69.1 / 82.2 / **85.6**
		LSUN	18.4 / 22.4 / 41.4 / **54.8**	74.7 / 76.5 / 90.1 / **93.2**	69.1 / 69.4 / 83.3 / **87.5**
	SVHN	CIFAR10	67.3 / 67.7 / 90.5 / **92.3**	89.8 / 89.7 / 97.9 / **98.0**	87.0 / 86.9 / 93.7 / **94.1**
		TinyImageNet	66.9 / 67.3 / 92.1 / **94.4**	89.0 / 89.0 / 98.2 / **98.4**	86.7 / 86.6 / 94.3 / **94.8**
		LSUN	62.2 / 62.5 / 88.6 / **90.8**	86.0 / 85.8 / 97.6 / **97.8**	84.2 / 84.1 / 93.4 / **93.6**

[1]True negative rate at 95% true positive rate. [2]Area under the receiver operating characteristic curve. [3]Detection accuracy [2]. [4]SoftMax+MPS means training with SoftMax loss and performing OOD detection using the maximum probability score (MPS), which is the approach defined in [35]. [5]SoftMax+ES means training with SoftMax loss and performing OOD detection using the entropic score (ES). [6]IsoMax+MPS means training with IsoMax loss and performing OOD detection using the maximum probability score (MPS). [7]IsoMax+ES means training with IsoMax loss and performing OOD detection using the entropic score (ES) (our proposal). The best results are shown in bold. To the best of our knowledge, our IsoMax+ES approach presents *state-of-the-art* performance under these severely restrictive assumptions.

improve the OOD detection performance, we trained DenseNets on SVHN using the SoftMax loss and IsoMax loss with distinct entropic scale values. We used the entropic score and the TNR@TPR95 (true negative rate at 95% true positive rate) to evaluate the OOD detection performance (Fig. 2).

Fig. 2a shows that the SoftMax loss generates posterior distributions with low mean entropy. Fig. 2b illustrates that the unitary entropic scale (E_s= 1) does not increase the mean entropy of the probability distributions. In other words, isotropy alone is not enough to produce low mean entropy probability distributions, and the *entropy maximization trick* is necessary. Nevertheless, Fig. 2e shows that the simple replacement of anisotropic logits based on the affine transformation by isotropic logits is enough to produce some OOD detection performance gains for all out-of-distribution (out-dist) data, even without the mentioned trick (E_s= 1). Fig. 2c shows that an intermediate entropic scale (E_s= 3) provides medium mean entropy probability distributions with the additional OOD detection performance gains regardless of the out-of-distribution data (Fig. 2e). Fig. 2d illustrates that a high entropic scale (E_s= 10) produces even higher mean entropy probability distributions and the highest OOD detection performance for all out-of-distribution data considered (Fig. 2e). We emphasize that regardless of training with the entropic scale, *if it is not removed for inference*, the IsoMax loss produces outputs with entropies as low as those produced by the SoftMax loss and high OOD detection performances are no long observed.

Hence, the *entropy maximization trick* enables the migration from low-entropy distributions (Fig. 2a,b) to high-entropy distributions (Fig. 2d). For a high entropic scale, the IsoMax loss minimizes the cross-entropy while producing high-entropy probability distributions as recommended by the principle of maximum entropy. More importantly, higher entropy posterior probability distributions directly correlate with increased OOD detection performances despite the out-of-distribution data. Fig. 2d shows that an entropic scale E_s= 10 is enough to produce essentially the maximum possible entropy. Therefore, we defined the entropic scale as a *constant* equal to ten.

B. Classification Accuracy and OOD Detection Performance Dependence on the Entropic Scale Value.

After defining E_s= 10 for the IsoMax loss based on the previous experiments, we performed additional analyses. Fig. 3 shows that *regardless of the combination of dataset and model*, the classification accuracy and the mean OOD detection performance are essentially stable for E_s= 10 or higher, as the entropic scale is already high enough to ensure near-maximal entropy. Hence, validation of E_s does not produce a considerable performance increase. In fact, this is not even possible because we consider access to OOD or outlier samples to be forbidden. Making E_s learnable did not significantly affect

TABLE III
NON-SEAMLESS OUT-OF-DISTRIBUTION DETECTION: UNFAIR COMPARISON OF APPROACHES WITH DIFFERENT SPECIAL REQUIREMENTS AND SIDE EFFECTS. NO CLASSIFICATION ACCURACY DROP. NO OUTLIER/BACKGROUND DATA.

Model	Data (training)	OOD (unseen)	Non-seamless Out-of-Distribution Detection: Approaches with Different Special Requirements and Side Effects.	
			AUROC (%) [↑]	DTACC[5] (%) [↑]
			ODIN[1] / ACET[2] / IsoMax+ES[3] (ours) / Mahalanobis[4]	ODIN[1] / ACET[2] / IsoMax+ES[3] (ours) / Mahalanobis[4]
DenseNet	CIFAR10	SVHN	92.8 / NA / **96.6 / 97.6**	86.5 / NA / **91.6 / 92.6**
		TinyImageNet	97.2 / NA / **97.8 / 98.8**	92.1 / NA / **93.2 / 95.0**
		LSUN	98.5 / NA / **98.8 / 99.2**	94.3 / NA / **94.9 / 96.2**
	CIFAR100	SVHN	88.2 / NA / 88.6 / **91.8**	80.7 / NA / **83.7 / 84.6**
		TinyImageNet	85.3 / NA / 92.6 / **97.0**	77.2 / NA / 86.6 / **91.8**
		LSUN	85.7 / NA / 94.7 / **97.9**	77.3 / NA / 89.1 / **93.8**
	SVHN	CIFAR10	91.9 / NA / **98.5 / 98.8**	86.6 / NA / **95.0 / 96.3**
		TinyImageNet	94.8 / NA / **99.1 / 99.8**	90.2 / NA / 96.1 / **98.9**
		LSUN	94.1 / NA / **99.1 / 99.9**	89.1 / NA / 95.9 / **99.2**
ResNet	CIFAR10	SVHN	86.5 / **98.1 / 97.1** / 95.5	77.8 / NA / **91.9 / 89.1**
		TinyImageNet	93.9 / 85.9 / 94.6 / **99.0**	86.0 / NA / 88.3 / **95.4**
		LSUN	93.7 / 85.8 / 96.9 / **99.5**	85.8 / NA / 91.5 / **97.2**
	CIFAR100	SVHN	72.0 / **91.2** / 85.3 / 84.4	67.7 / NA / **79.7** / 76.5
		TinyImageNet	83.6 / 75.2 / **92.0** / 87.9	75.9 / NA / **85.6 / 84.6**
		LSUN	81.9 / 69.8 / **93.2** / 82.3	74.6 / NA / **87.5** / 79.7
	SVHN	CIFAR10	92.1 / 97.3 / **98.0 / 97.6**	89.4 / NA / **94.1 / 94.6**
		TinyImageNet	92.9 / 97.7 / **98.4 / 99.3**	90.1 / NA / 94.8 / **98.8**
		LSUN	90.7 / **99.7** / 97.8 / **99.9**	88.2 / NA / 93.6 / **99.5**

ODIN, the Mahalanobis approach, and ACET present hyperparameters that must be validated for each combination of datasets and models. They also require previously known optimal adversarial perturbation values for each combination of datasets and models. [1]ODIN uses input preprocessing, temperature calibration, and adversarial validation, i.e., hyperparameter tuning using adversarial examples [11]. [2]ACET uses adversarial training, resulting in slower training, possibly reduced scalability for large images, and eventually classification accuracy drop [10]. [3]IsoMax+ES means training with IsoMax loss and performing OOD detection using the entropic score (ES). Considering that validating E_s using adversarial examples cannot produce significant gains (Fig. 3), we prefer to keep $E_s = 10$ to maintain the simplicity of the solution. [4]The Mahalanobis solution uses input preprocessing, feature ensemble, feature extraction followed by metric learning, and adversarial validation [2]. [5]Detection accuracy [2]. The best results are shown in bold (2% tolerance).

TABLE IV
NON-SEAMLESS OOD DETECTION: INFERENCE DELAYS. PRESUMED COMPUTATIONAL COST AND ENERGY CONSUMPTION RATES.

Model	Data (training)	Hardware (inference)	Non-seamless Out-of-Distribution Detection: Inference Delays. Presumed Computational Cost and Energy Consumption Rates.		
			SoftMax Loss [35] MPS / ES (ms) [↓]	IsoMax Loss (ours) MPS / ES (ms) [↓]	ODIN [11], Mahalanobis [2], Generalized ODIN [9] (ms) [↓]
DenseNet	CIFAR10	CPU	18.1 / 19.4	18.0 / 19.2	242.4 (≈ **10x slower)**
		GPU	11.6 / 13.0	11.6 / 11.5	39.2 (≈ **4x slower)**
	CIFAR100	CPU	18.4 / 19.8	18.4 / 19.3	261.0 (≈ **10x slower)**
		GPU	12.9 / 11.4	11.8 / 11.5	39.6 (≈ **4x slower)**
	SVHN	CPU	18.1 / 18.6	18.3 / 18.6	241.5 (≈ **10x slower)**
		GPU	11.6 / 11.9	11.7 / 11.6	39.6 (≈ **4x slower)**
ResNet	CIFAR10	CPU	22.3 / 23.2	23.0 / 23.5	250.4 (≈ **10x slower)**
		GPU	4.5 / 3.8	4.2 / 4.1	15.4 (≈ **4x slower)**
	CIFAR100	CPU	23.3 / 23.1	23.3 / 23.8	252.6 (≈ **10x slower)**
		GPU	4.3 / 3.9	4.3 / 4.2	14.8 (≈ **4x slower)**
	SVHN	CPU	23.1 / 23.4	23.4 / 23.3	263.8 (≈ **10x slower)**
		GPU	4.2 / 4.0	4.0 / 4.0	15.7 (≈ **4x slower)**

MPS means maximum probability score. ES means entropic score. For SoftMax loss and IsoMax loss, the inference delays combine both classification and detection computation. For the methods based on input preprocessing, the inference delays represent only the input preprocessing phase. All values are in milliseconds. The inference delay rates presumably reflect similar computational cost and energy consumption rates.

the OOD detection performance. Table I shows that the IsoMax loss trained models do not show classification accuracy drop.

C. Seamless Out-of-Distribution Detection.

To the best of our knowledge, the proposal presented in [35] and our method are the only solutions that qualify as *seamless* OOD detection approaches. Table II shows that the models trained with the SoftMax loss using the maximum probability as the score (SoftMax+MPS) always present the worst performance results and that replacing the maximum probability score by the entropic score (SoftMax+ES) produces OOD detection performance gains.

The combination of the models trained using IsoMax loss with the entropic score (IsoMax+ES), which is the proposed solution, significantly improves, usually by several percentage points, the OOD detection performance across almost all datasets, models, out-of-distribution data, and metrics.

The entropic score produces high OOD detection performance when the distributions present high entropy (IsoMax+ES). Indeed, both *producing high-entropy distributions* and the *entropic score* contribute to improving the OOD detection performance. However, the contribution of *producing high-entropy distributions is considerably more important.*

The model does not affect the analyses presented. Indeed, the comments shown above are valid for both DenseNet and ResNet models.

D. Non-seamless Out-of-Distribution Detection.

To tackle *non-seamless* OOD detection, IsoMax should work as a *baseline* to be combined with OOD techniques (e.g., outlier exposure, adversarial training, input preprocessing, energy score) rather than competing as a standalone solution. Nevertheless, Table III provides a perspective for how our *baseline seamless (standalone)* approach compares to *non-seamless (composed)* solutions.

From a qualitative perspective, ODIN and Mahalanobis use input preprocessing; i.e., to perform OOD detection, each inference requires a first neural network forward pass, a backpropagation, and a second forward pass. They produce slower and less energy-efficient inferences than models trained with IsoMax loss, which are as fast and computationally efficient as the models trained with SoftMax loss. Input preprocessing is indeed a limitation from an economic and environmental perspective [42].

ODIN requires temperature calibration after neural network training, while the Mahalanobis approach requires feature ensemble and metric learning. Unlike pretraining-based solutions, our approach requires no postprocessing after the neural network training. ACET requires adversarial training, which produces slower training and may limit the application of ACET to large images [43].

From a quantitative point of view, Table III shows that IsoMax+ES considerably outperforms ODIN in all evaluated scenarios. Therefore, in addition to avoiding hyperparameter tuning and access to the OOD or adversarial samples, the results show that the *entropy maximization trick* is much more effective in improving the OOD detection performance than temperature calibration, even when the latter is combined with input preprocessing. Furthermore, IsoMax+ES usually outperforms ACET, in some cases by a large margin. Moreover, in most cases, the Mahalanobis method surpasses IsoMax+ES by less than 2%. In some scenarios, IsoMax+ES outperforms the Mahalanobis method.

Table IV presents the inference delays for the SoftMax loss, IsoMax loss, and competing methods using a CPU and GPU. We observe that neural networks trained using the IsoMax loss produce inferences equally as fast as those produced by networks trained using the SoftMax loss, regardless of whether a CPU or GPU is used for inference.

Additionally, the entropic score is as fast as the usual maximum probability score. Moreover, the methods based on input preprocessing were more than ten times slower on the CPU and approximately four times slower on the GPU. These ratios also presumably apply to the computational cost and energy consumption.

To agree with the maximum entropy principle and achieve high performance, rather than generating *calibrated* maximum probabilities, IsoMax must produce the *lowest possible* maximum probabilities.

VI. Conclusion

We proposed a *seamless* OOD detection approach based on logit isotropy and the maximum entropy principle. The proposed IsoMax loss acts as a SoftMax loss drop-in replacement that produces accurate predictions in addition to fast energy- and computation-efficient inferences. No hyperparameter tuning is needed. Hence, no additional procedure other than straightforward neural network training is needed.

OOD detection is performed using the rapid entropic score. Collection of outlier/background data is also not required. To the best of our knowledge, the IsoMax loss does not present any drawbacks compared to the SoftMax loss.

The direct replacement of the SoftMax loss by the IsoMax loss significantly improves the baseline OOD detection performance of neural networks. Therefore, rather than the limitations of the models, the low OOD detection performance of deep networks is due to the SoftMax loss drawbacks, i.e., anisotropy and overconfidence.

In future work, the research community may combine the IsoMax loss with *data-based loss regularization techniques* [12], [13], [14] to improve the performance. *Approaches based on pretrained models* [11], [2], [44] or energy-based fine-tuning/score [15] may be applied on IsoMax loss pretrained networks rather than on SoftMax pretrained models.

Thus, rather than competitors, these approaches are actually *complementary* to IsoMax loss, as they may be *combined* to achieve even higher overall OOD detection performance. IsoMax loss may replace SoftMax loss as a *higher performance baseline for constructing OOD detection solutions.*

Another option is to use recent data augmentation techniques [45], [46]. We believe that the simplicity of our solution makes it scalable to large images. Hence, we intend to apply this approach to ImageNet [33]. Finally, since our approach consists of only loss replacement and is based on the general principles of isotropy and maximum entropy, it may be extended to other machine learning methods beyond neural networks.

References

[1] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," *European Conference on Computer Vision*, 2016.

[2] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," *Neural Information Processing Systems*, 2018.

[3] T. Mensink, J. J. Verbeek, F. Perronnin, and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2624–2637, 2013.

[4] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult, "Towards open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757–1772, 2013.

[5] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2317–2324, 2014.

[6] A. Bendale and T. Boult, "Towards open world recognition," *IEEE International Conference on Computer Vision and Pattern Recognition*, 2015.

[7] E. Rudd, L. P. Jain, W. J. Scheirer, and T. Boult, "The extreme value machine," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 762–768, 2018.

[8] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *CoRR*, vol. abs/1802.04865, 2018.

[9] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized ODIN: Detecting out-of-distribution image without learning from out-of-distribution data," *IEEE International Conference on Computer Vision and Pattern Recognition*, 2020.

[10] M. Hein, M. Andriushchenko, and J. Bitterwolf, "Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem," *IEEE International Conference on Computer Vision and Pattern Recognition*, 2018.

[11] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *International Conference on Learning Representations*, 2018.

[12] A. R. Dhamija, M. Günther, and T. Boult, "Reducing network agnostophobia," *Neural Information Processing Systems*, 2018.

[13] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," *International Conference on Learning Representations*, 2019.

[14] A.-A. Papadopoulos, M. R. Rajati, N. Shaikh, and J. Wang, "Outlier exposure with confidence control for out-of-distribution detection," *CoRR*, vol. abs/1906.03509, 2019.

[15] W. Liu, X. Wang, J. D. Owens, and Y. Li, "Energy-based out-of-distribution detection," *CoRR*, vol. abs/2010.03759, 2020.

[16] E. Techapanurak, M. Suganuma, and T. Okatani, "Hyperparameter-free out-of-distribution detection using cosine similarity," *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020.

[17] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," *Neural Information Processing Systems*, 2017.

[18] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl, "Leveraging uncertainty information from deep neural networks for disease detection," *Scientific Reports*, vol. 7, 2017.

[19] A. Subramanya, S. Srinivas, and R. V. Babu, "Confidence estimation in deep neural networks via density modelling," *CoRR*, vol. abs/1707.07013, 2017.

[20] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," *Neural Information Processing Systems*, 2018.

[21] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," *International Conference on Machine Learning*, 2018.

[22] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks.," *International Conference on Machine Learning*, 2016.

[23] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *International Conference on Machine Learning*, 2017.

[24] E. T. Jaynes, "Information theory and statistical mechanics," *Physical Review*, vol. 106, pp. 620–630, 1957.

[25] E. T. Jaynes, "Information theory and statistical mechanics. II," *Physical Review*, vol. 108, pp. 171–190, 1957.

[26] T. M. Cover and J. A. Thomas, "Elements of information theory," *Wiley Series in Telecommunications and Signal Processing*, 2006.

[27] A. Shafaei, M. Schmidt, and J. J. Little, "A less biased evaluation of out-of-distribution sample detectors," *British Machine Vision Conference*, 2019.

[28] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *International Conference on Artificial Intelligence and Statistics*, 2010.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *International Conference on Computer Vision*, 2016.

[30] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. A. DePristo, J. V. Dillon, and B. Lakshminarayanan, "Likelihood ratios for out-of-distribution detection," *Neural Information Processing Systems*, 2019.

[31] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[32] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," *IEEE International Conference on Computer Vision and Pattern Recognition*, 2019.

[33] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "ImageNet: A large-scale hierarchical image database," *IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.

[34] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "LSUN: construction of a large-scale image dataset using deep learning with humans in the loop," *CoRR*, vol. abs/1506.03365, 2015.

[35] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *International Conference on Learning Representations*, 2017.

[36] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *IEEE International Conference on Computer Vision and Pattern Recognition*, 2017.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *European Conference on Computer Vision*, 2016.

[38] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Science Department, University of Toronto*, 2009.

[39] Y. Netzer and T. Wang, "Reading digits in natural images with unsupervised feature learning," *Neural Information Processing Systems*, 2011.

[40] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," *International Conference on Learning Representations*, 2019.

[41] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. J. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *CoRR*, vol. abs/1902.06705, 2019.

[42] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[43] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. P. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!," *Neural Information Processing Systems*, 2019.

[44] C. S. Sastry and S. Oore, "Detecting out-of-distribution examples with gram matrices," *International Conference on Machine Learning*, vol. 119, pp. 8491–8501, 2020.

[45] S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," *Neural Information Processing Systems*, 2019.

[46] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," *International Conference on Computer Vision*, 2019.

APPENDIX B - ENTROPIC OUT-OF-DISTRIBUTION DETECTION: SEAMLESS DETECTION OF UNKNOWN EXAMPLES

Entropic Out-of-Distribution Detection: Seamless Detection of Unknown Examples

David Macêdo, *Member, IEEE,* Tsang Ing Ren, *Member, IEEE,* Cleber Zanchettin, *Member, IEEE,* Adriano L. I. Oliveira, *Senior Member, IEEE,* and Teresa Ludermir, *Senior Member, IEEE*

***Abstract*—In this paper, we argue that the unsatisfactory out-of-distribution (OOD) detection performance of neural networks is mainly due to the SoftMax loss anisotropy and propensity to produce low entropy probability distributions in disagreement with the principle of maximum entropy. Current out-of-distribution (OOD) detection approaches usually do not directly fix the SoftMax loss drawbacks, but rather build techniques to circumvent it. Unfortunately, those methods usually produce undesired side effects (e.g., classification accuracy drop, additional hyperparameters, slower inferences, and collecting extra data). In the opposite direction, we propose replacing SoftMax loss with a novel loss function that does not suffer from the mentioned weaknesses. The proposed IsoMax loss is isotropic (exclusively distance-based) and provides high entropy posterior probability distributions. Replacing the SoftMax loss by IsoMax loss requires no model or training changes. Additionally, the models trained with IsoMax loss produce as fast and energy-efficient inferences as those trained using SoftMax loss. Moreover, no classification accuracy drop is observed. The proposed method does not rely on outlier/background data, hyperparameter tuning, temperature calibration, feature extraction, metric learning, adversarial training, ensemble procedures, or generative models. Our experiments showed that IsoMax loss works as a seamless SoftMax loss drop-in replacement that significantly improves neural networks' OOD detection performance. Hence, it may be used as a baseline OOD detection approach to be combined with current or future OOD detection techniques to achieve even higher results.**

***Index Terms*—Out-of-Distribution Detection, Isotropy Maximization Loss, Maximum Entropy Principle, Entropic Score**

I. Introduction

NEURAL networks have been used as classifiers in a wide range of applications. Their design usually considers that the model receives an instance of one of the trained classes at inference. If this holds, neural networks commonly present satisfactory performance. However, in real-world applications, this assumption may not be fulfilled.

The ability to detect whether an input applied to a neural network does not represent an example of the trained classes is essential to applications in medicine, finance, agriculture, business, marketing, and engineering. In such situations, it is better to have a system able to acknowledge that the sample should not be classified. The rapid adoption of neural networks in modern applications makes the development of such capability a primary necessity from a practical point of view. This problem has been studied under many similar nomenclatures, such as open set recognition [4], [5] and open-world recognition [6], [7].

David Macêdo, Tsang Ing Ren, Cleber Zanchettin, Adriano L. I. Oliveira, and Teresa Ludermir are with the Centro de Informática, Universidade Federal de Pernambuco, Brazil. E-mail: dlm@cin.ufpe.br.

David Macêdo was with Montreal Institute for Learning Algorithms (MILA), Université de Montréal (UdeM), Quebec, Canada.

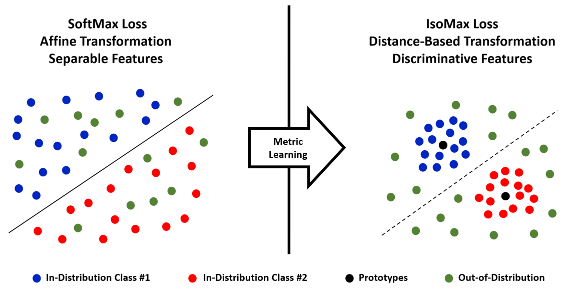


Fig. 1. Adapted from [10, Fig. 1]. SoftMax loss produces separable features. Metric learning on features extracted from SoftMax loss trained networks may convert from the situation on the left to the situation on the right. Exclusively distance-based (isotropic) losses tend to generate more discriminative features. Out-of-distribution examples are more discernible when in-distribution examples are concentrated around prototypes.

Recently, Hendrycks and Gimpel [8] defined out-of-distribution (OOD) detection as the task of evaluating whether a sample comes from the in-distribution on which a neural network was trained. They also introduced benchmark datasets and metrics for this task. Additionally, Hendrycks and Gimpel established the baseline performance by proposing an OOD detection approach that uses the maximum predicted probability (MPS) as the score to detect OOD examples.

OOD detection is closely related to anomaly and novelty detection [9]. However, in OOD detection, we have multiple (usually many more than two) classes. From an anomaly detection perspective, examples that belong to any of these classes are considered "normal". Additionally, we have labels that individually identify examples from each of these "normal" classes, which collectively represent what we call the in-distribution. There are no training examples of the "abnormal" class, which are called out-of-distribution examples in the context of OOD detection. We have to decide whether we have an in-distribution ("normal") example or an out-of-distribution ("abnormal") example during inference. In the first case, we additionally have to predict the correct class.

We argue that the unsatisfactory OOD detection performance of modern neural networks is mainly due to the drawbacks of the currently used loss functions rather than model limitations. The SoftMax loss[1] anisotropy does not incentivize the concentration of high-level representations in the feature space [3], [10], which makes OOD detection

[1]We follow the "SoftMax loss" expression as defined in [11].

TABLE I
OUT-OF-DISTRIBUTION DETECTION: APPROACHES, TECHNIQUES, AND SIDE EFFECTS.

Techniques and Side Effects	SoftMax Loss (current baseline)	ODIN [1]	Mahalanobis [2]	ACET [3]	IsoMax Loss (proposed baseline)
Inference input preprocessing: Multiple forward passes and backpropagation. At least three times slower inference. **Slow inferences.**	Not Required (Fast)	Required (Slow)	Required (Slow)	Not Required (Fast)	Not Required (Fast)
Inference input preprocessing: At least three times higher energy consumption. At least three times higher computational cost. **Energy-inefficient inferences.**	Not Required (Efficient)	Required (Not Efficient)	Required (Not Efficient)	Not Required (Efficient)	Not Required (Efficient)
Adversarial training: Previously known adversarial hyperparameters. Slower and more hyperparametrized training. **Limited scalability.**	Not Required (Scalable)	Not Required (Scalable)	Not Required (Scalable)	Required (Not Scalable)	Not Required (Scalable)
Adversarial validation: Previously known adversarial hyperparameters. Adversarial examples generation. **Hyperparameters tuning.**	Not Required (Turnkey)	Required (Not Turnkey)	Required (Not Turnkey)	Not Required (Turnkey)	Not Required (Turnkey)

IsoMax loss has no drawbacks compared with SoftMax loss while presenting higher OOD detection performance (Table II). ODIN, the Mahalanobis method, and ACET present weaknesses from a practical use perspective. However, if these limitations are not a concern for an application, they may be combined with IsoMax loss to improve overall OOD detection performance. The strengths of the competing approaches are in blue, while weaknesses are in red.

difficult [3]. Indeed, its affine transformation produces separable rather than discriminative features, which would limit OOD detection (Fig. 1). Additionally, SoftMax loss produces overconfident predictions [12], equivalent to low mean entropy probability distributions, which is in frontal disagreement with the maximum entropy principle [13]–[15].

Rather than directly fixing the above-mentioned critical SoftMax loss drawbacks, current OOD detection approaches improve OOD detection performance by adding novel techniques to circumvent its weakness. However, this strategy usually produces undesired side effects and adds problematic requirements to the solution (Table I). For example, Out-of-DIstribution detector for Neural networks (ODIN) [1] and the Mahalanobis distance-based method [2] use *input preprocessing*, which produces remarkably slow and energy-inefficient inferences [16]. Additionally, to tune hyperparameters, these solutions need unrealistic access to OOD samples or adversarial examples, which require a cumbersome process to generate.

In other situations, similar to the Mahalanobis distance-based approach, [4]–[7] require metric learning on features extracted from pretrained models. Another drawback usually present in OOD detection approaches is the *classification accuracy drop* [17], [18], which is a harmful side effect because classification is commonly the primary aim of the system [19]. Methods based on adversarial training, such as Adversarial Confidence Enhancing Training (ACET) [3], often increase training times [20] and present limited scalability when dealing with large-size images [21]. Furthermore, adversarial training may produce classification accuracy drop [22].

In some cases, OOD detection proposals require architecture modifications [23] or ensemble methods [24], [25]. Despite significantly improving the OOD detection performance, loss enhancement (regularization) techniques, such as outlier exposure [26], [27], background methods [28], and the energy-based fine-tuning [29] require the addition of carefully chosen extra/outlier/background data and expand memory usage. Moreover, they usually add hyperparameters to the solution.

Solutions based on uncertainty (or confidence) estimation (or calibration) [30]–[34] usually present additional complexity, slow and energy-inefficient inferences [16], and OOD detection performance worse than ODIN [18], [35].

In this paper, rather than circumvent SoftMax loss limitations, we follow a different strategy: we propose replacing it altogether with a novel loss that does not present the SoftMax loss mentioned drawbacks. Additionally, we used entropy for OOD detection. By doing so, we were able to improve the neural networks' OOD detection baseline performance significantly. *Using this novel baseline OOD detection approach, previously mentioned or future techniques may improve the neural networks' OOD detection performance even further*. Therefore, we propose IsoMax, a loss that is isotropic (exclusively distance-based) and produces high entropy posterior probability distributions in agreement with the maximum entropy principle. We also propose using the entropy of the output probabilities as a score to perform OOD detection.

As we aim to increase the neural networks' baseline OOD detection performance, we do not rely on collecting additional/outlier/background data and validating the appropriated hyperparameters to improve the OOD detection results [26]–[29]. As mentioned before, these (and other) current OOD techniques may be used in future works to improve our baseline OOD detection performance. Furthermore, with the sole aim to perform OOD detection, we emphasize that, unlike uncertainty/confidence estimation/calibration methods, our intention is not to produce calibrated probabilities. In the opposite direction, we plan to force the network to provide a posterior probability distribution with the highest possible entropies (as stated by the maximum entropy principle), which correspond to low maximum probabilities.

Networks trained using IsoMax loss produce accurate predictions, as no classification accuracy drop is observed compared to networks trained with SoftMax loss. Additionally, the models trained using IsoMax loss provide *inferences that are fast and have energy and computational efficiency* *equivalent to models trained with SoftMax loss*, making our solution viable from an economical and environmental point of view [36]. Moreover, our approach does not require validation using unrealistic access to OOD samples or the generation of adversarial examples. Furthermore, our solution is turnkey since it does not require additional/outlier data, feature extraction, metric learning, or hyperparameter tuning. No extra procedures other than typical network training are required.

We provide insights based on the maximum entropy principle to explain why our approach works. Experimental evidence confirms our theoretical assumptions and shows that our straightforward OOD detection solution significantly outperforms the SoftMax loss performance and is even competitive with OOD detection methods without requiring their supplementary techniques and associated side effects (Table I).

Indeed, despite being proposed as a baseline OOD detection solution that avoids current approaches drawbacks (e.g., classification accuracy drop, slow and energy-inefficient inferences, hyperparameter tuning, feature extraction, metric learning, and additional data), our approach provides high OOD detection performance. *Naturally, additional performance gains may be obtained in the future by adapting OOD methods to use IsoMax loss rather than SoftMax loss.*

In summary, the major contributions of this article are:

1) The intuitions that associate the unsatisfactory OOD detection performance of current neural networks with the SoftMax loss anisotropy and disagreement with the maximum entropy principle.
2) The IsoMax loss, which is isotropic, in agreement with the maximum entropy principle, and works as a *seamless SoftMax loss drop-in replacement*. The IsoMax loss trained models present accurate predictions (no classification accuracy drop) and fast inferences that are energy- and computation-efficient. Besides, it does not require additional/outlier/background data. Under these severe restrictions, our solution provides *state-of-the-art* OOD detection performance.
3) The experimental demonstration that entropy, which is a fast and energy-efficient computation, provides high OOD detection performance when the posterior probability distribution follows the maximum entropy principle, i.e., it presents high mean entropy. When using the entropy, rather than just one output, all network outputs are considered.
4) Our approach improves deep neural networks' baseline OOD detection performance. In applications where the previously mentioned requirements and side effects of current OOD techniques are not a concern, future works may combine them with our loss to achieve even higher OOD detection performance.

In Section II, we emphasize the technical aspects of modern OOD detection approaches. In Section III, we present our proposal for a novel baseline OOD detection method. We present the experiments, results, and discussions in Section IV. In Section V, we conclude and present future works. This paper is an expanded version of [37].

II. Background

A. Distance-based Losses

Recently, neural network distance-based losses have been proposed in the context of face recognition. For example, the contrastive [38] and triplet [39] losses use high-level feature (embeddings) *pairwise* distances. In both cases, the SoftMax *function*[2] is not present, and the *squared* Euclidean distance is used. One of the main drawbacks is the need for using *Siamese neural networks*, which adds complexity to the solution and expands memory requirements during training [10]. Additionally, the *triplet sampling* and *pairwise training*, which implicate the recombination of the training samples with dramatic data expansion, slow convergence and instability [10]. Finally, no prototypes are learned during training. The challenge to train networks using *purely* distance-based losses while avoiding *triplet sampling* and *pairwise training* was discussed in [10], which proposed a *squared* Euclidean distance-based *regularization* method.

The center loss [10] has two parameters α and λ. We call a loss isotropic when its dependency on the high-level features (embeddings) is performed *exclusively* through distances, which are usually calculated from the class prototypes. In this sense, the center loss is *not* isotropic, as it presents an affine transformation in its SoftMax classification term. Therefore, the center loss inherits the previously mentioned drawbacks of the SoftMax loss affine transformation.

In [40], the authors proposed a solution based on *squared* Euclidean distance to address few-shot learning. However, this approach does not work as a SoftMax loss drop-in replacement, as it does not simultaneously learn high-level features (embeddings) *and prototypes* using *exclusively* stochastic gradient descent (SGD) and *end-to-end* backpropagation. Indeed, despite learning embeddings using regular SGD and backpropagation, *additional offline procedures are required to calculate the class prototypes* in the mentioned approach.

B. Out-of-Distribution Detection

ODIN was proposed in [1] by combining input preprocessing with *temperature calibration*, which consists of changing the scale of the logits of a pretrained model. Despite significantly outperforming the SoftMax loss, the input preprocessing introduced in ODIN considerably increases the inference time by requiring an initial forward pass, a backpropagation, and finally a second forward pass to perform an inference that can be used for OOD detection. Considering that backpropagation is typically slower than a forward pass, *input preprocessing makes ODIN inferences at least three times slower than normal*. Additionally, input preprocessing multiplies the inference power consumption and computational cost by a factor of at least three. Those are severe limitations from an economic and environmental perspective [36][3]. Several subsequent OOD detection proposals incorporated input preprocessing and its drawbacks [1], [2], [18], [41]. Both input preprocessing and temperature calibration *require hyperparameter tuning*.

[2]We follow the "SoftMax *function*" expression as defined in [11].
[3]https://www.youtube.com/watch?v=KnOpWgUCtaM

Consequently, ODIN requires unrealistic access to OOD samples to validate hyperparameters. Even if the supposed OOD samples are available during design-time, using these examples to tune hyperparameters makes the solution overfit to detect this particular type of out-distribution[4]. The system will likely operate under unknown out-distributions in real-world applications, and the estimated OOD detection performance could degrade significantly. Therefore, using OOD samples to validate hyperparameters may produce over-optimistic OOD detection performance estimations [35].

A method to avoid using OOD samples was proposed in [18]. The mentioned method uses only *in-distribution* validation data for tuning the required hyperparameters. However, the proposed loss produced a significant classification accuracy drop in some situations. Moreover, the *in-distribution* CIFAR10/100 validation sets were used for both hyperparameter tuning and OOD detection evaluation. Therefore, the classification accuracy drop reported in [18] is probably underestimated. In practice, the necessary *in-distribution* validation data for hyperparameter tuning will have to be removed from the training data, which will presumably decrease even further the classification accuracy and, consequently, also the OOD detection performance. Additionally, [18] also uses input preprocessing, making inferences considerably inefficient from an environment, economic, and energy perspective.

The Mahalanobis distance-based method[5] [2] overcomes the need for access to OOD samples by validating hyperparameters using adversarial examples and producing more realistic OOD detection performance estimates. However, using adversarial examples has the disadvantage of adding a cumbersome procedure to the solution. Even worse, the generation of adversarial samples itself requires hyperparameter tuning such as the maximum perturbations. While adequate hyperparameters may be known for research datasets, they may be challenging to find for novel real-world data.

Moreover, as the Mahalanobis approach also requires input preprocessing, the previously mentioned drawbacks associated with this technique are still present in the Mahalanobis solution. *Feature ensemble* introduced in this approach also presents limitations. Indeed, *feature ensembles* require training of extra classification and regression models on features extracted from many network layers. For applications using real-world large-size images, these shallow models may present scalability problems, as they would be required to work in spaces of tens of thousands of dimensions. Finally, the Mahalanobis method is not turnkey, as it involves feature extraction and metric learning post-processing.

Hyperparameter tuning is also a drawback to methods based on adversarial training (e.g., ACET [3]), as we need adequate adversarial perturbations. Additionally, adversarial training is known for increasing training time [20] and reducing classification accuracy [22]. Furthermore, solutions based on adversarial training may not scale to applications dealing with real-world large-sized images [21], which may also be the case of approaches with high computational cost inferences.

The Entropic Open-Set loss and the Objectosphere loss were proposed in [28]. These two losses used background samples to improve the performance of detecting unknown inputs. The Entropic Open-Set loss works like the usual SoftMax loss in the in-distribution training data, producing low entropy for these samples. However, it forces maximum entropy in the background samples. The Objectosphere loss is the Entropic Open-Set loss with an added regularization factor that forces the feature magnitude of in-distribution samples to be near a predefined value ξ while minimizing the feature magnitude of background samples.

Domain-specialized methods have also been proposed. Tack et al. [42] used data augmentation for image data to improve OOD detection in a self-supervised setting. Sastry et al. [43] analyzed statistics of the activations of the pretrained model on training and validation data to detect OOD examples. Hence, we believe one of the main advantages of our approach is that it is *domain-agnostic*, as it can be applied to any domain.

III. Entropic Out-of-Distribution Detection

A. Isotropy Maximization Loss

To mitigate the SoftMax loss drawbacks (i.e., anisotropy and low entropy posterior probability distributions), we design the IsoMax loss by imposing isotropy combined with high (almost maximum) entropy posterior probability distributions, which is obtained using what we call "the entropy maximization trick".

1) Isotropy: To fix the SoftMax loss anisotropy caused by its affine transformation, we propose the IsoMax loss, which is isotropic in the sense that its logits depend *exclusively* on distances from high-level features to class prototypes. For more information regarding the SoftMax loss anisotropy, how it harms the OOD detection performance, and the advantages of circumventing this problem using an isotropic loss, please see Supplementary Material A.

We designed the IsoMax loss to work as a SoftMax loss drop-in replacement. Therefore, the swap of the SoftMax loss with the IsoMax loss requires changes in neither the model's architecture nor training procedures or parameters.

Let $\boldsymbol{f_\theta}(\boldsymbol{x})$ represent the high-level feature (embedding) associated with $\boldsymbol{x}$, $\boldsymbol{p}_{\boldsymbol{\phi}}^j$ represent the prototype associated with the class j, and $d()$ represent a distance. To construct a isotropic loss, we need to avoid *direct* dependency on $\boldsymbol{f_\theta}(\boldsymbol{x})$ or $\boldsymbol{p}_{\boldsymbol{\phi}}^j$. Therefore, the loss has to be a function that *exclusively* depends on the embedding-prototype distances given by $d(\boldsymbol{f_\theta}(\boldsymbol{x}), \boldsymbol{p}_{\boldsymbol{\phi}}^j)$. Therefore, we can write:

$$\mathcal{L}_I = g(d(\boldsymbol{f_\theta}(\boldsymbol{x}), \boldsymbol{p}_{\boldsymbol{\phi}}^j)) \tag{1}$$

In the previous equation, $g()$ represents a scalar function. The expression $d(\boldsymbol{f_\theta}(\boldsymbol{x}), \boldsymbol{p}_{\boldsymbol{\phi}}^j)$ represents the isotropic layer, where its weights are given by the learnable prototypes $\boldsymbol{p}_{\boldsymbol{\phi}}^j$.

We decided to normalize the embedding-prototype distances using the SoftMax *function* to allow interpretation in terms of probabilities. Therefore, the embedding-prototype distances represent the logits of the SoftMax *function* and correspond

[4]In this paper, the expression "out-distribution" refers to any distribution that generates OOD samples.

[5]For the rest of this paper, the expression "the Mahalanobis distance-based method" is replaced by "the Mahalanobis method".

to the output of the isotropic layer. We also decided to use cross-entropy for efficient optimization. Hence, we can write the following equation:

$$\mathcal{L}_I(\hat{y}^{(k)}|\boldsymbol{x}) = -\log\left(\frac{\exp(-d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^k))}{\sum\limits_j \exp(-d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^j))}\right) \tag{2}$$

In the above equation, $\hat{y}^{(k)}$ represents the label of the correct class, while the negative logarithm represents the cross-entropy. The negative terms before the distances are necessary to indicate the negative correlation between distances and probabilities. The expression between the outermost parentheses applied to the term $-d(\boldsymbol{f_\theta}(\boldsymbol{x}),\boldsymbol{p}_{\boldsymbol{\phi}}^j)$ represents the SoftMax *function*. We use the non-squared Euclidean distance for $d(.,.)$, and justify its use in Supplementary Material B.

Unlike usual metric learning-based OOD detection approaches, rather than learning a metric from a preexisting feature space (metric learning on features extracted from a pretrained model), when using the IsoMax loss, we learn a feature space that is, from the start, consistent with the chosen metric. Indeed, the minimization of Equation (2) is achieved by making the expression inside the outer parentheses goes to one. This is only possible by reducing the distances between the high-level features (embeddings) and the associated class prototypes while simultaneously keeping high distances among class prototypes. Hence, the main aim of metric learning, which is to reduce intraclass distances while increasing interclass distances, is performed naturally during the neural network training, avoiding feature extraction and metric learning post-processing phases.

2) Entropy Maximization: Isotropy increases OOD detection performance. However, we need to circumvent the SoftMax loss propensity to produce significantly low entropy posterior probability distributions for further gains. To achieve high entropy probability distributions in agreement with the maximum entropy principle, we introduce the *entropic scale*, a constant scalar multiplicative factor applied to the logits *throughout training* that is, nevertheless, *removed before inference*. We call this procedure "the entropy maximization trick".

Our experiments show that the proposed strategy forces the neural networks to produce outputs with significantly high entropy probability distributions (as recommended by the maximum entropy principle) rather than the usual low entropy (overconfident) posterior probability distributions.

The entropic scale is related to the *inverse of the temperature* of the SoftMax *function*. However, training with a *predefined constant* entropic scale and then *removing it before inference* is *completely* different from temperature calibration. On the one hand, in ODIN and similar methods based on temperature calibration, the temperature of a *pretrained* model is validated *after training*, which nevertheless was performed with a temperature equal to one. This validation usually requires unrealistic access to OOD or adversarial examples, which additionally make them not turnkey. Additionally, over-optimistic performance estimation is commonly produced [35]. On the other hand, our approach requires neither hyperparameter validation nor access to OOD or adversarial data.

The principle of maximum entropy, formulated by E. T. Jaynes to unify the statistical mechanics and information theory entropy concepts [13], [14], states that when estimating probability distributions, we should choose the one that produces the maximum entropy consistent with the given constraints [15]. Following this principle, we avoid introducing additional assumptions or bias not presented in the data.

Hence, from a set of trial probability distributions that satisfactorily describe the prior knowledge available, the distribution that presents the maximal information entropy (i.e., the least informative option) represents the best possible choice. In other words, we must produce posterior probability distributions as underconfident as possible as long as they match the correct predictions.

The principle of maximum entropy has been studied as a *regularization factor* [44], [45]. In some cases, it has also been used as a direct optimization procedure without connection to cross-entropy minimization or backpropagation. For example, in [46]–[48], the maximization of the entropy subject to a constraint on the expected classification error was shown to be equivalent to solving an unconstrained Lagrangian.

The relation between the principle of maximum entropy and Bayesian methods has been studied in [49]. Despite being theoretical well-grounded [50]–[53], *direct* entropy maximization presents high computational complexity, as it is an NP-complete problem [50], [52]. Alternatively, modern neural networks are trained using computationally efficient cross-entropy. However, this procedure does not prioritize high entropy (low confidence) posterior probability distributions. *Actually, the opposite is true*. Indeed, the minimization of cross-entropy has the undesired side effect of producing low entropy (overconfident) posterior probability distributions [12].

Unlike the previously mentioned works, we use the principle of maximum entropy neither to motivate the construction of regularization mechanisms, such as label smoothing or the confidence penalty [44], [45], nor to perform *direct* maximum entropy optimization [46]–[48]. The entropy is not even calculated during IsoMax loss training. In the opposite direction, we use the principle of maximum entropy as a theoretical motivation for constructing high entropy (low confidence) posterior probabilities, still relying on computationally efficient cross-entropy minimization. Since our approach does *not* directly maximize the entropy, we cannot state that IsoMax produces the maximum entropy posterior probability distribution. However, the entropies produced by IsoMax loss are high enough to improve the OOD detection performance significantly.

$$\mathcal{L}_{\textsf{SoftMax}} = -\log\left(\frac{\exp(L_k)}{\sum\limits_j \exp(L_j)}\right) \rightarrow 0 \tag{3}$$

$$\implies \mathcal{P}(y|\boldsymbol{x}) \rightarrow 1 \tag{4}$$

$$\implies \mathcal{H}_{\textsf{SoftMax}} \rightarrow 0 \tag{5}$$

Equation (3) describes the behavior of the cross-entropy and entropy for the SoftMax loss. L_j represents the logits associated with class j, and L_k represents the logits associated with the correct class k. When minimizing the loss

(Equation (3)), high probabilities are generated (Equation (4)). Consequently, significantly low entropy posterior probability distributions are produced (Equation (5)). Hence, the usual cross-entropy minimization tends to generate unrealistic over-confident (low entropy) probability distributions. Therefore, we have an opposition between cross-entropy minimization and the principle of maximum entropy.

$$\mathcal{L}_{\text{IsoMax}} = -\log\left(\frac{\exp(-E_s \times D_k)}{\sum\limits_j \exp(-E_s \times D_j)}\right) \rightarrow 0 \tag{6}$$

$$\not\Rightarrow \mathcal{P}(y|\boldsymbol{x}) \rightarrow 1 \tag{7}$$

$$\not\Rightarrow \mathcal{H}_{\text{IsoMax}} \rightarrow 0 \tag{8}$$

The IsoMax loss conciliates these contradictory objectives (loss minimization and entropy maximization) by multiplying the logits by a constant positive scalar E_s, which is presented during training but removed before inference. Equation (6) demonstrates how the entropic scale (presented at training time but removed at inference time) allows the production of high entropy posterior distributions despite using cross-entropy minimization. D_j represents the distances associated with class j, and D_k represents the distances associated with the correct class k. The E_s present during training allows the term $-E_s \times D_k$ to become high enough (less negative compared to $-E_s \times D_j$) to produce low loss (Equation (6)) *without* producing high probabilities for the correct classes, as they are calculated with the E_s removed (Equation (7)). Thus, it is possible to build posterior probability distributions with high entropy (Equation (8)) in agreement with the fundamental principle of maximum entropy despite using cross-entropy to minimize the loss.

We emphasize that, regardless of training with the entropic scale, if we do not remove it before performing inference, the IsoMax loss produces outputs with entropies as low as those produced by the SoftMax loss, the OOD detection performance does not further increase. Hence, returning to Equation (2), multiplying the embedding-prototype distances by E_s, and making $d()$ equal to the *nonsquared* Euclidean distance, we write the definitive IsoMax loss as:

$$\begin{aligned}\mathcal{L}_I(\hat{y}^{(k)}|\boldsymbol{x}) &= -\log^\dagger\left(\frac{\exp(-E_s\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^k_{\boldsymbol{\phi}}\|)}{\sum\limits_j \exp(-E_s\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^j_{\boldsymbol{\phi}}\|)}\right)\\ &= -\log^\dagger\left(\frac{\exp(-E_s\sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^k_{\boldsymbol{\phi}})\cdot(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^k_{\boldsymbol{\phi}})})}{\sum\limits_j \exp(-E_s\sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^j_{\boldsymbol{\phi}})\cdot(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^j_{\boldsymbol{\phi}})})}\right)\end{aligned} \tag{9}$$

We emphasize that removing the entropic scale after the training does not affect the solution's ability to represent the prior knowledge available, as it does not change the predictions. Therefore, the expression for the probabilities with the entropic scale removed is preferable, as it increases the entropy of the posterior distribution in agreement with the principle of maximum entropy. Hence, the inference probabilities for the IsoMax loss are defined as follows:

$$\begin{aligned}p_I(y^{(i)}|\boldsymbol{x}) &= \frac{\exp(-\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^i_{\boldsymbol{\phi}}\|)}{\sum\limits_j \exp(-\|\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^j_{\boldsymbol{\phi}}\|)}\\ &= \frac{\exp(-\sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^k_{\boldsymbol{\phi}})\cdot(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^k_{\boldsymbol{\phi}})})}{\sum\limits_j \exp(-\sqrt{(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^j_{\boldsymbol{\phi}})\cdot(\boldsymbol{f_\theta}(\boldsymbol{x})-\boldsymbol{p}^j_{\boldsymbol{\phi}})})}\end{aligned} \tag{10}$$

Please, notice that the entropic scale E_s was intentionally removed from the loss Equation 9 to build the inference Equation 10 to perform "the entropy maximization trick".

3) Initialization and Implementation Details: We experimentally observed that using the Xavier [54] or Kaiming [55] initializations for prototypes makes the OOD detection performance oscillate. Sometimes it improves, sometimes it decreases. Hence, we decided to initialize all prototypes to zero vector, as this is the most natural value for untrained embeddings. Weight decay is applied to the prototypes, as they are regular trainable parameters.

To calculate losses based on cross-entropy, deep learning libraries usually combine the probability and logarithm calculations into a single computation. *However, we experimentally observed that sequentially computing these calculations as stand-alone operations improves IsoMax performance.*

The class prototypes have the same dimension as the neural network last layer representations. Naturally, the number of prototypes is equal to the number of classes. Therefore, the IsoMax loss has fewer parameters than the SoftMax loss, as it has no bias to be learned.

Finally, we verified classification accuracy drop and low or oscillating OOD detection performance when trying to integrate E_s with cosine similarity [11], [56], [57] or the affine transformations used in SoftMax loss. In such cases, the above trick to initialize the prototypes to zero vector cannot be performed. Moreover, the Mahalanobis distance cannot be used directly during the neural network training because the covariance matrix is not differentiable. Therefore, we confirmed *nonsquared* Euclidean distance as the best option to build the IsoMax loss.

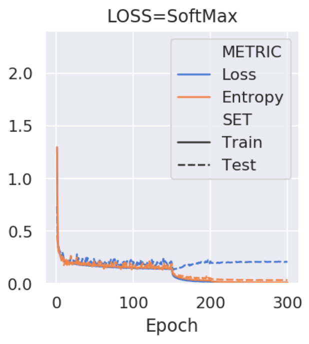
B. Entropic Score

We define our score to perform OOD detection, called the entropic score, as the output probabilities negative entropy:

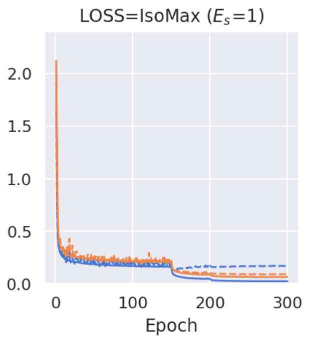
$$\mathcal{ES} = -\sum_{i=1}^{N} p(y^{(i)}|\boldsymbol{x})\log p(y^{(i)}|\boldsymbol{x}) \tag{11}$$

Using the negative entropy as a score to evaluate whether a particular sample is OOD, we consider the information provided by all available network outputs rather than just

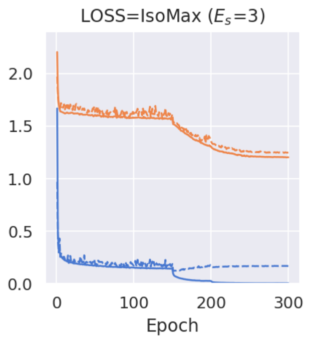
†*The probability (i.e., the expression between the outermost parenteses) and logarithm operations are computed sequentially and separately for higher OOD detection performance (please, see the source code).*



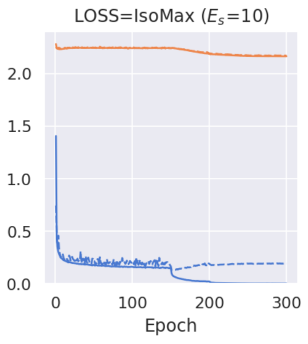
(a)



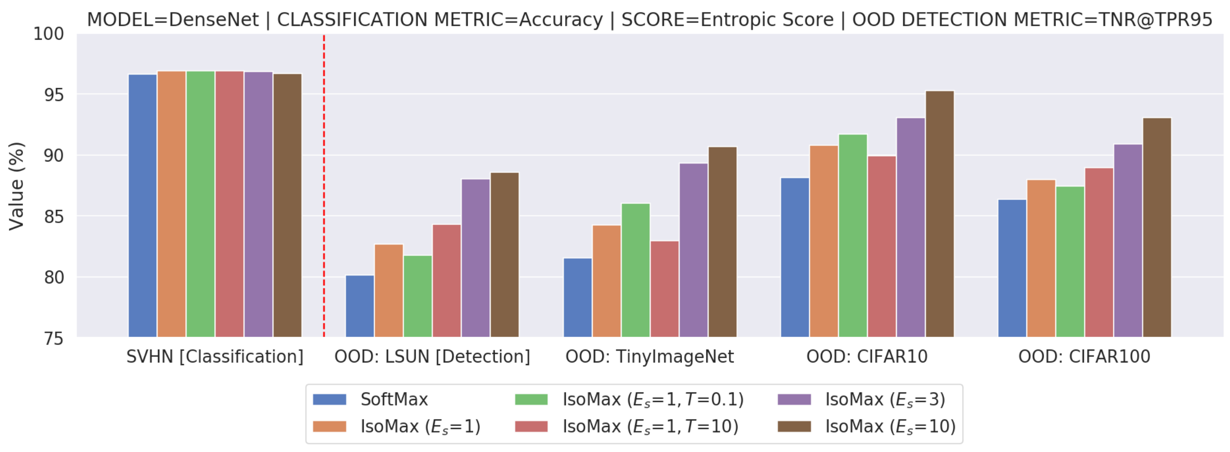
(b)



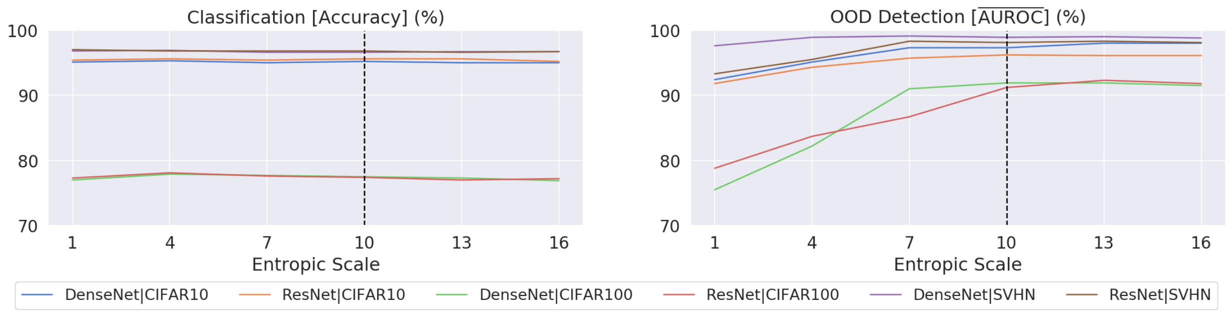
(c)



(d)



(e)



(f)

Fig. 2. IsoMax loss effects: (a) SoftMax loss simultaneously minimizes both the cross-entropy and the entropy of the posterior probabilities. (b) IsoMax loss produces low entropy posterior probabilities for a low entropic scale (E_s=1). (c) IsoMax loss produces medium mean entropy for an intermediate entropic scale (E_s=3). (d) IsoMax loss minimizes the cross-entropy while producing high mean entropies for the high entropic scale (E_s=10). *"The entropy maximization trick" is the fundamental mechanism that allows us to migrate from low entropy posterior probability distributions (a,b) to high entropy posterior probability distributions (d). Higher entropic scale values correlate to higher mean entropies as recommended by the principle of maximum entropy. Regardless of training with a high entropic scale, if we do not remove it for inference ("the entropy maximization trick"), the IsoMax loss always produces posterior probability distributions with entropies as low as those generated by the SoftMax loss. An entropic scale equal to ten is sufficient to virtually produce the maximum possible entropy posterior probability distribution, as the highest possible value of the entropy is* $\log(N)$, *where* N *is the number of classes.* (e) The left side of the dashed vertical red line presents classification accuracies. The dashed vertical red line's right side shows OOD detection performance using the entropic score and the TNR@TPR95 metric. Higher mean entropies produce increased OOD detection performance regardless of the out-distribution. We emphasize that OOD examples were never used during training and no validation was required to tune hyperparameters. Additionally, isotropy enables IsoMax loss to produce higher OOD performance than SoftMax loss, even for the unitary value of the entropic scale. Training using E_s=1 and then making the temperature T=0.1 or T=10 during inference produces lower OOD detection performance than training using E_s=10 and removing it for inference, which consists in our proposal. Finally, IsoMax loss presents similar classification accuracy compared with SoftMax regardless of the entropic scale. The entropic score, which is the negative entropy of the network output probabilities, was used as the score. (f) Study of classification accuracy and OOD detection performance dependence on the entropic scale. $\overline{\text{AUROC}}$ represents the mean AUROC considering all out-of-distribution data. The classification accuracy and the mean OOD detection performance are approximately stable for E_s=10 or higher regardless of the dataset and model. It also does not depend on the number of training classes N, which is probably explained by the fact that the entropic scale is inside an exponential function while the entropy increases only with the logarithm of the number of classes. Making E_s learnable did not considerably improve or decrease the OOD detection results.

one. For instance, ODIN and ACET only use the maximum probability, while the Mahalanobis method only uses the distance to the nearest prototype. The entropy has been studied before for anomaly detection [58].

Regarding the use of entropy as an OOD score, one of the most relevant contributions of this paper is to experimentally show that it does not significantly overcome the maximum probability score when dealing with the usual low entropy (overconfident) neural networks' posterior probability distributions. However, the entropic score significantly improves the OOD detection performance when used with the high entropy (underconfident) neural networks' posterior probability distributions produced by the IsoMax loss in agreement with the principle of maximum entropy. For more information regarding the use of entropy as a score to perform OOD, please see Supplementary Material C. For a detailed differentiation between our approach and the proposed in [17], please see Supplementary Material D.

IV. Experimental Analyses

All datasets, models, training procedures, and evaluation metrics followed the baseline established in [8] and were subsequently used in major OOD detection papers [1]–[3], [26]. We only compared approaches that did not present classification accuracy drop in the experiments presented in the work in which they were proposed. The *code* is available[6].

We trained from scratch 100-layer DenseNet-BC [61] (growth rate $k=12$, 0.8M parameters) and 34-layer ResNets [62] on CIFAR10 [63], CIFAR100 [63] and SVHN [64] using SoftMax and IsoMax losses. We used SGD with the Nesterov moment equal to 0.9, 300 epochs with a batch size of 64, and an initial learning rate of 0.1 with a learning rate decay rate equal to ten applied in epochs 150, 200, and 250. We used dropout equal to zero. The weight decay was 0.0001. We used resized images from the TinyImageNet [65][7] and the Large-scale Scene UNderstanding (LSUN) [66][7] datasets as OOD data. Each set of OOD data presents ten thousand images.

Finally, we separately added these OOD data to the CIFAR10, CIFAR100, and SVHN validation sets to construct the respective OOD detection test sets. For example, the OOD detection test set for in-distribution CIFAR10 and out-distribution TinyImageNet is composed by combing the CIFAR validation set and the TinyImageNet OOD data. In some cases, CIFAR10 and SVHN work as out-distribution and the respective validation set is used to construct the OOD detection test set. We emphasize that our solution does not use validation sets for training or validation. Therefore, validation sets are used exclusively for building OOD detection test sets.

For text data experiments, we followed the experimental setting presented in [26]. Therefore, we used the 20 Newsgroups as in-distribution data. The 20 Newsgroups is a text classification dataset of newsgroup documents. It has 20 classes and approximately 20,000 examples that are split evenly among the classes. We used the standard 60/40 train/test split. We used the IMDB, Multi30K, and Yelp Reviews datasets as out-distribution. IMDB is a dataset of movie review sentiment classification. Multi30K is a dataset of English-German image descriptions, of which we use the English descriptions. Yelp Reviews is a dataset of restaurant reviews. Finally, we trained from scratch 2-layer GRUs (GRU2L) [67] using SoftMax and IsoMax losses.

We evaluated the performance using three detection metrics. First, we calculated the true negative rate at the 95% true positive rate (TNR@TPR95) and the false positive rate at the 90% true positive rate (FPR@TPR90) using the adequate thresholds. In addition, we evaluated the area under the receiver operating characteristic curve (AUROC)and the detection accuracy (DTACC), which corresponds to the maximum classification probability over all possible thresholds δ:

$$1-\min_{\delta}\big\{P_{\texttt{in}}\left(o\left(\mathbf{x}\right)\leq\delta\right)P\left(\mathbf{x}\text{ is from }P_{\texttt{in}}\right) \\ +P_{\texttt{out}}\left(o\left(\mathbf{x}\right)>\delta\right)P\left(\mathbf{x}\text{ is from }P_{\texttt{out}}\right)\big\}, \tag{12}$$

where $o(\mathbf{x})$ is a given OOD detection score. It is assumed that both positive and negative samples have equal probability. In other words, the DTACC represents the probability that we can correctly classify whether a sample belongs to the in-distribution or the out-distribution considering the best case scenario, i.e., using the ideal value for δ. AUROC and DTACC are threshold independent.

A. Entropic Scale, High Entropy, and OOD Detection

To experimentally show that higher entropic scales lead to higher entropy probability distributions and consequently improved OOD detection performance, we trained DenseNets on SVHN using the SoftMax loss and IsoMax loss with distinct entropic scale values. We used the entropic score and the TNR@TPR95 to evaluate OOD detection performance (please, see Fig. 2).

As expected, Fig. 2a shows that the SoftMax loss generates posterior distributions with low entropy. Fig. 2b illustrates that the unitary entropic scale ($E_s=1$) only slightly increases the distribution mean entropy. In other words, isotropy alone is not enough to circumvent the cross-entropy propensity to produce low entropy probability distributions and the entropic scale is indeed necessary. Nevertheless, the replacement of anisotropic logits based on affine transformation by isotropic logits is enough to produce initial OOD detection performance gains regardless of the out-distribution (Fig. 2e).

Training using $E_s=1$ and then making during inference the temperature $T=0.1$, which is equivalent to make $E_s=10$, produces different results from training using $E_s=10$ and removing it for inference. Indeed, in the first case we only change the last layer, while in the second case, we learn different weights for the entire neural network in comparison with training with $E_s=1$ (Fig. 2e).

Fig. 2c shows that an intermediate entropic scale ($E_s=3$) provides medium entropy probability distributions with corresponding additional OOD detection performance gains for all out-distributions (Fig. 2e). Fig. 2d illustrates that a high entropic scale ($E_s=10$) produces even higher entropy probability distributions and the highest OOD detection performance regardless of the out-distribution considered (Fig. 2e).

[6]https://github.com/dlmacedo/entropic-out-of-distribution-detection
[7]https://github.com/facebookresearch/odin

TABLE II
CURRENT AND PROPOSED BASELINE OUT-OF-DISTRIBUTION DETECTION APPROACHES COMPARISON.
FAST AND ENERGY-EFFICIENT INFERENCES. TURNKEY APPROACHES. NO EXTRA/OUTLIER/BACKGROUND DATA USED.

Model	In-Data (training)	Out-Data (unseen)	Baseline Out-of-Distribution Detection Approaches Comparison. Fast and Energy-Efficient Inferences. Turnkey Approaches. No outlier data used.		
			TNR@TPR95 (%) [↑]	AUROC (%) [↑]	DTACC (%) [↑]
			SoftMax+MPS [8] / SoftMax+ES / IsoMax+ES (ours)		
DenseNet	CIFAR10	SVHN	32.1 ± 0.3 / 33.1 ± 0.4 / $\mathbf{77.1\pm0.3}$	86.5 ± 0.4 / 86.8 ± 0.3 / $\mathbf{96.7\pm0.4}$	79.8 ± 0.3 / 79.8 ± 0.3 / $\mathbf{91.8\pm0.3}$
		TinyImageNet	55.7 ± 0.3 / 59.7 ± 0.4 / $\mathbf{88.1\pm0.4}$	93.5 ± 0.4 / 94.1 ± 0.4 / $\mathbf{97.9\pm0.3}$	87.5 ± 0.2 / 87.9 ± 0.3 / $\mathbf{93.3\pm0.3}$
		LSUN	64.8 ± 0.4 / 69.6 ± 0.3 / $\mathbf{94.6\pm0.2}$	95.1 ± 0.3 / 95.8 ± 0.2 / $\mathbf{98.9\pm0.3}$	89.8 ± 0.3 / 90.1 ± 0.3 / $\mathbf{95.0\pm0.4}$
	CIFAR100	SVHN	20.5 ± 0.6 / $\mathbf{24.8\pm0.8}$ / $\mathbf{23.6\pm0.9}$	80.1 ± 0.7 / 81.8 ± 0.7 / $\mathbf{88.8\pm0.6}$	73.8 ± 0.6 / 74.4 ± 0.7 / $\mathbf{83.9\pm0.6}$
		TinyImageNet	19.3 ± 0.9 / 23.8 ± 0.8 / $\mathbf{49.0\pm0.6}$	77.1 ± 0.6 / 78.7 ± 0.7 / $\mathbf{92.8\pm0.6}$	70.5 ± 0.6 / 71.3 ± 0.7 / $\mathbf{86.5\pm0.6}$
		LSUN	18.7 ± 0.6 / 24.3 ± 0.8 / $\mathbf{63.1\pm0.6}$	75.8 ± 0.7 / 77.8 ± 0.6 / $\mathbf{94.8\pm0.6}$	69.4 ± 0.8 / 70.3 ± 0.9 / $\mathbf{89.2\pm0.6}$
	SVHN	CIFAR10	81.5 ± 0.2 / 83.7 ± 0.3 / $\mathbf{94.1\pm0.2}$	96.5 ± 0.3 / 96.9 ± 0.2 / $\mathbf{98.5\pm0.2}$	91.9 ± 0.3 / 92.1 ± 0.2 / $\mathbf{95.0\pm0.2}$
		TinyImageNet	88.3 ± 0.2 / 90.1 ± 0.2 / $\mathbf{97.2\pm0.3}$	97.7 ± 0.3 / 98.1 ± 0.2 / $\mathbf{99.1\pm0.2}$	93.4 ± 0.4 / 93.8 ± 0.2 / $\mathbf{96.3\pm0.3}$
		LSUN	86.4 ± 0.2 / 88.4 ± 0.4 / $\mathbf{96.8\pm0.2}$	97.3 ± 0.2 / 97.8 ± 0.3 / $\mathbf{99.1\pm0.2}$	92.8 ± 0.2 / 93.0 ± 0.4 / $\mathbf{96.0\pm0.2}$
ResNet	CIFAR10	SVHN	43.2 ± 0.4 / 44.5 ± 0.3 / $\mathbf{83.6\pm0.4}$	91.6 ± 0.3 / 92.0 ± 0.3 / $\mathbf{97.1\pm0.4}$	86.5 ± 0.2 / 86.4 ± 0.4 / $\mathbf{91.9\pm0.3}$
		TinyImageNet	46.4 ± 0.4 / 48.0 ± 0.3 / $\mathbf{70.3\pm0.3}$	89.8 ± 0.4 / 90.1 ± 0.2 / $\mathbf{94.6\pm0.4}$	84.0 ± 0.4 / 84.2 ± 0.3 / $\mathbf{88.3\pm0.4}$
		LSUN	51.2 ± 0.4 / 53.3 ± 0.2 / $\mathbf{82.3\pm0.3}$	92.2 ± 0.4 / 92.6 ± 0.4 / $\mathbf{96.9\pm0.3}$	86.5 ± 0.2 / 86.6 ± 0.4 / $\mathbf{91.5\pm0.3}$
	CIFAR100	SVHN	15.9 ± 0.8 / 18.0 ± 0.7 / $\mathbf{20.2\pm0.6}$	71.3 ± 0.6 / 72.7 ± 0.7 / $\mathbf{85.3\pm0.6}$	66.1 ± 0.6 / 66.3 ± 0.7 / $\mathbf{79.7\pm0.6}$
		TinyImageNet	18.5 ± 0.8 / 22.4 ± 0.6 / $\mathbf{50.6\pm0.7}$	74.7 ± 0.6 / 76.3 ± 0.7 / $\mathbf{92.0\pm0.7}$	68.8 ± 0.6 / 69.1 ± 0.6 / $\mathbf{85.6\pm0.7}$
		LSUN	18.3 ± 0.8 / 22.4 ± 0.5 / $\mathbf{54.9\pm0.6}$	74.7 ± 0.6 / 76.5 ± 0.7 / $\mathbf{93.3\pm0.6}$	69.1 ± 0.5 / 69.4 ± 0.7 / $\mathbf{87.6\pm0.8}$
	SVHN	CIFAR10	67.3 ± 0.2 / 67.7 ± 0.3 / $\mathbf{92.3\pm0.2}$	89.8 ± 0.2 / 89.7 ± 0.3 / $\mathbf{98.0\pm0.2}$	87.0 ± 0.3 / 86.9 ± 0.3 / $\mathbf{94.1\pm0.2}$
		TinyImageNet	66.8 ± 0.3 / 67.3 ± 0.2 / $\mathbf{94.6\pm0.2}$	89.0 ± 0.3 / 89.0 ± 0.2 / $\mathbf{98.3\pm0.2}$	86.8 ± 0.2 / 86.6 ± 0.4 / $\mathbf{94.8\pm0.4}$
		LSUN	62.1 ± 0.2 / 62.5 ± 0.3 / $\mathbf{90.9\pm0.4}$	86.0 ± 0.2 / 85.8 ± 0.2 / $\mathbf{97.8\pm0.2}$	84.2 ± 0.2 / 84.1 ± 0.3 / $\mathbf{93.6\pm0.4}$

The baseline OOD detection approaches are turnkey (no outlier/background/extra data required, no additional procedures other than typical straightforward neural network training are required), produce no classification accuracy drop, and present fast and energy-efficient inferences. Neither adversarial training, input preprocessing, temperature calibration, feature ensemble, nor metric learning is used. Since there is no need to tune hyperparameters, no access to OOD or adversarial examples is required. SoftMax+MPS means training with SoftMax loss and performing OOD detection using the maximum probability score, which is the approach defined in [8]. SoftMax+ES means training with SoftMax loss and performing OOD detection using the entropic score. IsoMax+ES means training with IsoMax loss and performing OOD detection using the entropic score. The OOD detection test sets are composed of images from both in-data and out-data (see Section IV for additional details). The results represent the mean and standard deviation of five executions. The best values are bold when they overcome the competing approach value outside of the margin of error given by the standard deviations. To the best of our knowledge, *IsoMax+ES presents the state-or-the-art under these restrictive assumptions*.

Despite the entropy scale value used, the cross-entropy is minimized and the classification accuracies produced by SoftMax and IsoMax losses are similar. *For a high entropic scale, the IsoMax loss minimizes the cross-entropy while producing high entropy posterior probability distributions as recommended by the principle of maximum entropy.*

More importantly, higher entropy posterior probability distributions directly correlate with increased OOD detection performances despite the OOD data. *Fig. 2d shows that an entropic scale $E_s\!=10$ is enough to produce probability distributions with essentially the maximum possible entropies. Hence, no additional gains may be obtained by increasing the entropic scale even further.* Indeed, we experimentally observed no further gains for entropic scales higher than ten.

Fig. 2f shows that *regardless of the combination of dataset and model*, the classification accuracy and the mean OOD detection performance are stable for $E_s\!=10$ or higher, as the entropic scale is already high enough to ensure near-maximal entropy. We speculate that $E_s\!=10$ worked satisfactorily regardless of the number of training classes because it is used inside an exponential function while the entropy increases only with the logarithm of the number of training classes. Hence, an eventual validation of E_s would produce an insignificant performance increase. Actually, this is not possible because we consider access to OOD or outlier samples forbidden. Moreover, making E_s learnable did not significantly affect the OOD detection performance.

In other words, considering that the entropic scale E_s is present in Equation (9), we are initially led to think that it needs to be tuned to achieve the highest possible OOD detection performance. However, we emphasize that the experiments showed that the dependence of the OOD detection performance on the entropic scale is remarkably well-behaved. *Essentially, the OOD detection performance monotonically increases with the entropic scale until it reaches a saturation point near $E_s\!=10$ regardless of the dataset or the number of training classes (Fig. 2f).* It may be explained by the fact that the entropic scale is inside an exponential function and that we experimentally observed that $\exp(-10\times d)$ is enough to produce almost maximum entropy regardless of the dataset, model or number of training classes under consideration. Finally, even if we consider that validating for entropic scales higher than ten would allow some minor OOD detection performance improvement, we usually cannot do this because we do not often have access to out-distribution data.

Hence, the well-behaved dependence of the OOD detection performance on the entropic scale allowed us to define the E_s as a *constant scalar* equals to 10 rather than a hyperparameter that needs to be tuned for each novel dataset and model. It is the reason our approach may be used without requiring access to OOD/outlier data. Therefore, *we kept the entropic scale as a constant (global value) equal to ten for all subsequent experiments*. Hence, no validation of the entropic score was performed for different models, datasets, or number of classes.

TABLE III
COMPARISON OF ENHANCED VERSIONS OF SOFTMAX LOSS AND ISOMAX LOSS:
ADDING LABEL SMOOTHING, CENTER LOSS REGULARIZATION, ODIN, AND OUTLIER EXPOSURE TO SOFTMAX LOSS AND ISOMAX LOSS.
THE SIDE EFFECTS AND REQUIREMENTS ADDED TO THE SOLUTION DEPEND ON THE ADD-ON TECHNIQUE.

Model	In-Data (training)	OOD Detection Approach	Enhanced Versions of SoftMax loss and IsoMax loss. Add-on Techniques produce different Side Effects and Requirements.		
			Class. Accuracy (%) [↑] SoftMax / IsoMax	TNR@TPR95 (%) [↑] SoftMax / IsoMax	AUROC (%) [↑] SoftMax / IsoMax
DenseNet	CIFAR10	Baseline	**95.4±0.3 / 95.2±0.3**	53.3±0.4 / **84.1±0.3**	91.9±0.4 / **97.3±0.3**
		+ Label Smoothing	**95.2±0.4 / 95.0±0.3**	**70.7±0.4** / 60.3±0.3	**94.9±0.3** / 80.8±0.3
		+ Center Loss Regularization	**95.3±0.3 / 95.1±0.4**	54.7±0.6 / **87.1±0.3**	92.8±0.3 / **97.6±0.3**
		+ ODIN	**95.4±0.3 / 95.2±0.3**	91.9±0.3 / **95.3±0.4**	**98.2±0.3 / 98.4±0.4**
		+ Outlier Exposure	**95.3±0.4 / 95.6±0.4**	93.8±0.3 / **94.7±0.3**	**98.5±0.3 / 98.8±0.4**
	CIFAR100	Baseline	**77.5±0.6 / 77.5±0.4**	22.3±0.7 / **45.1±0.6**	77.4±0.8 / **91.9±0.6**
		+ Label Smoothing	**77.0±0.4 / 77.2±0.6**	31.5±0.6 / **33.0±0.6**	82.0±0.6 / **88.3±0.7**
		+ Center Loss Regularization	**77.2±0.7 / 77.0±0.6**	30.3±0.7 / **43.7±0.6**	79.5±0.8 / **92.2±0.6**
		+ ODIN	**77.5±0.6 / 77.5±0.4**	64.4±0.7 / **83.1±0.8**	92.5±0.6 / **96.9±0.7**
		+ Outlier Exposure	**77.8±0.6 / 77.5±0.7**	23.0±0.7 / **36.4±0.8**	80.5±0.8 / **89.6±0.6**
	SVHN	Baseline	**96.6±0.2 / 96.6±0.2**	90.1±0.2 / **95.9±0.1**	98.2±0.2 / **98.9±0.2**
		+ Label Smoothing	**96.6±0.2 / 96.7±0.3**	87.5±0.2 / **93.5±0.2**	97.0±0.2 / **97.8±0.3**
		+ Center Loss Regularization	**96.7±0.3 / 96.6±0.2**	88.0±0.3 / **95.9±0.2**	97.9±0.2 / **98.9±0.1**
		+ ODIN	**96.6±0.2 / 96.6±0.2**	95.5±0.2 / **96.7±0.2**	98.8±0.1 / **99.1±0.1**
		+ Outlier Exposure	**96.6±0.3 / 96.7±0.3**	**99.9±0.1 / 99.9±0.1**	**99.9±0.1 / 99.9±0.1**
ResNet	CIFAR10	Baseline	**95.4±0.3 / 95.6±0.4**	50.8±0.4 / **78.6±0.3**	91.2±0.3 / **96.1±0.4**
		+ Label Smoothing	**95.5±0.4 / 95.4±0.3**	54.0±0.4 / **63.5±0.3**	78.9±0.4 / **84.5±0.3**
		+ Center Loss Regularization	**95.6±0.3 / 95.4±0.4**	53.5±0.4 / **80.6±0.3**	90.5±0.3 / **96.6±0.2**
		+ ODIN	**95.4±0.3 / 95.6±0.4**	73.7±0.3 / **85.6±0.3**	93.4±0.2 / **97.2±0.3**
		+ Outlier Exposure	**95.5±0.3 / 95.6±0.3**	91.1±0.3 / **94.2±0.4**	97.7±0.2 / **98.6±0.3**
	CIFAR100	Baseline	**77.3±0.6 / 77.4±0.7**	22.4±0.6 / **41.8±0.7**	80.5±0.6 / **90.1±0.7**
		+ Label Smoothing	**77.7±0.5 / 77.3±0.5**	21.0±0.7 / **33.4±0.6**	81.6±0.6 / **85.9±0.6**
		+ Center Loss Regularization	**77.9±0.5 / 77.4±0.5**	29.5±0.9 / **48.2±0.7**	80.3±0.6 / **90.6±0.6**
		+ ODIN	**77.3±0.6 / 77.4±0.7**	64.0±0.6 / **77.9±0.6**	92.7±0.7 / **95.8±0.6**
		+ Outlier Exposure	**77.3±0.5 / 77.0±0.5**	37.8±0.9 / **41.7±0.8**	86.6±0.6 / **88.6±0.7**
	SVHN	Baseline	**96.8±0.2 / 96.7±0.2**	71.7±0.3 / **92.4±0.2**	94.7±0.3 / **98.0±0.2**
		+ Label Smoothing	**96.9±0.2 / 96.9±0.3**	**86.3±0.2 / 86.4±0.2**	**97.2±0.3** / 94.9±0.2
		+ Center Loss Regularization	**96.9±0.2 / 96.7±0.2**	77.2±0.2 / **82.2±0.2**	94.5±0.2 / **96.1±0.3**
		+ ODIN	**96.8±0.2 / 96.7±0.2**	79.9±0.3 / **93.5±0.3**	95.1±0.2 / **98.2±0.3**
		+ Outlier Exposure	**96.9±0.3 / 96.8±0.2**	**99.9±0.1 / 99.9±0.1**	**99.9±0.1 / 99.9±0.1**

Regardless of using SoftMax loss or IsoMax loss, adding label smoothing (LS) requires validation of the hyperparameter ϵ [59]. The values searched for ϵ were 0.1 [59], [60] and 0.01 [60]. Adding center loss regularization (CLR) to SoftMax loss or IsoMax loss requires validation of the hyperparameter λ [10]. The values searched for λ were 0.01 and 0.003 [10]. We emphasize that the center loss is not used as a stand-alone loss but rather combined as a regularization term with a preexisting baseline loss. In the case of the original paper, the baseline loss was the SoftMax loss. In this paper, we added the mentioned regularization term [10, Equation (5)] also to IsoMax loss to construct the center loss enhanced version of our loss. Regardless of using SoftMax loss or IsoMax loss, adding ODIN [1] requires validation of the hyperparameters ϵ and T. The values searched for these hyperparameters were the same used in the original paper [1]. Adding ODIN implies using input preprocessing, which makes inferences much slower, and energy- and cost-inefficient. We used OOD data to validate the LS, CLR, and ODIN hyperparameters. Adding outlier exposure (OE) [26] to SoftMax loss or IsoMax loss requires collecting outlier data. We used the same outlier data used in [26]. The add-on techniques were not applied to the SoftMax and IsoMax losses combined, but rather individually. The values of the performance metrics TNR@TPR95 and AUROC were averaged over all out-of-distribution data. All results used the entropic score, as it always overcame the maximum probability score. The results represent the mean and standard deviation of five executions. The best values are bold when they overcome the competing approach value outside of the margin of error given by the standard deviations. For OOD detection, the variants of SoftMax and IsoMax losses that presented the best performance for each combination of architecture and dataset are blue. The baseline results are in blue.

TABLE IV
BASELINE OUT-OF-DISTRIBUTION DETECTION APPROACHES COMPARISON: TEXT DATA

Model	In-Data (training)	Out-Data (unseen)	Baseline Out-of-Distribution Detection Approaches Comparison. Fast and Energy-Efficient Inferences. Turnkey Approaches. No outlier data used.	
			FPR@TPR90 (%) [↓]	AUROC (%) [↑]
			SoftMax+MPS [8] / IsoMax+ES (ours)	
GRU2L	20 Newsgroups	IMDB	34.9±0.3 / **22.7±0.4**	85.4±0.3 / **90.0±0.2**
		Multi30K	47.2±0.4 / **44.5±0.3**	80.9±0.4 / **82.7±0.4**
		Yelp Reviews	39.3±0.3 / **26.1±0.6**	82.5±0.4 / **88.2±0.4**

SoftMax+MPS means training with SoftMax loss and performing OOD detection using the maximum probability score. IsoMax+ES means training with IsoMax loss and performing OOD detection using the entropic score. The results represent the mean and standard deviation of five executions. The best values are bold when they overcome the competing approach value outside of the margin of error given by the standard deviations.

TABLE V
COMPARISON OF THE PROPOSED BASELINE OUT-OF-DISTRIBUTION DETECTION APPROACH WITH NO SEAMLESS SOLUTIONS.
UNFAIR COMPARISON OF APPROACHES WITH DIFFERENT REQUIREMENTS AND SIDE EFFECTS. NO EXTRA/OUTLIER/BACKGROUND DATA USED.

Model	In-Data (training)	Out-Data (unseen)	ODIN, ACET, and Mahalanobis present troublesome requirements. ODIN, ACET, and Mahalanobis produce undesired side effects.	
			AUROC (%) [↑]	DTACC (%) [↑]
			ODIN [1] / ACET [3] / IsoMax+ES (ours) / Mahalanobis [2]	
DenseNet	CIFAR10	SVHN	92.7±0.4 / NA / **96.7±0.4 / 97.5±0.6**	86.4±0.4 / NA / **91.8±0.4 / 92.4±0.5**
		TinyImageNet	97.3±0.4 / NA / **97.9±0.4 / 98.5±0.6**	92.1±0.4 / NA / **93.5±0.4 / 94.3±0.6**
		LSUN	98.4±0.4 / NA / **98.9±0.4 / 99.1±0.6**	94.3±0.3 / NA / **95.1±0.4 / 95.9±0.4**
	CIFAR100	SVHN	88.1±0.6 / NA / 88.7±0.7 / **91.7±0.6**	80.8±0.6 / NA / **83.9±0.6 / 84.3±0.7**
		TinyImageNet	85.2±0.6 / NA / 92.8±0.5 / **96.9±0.7**	77.1±0.6 / NA / 86.7±0.5 / **91.7±0.6**
		LSUN	85.8±0.6 / NA / 94.6±0.4 / **97.7±0.6**	77.3±0.6 / NA / 89.2±0.7 / **93.5±0.5**
	SVHN	CIFAR10	91.8±0.2 / NA / **98.6±0.2 / 98.7±0.3**	86.7±0.2 / NA / **95.7±0.3 / 96.1±0.3**
		TinyImageNet	94.8±0.2 / NA / **99.3±0.2 / 99.7±0.3**	90.2±0.2 / NA / 96.2±0.2 / **98.8±0.3**
		LSUN	94.0±0.2 / NA / **99.5±0.2 / 99.8±0.3**	89.1±0.2 / NA / 96.1±0.2 / **99.0±0.1**
ResNet	CIFAR10	SVHN	86.4±0.4 / **97.7±0.4 / 97.4±0.4** / 95.5±0.6	77.7±0.4 / NA / **91.8±0.4** / 89.0±0.5
		TinyImageNet	93.9±0.3 / 85.7±0.4 / 94.7±0.3 / **99.1±0.5**	86.1±0.3 / NA / 88.5±0.3 / **95.4±0.5**
		LSUN	93.4±0.4 / 85.9±0.4 / 96.8±0.3 / **99.5±0.3**	85.7±0.4 / NA / 91.3±0.3 / **97.3±0.4**
	CIFAR100	SVHN	72.1±0.6 / **91.1±0.6** / 85.2±0.6 / 84.3±0.5	67.8±0.4 / NA / **79.9±0.6** / 76.4±0.7
		TinyImageNet	83.7±0.6 / 75.3±0.5 / **92.4±0.5** / 87.7±0.6	75.7±0.5 / NA / **85.8±0.5** / 84.3±0.5
		LSUN	81.8±0.6 / 69.7±0.5 / **93.3±0.6** / 82.2±0.7	74.7±0.6 / NA / **87.6±0.5** / 79.6±0.6
	SVHN	CIFAR10	92.1±0.2 / 97.3±0.3 / **98.2±0.2** / 97.3±0.2	89.3±0.3 / NA / **94.3±0.2 / 94.5±0.3**
		TinyImageNet	92.8±0.2 / 97.6±0.2 / **98.8±0.1 / 99.0±0.3**	90.0±0.2 / NA / 94.6±0.3 / **98.7±0.2**
		LSUN	90.6±0.2 / **99.7±0.3** / 97.9±0.2 / **99.8±0.2**	88.3±0.2 / NA / 93.7±0.3 / **99.4±0.2**

ODIN uses input preprocessing, temperature calibration, and adversarial validation (hyperparameter tuning using adversarial examples). The Mahalanobis solution uses input preprocessing, adversarial validation, feature extraction, feature ensemble, and metric learning. Input preprocessing makes the inferences of ODIN and the Mahalanobis method at least three times slower and *at least three times less energy/computationally efficient than SoftMax or IsoMax inferences*. Feature ensembles may limit the Mahalanobis method scalability to deal with large-size images used in real-world applications. ACET uses adversarial training, which results in slower training, possibly reduced scalability for large-size images, and eventually, classification accuracy drop. The ODIN, the Mahalanobis approach, and ACET present hyperparameters that need to be validated for each combination of datasets and models presented in the table. Furthermore, considering that adversarial hyperparameters (e.g., the adversarial perturbation) used in ODIN/Mahalanobis/ACET were validated using the SVHN/CIFAR10/CIFAR100 validation sets and that these sets were reused as OOD detection test set, we conclude that the OOD detection performance reported by those papers, which we are reproducing in this table, may be overestimated. ODIN/Mahalanobis/ACET results were obtained using SoftMax loss rather than IsoMax loss as baseline. No outlier/extra/background data is used. IsoMax+ES means training with IsoMax loss and performing OOD detection using the entropic score. IsoMax+ES does not use these techniques and does not have such requirements, side effects, and hyperparameters to tune (as IsoMax+ES works as a baseline OOD detection approach, those techniques may be incorporated in future research). The OOD detection test sets are composed of images from both in-data and out-data (see Section IV for further details). The results represent the mean and standard deviation of five executions. The best values are bold when they overcome the competing approach value outside the margin of error given by the standard deviations.

TABLE VI
OUT-OF-DISTRIBUTION DETECTION: INFERENCE DELAYS / PRESUMED COMPUTATIONAL COST AND ENERGY CONSUMPTION RATES.

Model	In-Data (training)	Hardware (inference)	Out-of-Distribution Detection: Inference Delays / Presumed Computational Cost and Energy Consumption Rates.		
			SoftMax Loss [8] (current baseline) MPS (ms) [↓] / ES (ms) [↓]	IsoMax Loss (ours) (proposed baseline) MPS (ms) [↓] / ES (ms) [↓]	Input Preprocessing: ODIN [1], Mahalanobis [2], Generalized ODIN [18] (ms) [↓]
DenseNet	CIFAR10	CPU	18.1 / 19.4	18.0 / 19.2	242.4 (≈ **10x slower)**
		GPU	11.6 / 13.0	11.6 / 11.5	39.2 (≈ **4x slower)**
	CIFAR100	CPU	18.4 / 19.8	18.4 / 19.3	261.0 (≈ **10x slower)**
		GPU	12.9 / 11.4	11.8 / 11.5	39.6 (≈ **4x slower)**
	SVHN	CPU	18.1 / 18.6	18.3 / 18.6	241.5 (≈ **10x slower)**
		GPU	11.6 / 11.9	11.7 / 11.6	39.6 (≈ **4x slower)**
ResNet	CIFAR10	CPU	22.3 / 23.2	23.0 / 23.5	250.4 (≈ **10x slower)**
		GPU	4.5 / 3.8	4.2 / 4.1	15.4 (≈ **4x slower)**
	CIFAR100	CPU	23.3 / 23.1	23.3 / 23.8	252.6 (≈ **10x slower)**
		GPU	4.3 / 3.9	4.3 / 4.2	14.8 (≈ **4x slower)**
	SVHN	CPU	23.1 / 23.4	23.4 / 23.3	263.8 (≈ **10x slower)**
		GPU	4.2 / 4.0	4.0 / 4.0	15.7 (≈ **4x slower)**

MPS means maximum probability score. ES means entropic score. For SoftMax and IsoMax losses (baseline OOD detection approaches), the inference delays combine both classification and detection computation. For the methods based on input preprocessing, the inference delays represent only the input preprocessing phase. All values are in milliseconds. In addition to presenting similar classification accuracy (Table III, Supplementary Material E) and much better OOD detection performance (Table II), IsoMax loss trained networks produce inferences as fast as SoftMax trained networks. Moreover, the entropic score is as fast as the maximum probability score. Using CPU (Intel i7-4790K, 4.00GHz, x64, octa-core) for inference (the case more relevant from a cost point of view), methods based on input preprocessing are *about ten times slower* than the baseline approaches. Using GPU (Nvidia GTX 1080 Ti) for inference, our approach is *about four times faster* than the methods based on input preprocessing. The inference delay rates presumably reflect in similar the computational cost and energy consumption rates. The inference delays presented are the mean value of the inference delay of each image calculated (batch size equals one) over the entire dataset. The standard deviation was below 0.3 for all cases.

B. Fair Comparison: SoftMax versus IsoMax

Table II shows that models trained with the SoftMax loss using the maximum probability as the score (SoftMax+MPS) almost always present the worst OOD detection performance.

In the case of models trained with SoftMax loss, replacing the maximum probability score by the entropic score (SoftMax+ES) produces small OOD detection performance gains. However, the combination of models trained using IsoMax loss with the entropic score (IsoMax+ES), which is the proposed solution, significantly improves, usually by several percentage points, the OOD detection performance across almost all datasets, models, out-distributions, and metrics. *We emphasize that the entropic score only produces high OOD detection performance when the probability distributions present high entropy (IsoMax+ES). For low entropy probability distributions (SoftMax+ES), the performance increase is minimal.* For a study showing that the IsoMax loss superiority in relation to SoftMax is robust for a different number of training examples per class, see Supplementary Material E.

The IsoMax loss is well-positioned to replace SoftMax loss as a novel baseline OOD detection approach for neural networks OOD detection, as the former does not present an accuracy drop in relation to the latter and simultaneously improves the OOD detection performance. Additional techniques (e.g., input preprocessing, adversarial training, and outlier exposure) may be added to improve OOD detection performance gains further.

Table III shows that IsoMax loss consistently produces classification accuracy similar to SoftMax loss, regardless of being used as baseline approach or combined with additional techniques to improve OOD detection performance. The Supplementary Material E presents further evidence that IsoMax loss does not present classification accuracy drop in comparison with SoftMax loss.

The IsoMax loss enhanced versions almost always outperform the OOD detection performance of the corresponding SoftMax loss enhanced version when both are using the same add-on technique. The major drawback of adding label smoothing or center loss regularization is the need to tune the hyperparameters presented by these add-on techniques. Additionally, different hyperparameters values need to be validated for each pair of in-distribution and out-distribution. As mentioned before, it is highly optimistic to assume access to OOD data, as we usually do not know what OOD data the solution we will face in the field. *Even considering this best-case scenario, SoftMax loss combined with label smoothing or center loss regularization always presented significantly lower OOD detection performance than IsoMax loss without using them and, consequently, avoiding unrealistically optimist access to OOD data and the mentioned validations.* Enhancing SoftMax loss or IsoMax loss with ODIN presents the same problems from a practical perspective. *In CIFAR100, SoftMax loss with outlier exposure produces lower performance than IsoMax loss without it for both DenseNet and ResNet.*

The Table IV presents results for text data. As expected, the results show that our approach is *domain-agnostic* and therefore may be applied to data other than images.

C. Unfair Comparison: SoftMax versus No Seamless Solutions

Table V presents a perspective about how our proposed baseline OOD detection approach compares with no seamless solutions. Hence, we need to analyze the mentioned table considering that it shows an *unfair* comparison of approaches that present different requirements and side effects. ODIN and the Mahalanobis solution use input preprocessing[8].

Consequently, they present solutions with much slower and less energy-efficient inferences than models trained with IsoMax loss, which are as fast and computation-efficient as the models trained with common SoftMax loss. Moreover, they also require validation using adversarial samples.

Additionally, ODIN requires temperature calibration, while the Mahalanobis approach uses feature ensemble and metric learning, which may be implicated in limited scalability for large-size images. ACET requires adversarial training, which may also prevent its use in large-size images. ODIN, the Mahalanobis approach, and ACET have hyperparameters tuned for each combination of in-data and models in the table. ODIN, the Mahalanobis method, and ACET used SoftMax loss rather than IsoMax loss. IsoMax+ES exhibits neither the mentioned unfortunate requirements nor undesired side effects.

Additionally, taking into consideration that ODIN, Mahalanobis, and ACET used adversarial hyperparameters (e.g., the adversarial perturbation) that were validated using the validation sets of SVHN/CIFAR10/CIFAR100 and noticing that these sets composed the OOD detection test sets, we conclude that some overestimation may be present in the OOD detection performance reported by these papers.

Regardless of the previous considerations, Table V shows that IsoMax+ES considerably outperforms ODIN in all evaluated scenarios. Therefore, in addition to avoiding hyperparameter tuning and access to OOD or adversarial samples, the results show that removing the entropic scale is much more effective in increasing the OOD detection performance than performing temperature calibration.

Furthermore, IsoMax+ES usually outperforms ACET by a large margin. Moreover, in most cases, even operating under much more favorable conditions, the Mahalanobis method surpasses IsoMax+ES by less than 2%. In some scenarios, the latter overcomes the former despite avoiding hyperparameter validation, being seamless and producing much faster and more computation-efficient inferences, as no input preprocessing technique is required.

D. Inference Efficiency

Table VI presents the inference delays for SoftMax loss, IsoMax loss, and competing methods using CPU and GPU. We observe that neural networks trained using IsoMax loss produce inferences equally fast as the ones produced by networks trained using SoftMax loss, regardless of using CPU or GPU for inference. Additionally, the entropic score is as fast as the usual maximum probability score. Moreover, methods based on input preprocessing were more than ten times slower on CPU and about four times slower on GPU. Finally, those

[8]To allow OOD detection, each inference requires a first neural network forward pass, a backpropagation, and a second forward pass.

rates presumably apply to the computational cost and energy consumption as well.

At first sight, inference methods (i.e., methods that can be applied to pre-trained models) may be seen as "low cost" compared with training methods like our IsoMax loss, as we avoid training or fine-tuning the neural network. However, this conclusion may be misleading, as we have to keep in mind that inference methods (e.g., ODIN [1] and Mahalanobis [2]) produce inferences that are much more energy, computation, and time inefficient (Table VI). For example, consider initially the rare practical situation where a pretrained model is available, and no fine-tuning to a custom dataset is required. In such cases, an inference method may indeed be applied without requiring any loss function. However, despite avoiding training or fine-tuning a neural network once or a few times, all the subsequent inferences, which are usually performed thousands or millions of times on the field (sometimes even by constrained devices), will be about 6 to 10 times more computational, energy, environment, and time inefficient (Table VI).

Alternatively, consider the case where a pretrained model is not available or fine-tuning to a custom dataset is needed. In this situation, which is much more likely in practice, we cannot avoid training or fine-tuning the neural network, and a training method like ours will be required anyway. In these cases, it would be recommended to train or fine-tune using IsoMax rather than SoftMax loss, as our experiments showed that both training times are the same and IsoMax loss produces considerably higher OOD detection performance.

Hence, inference methods are more inference inefficient because they coexist with a model that was trained with a loss not designed from the start with OOD detection in mind. The drawback is to produce an enormous amount of inefficient inferences on the field that usually uses constrained computational resource devices such as embedded systems.

Nevertheless, suppose the increased inference time, computation, environment damage, and energy consumption required to use an inference method is not a concern from a practical point of view. In that case, the model pretrained or fine-tuned using a training method may be subsequently subjected to the desired inference approach to increase overall OOD detection performance further. In other words, we do not claim that inference methods such as ODIN and Mahalanobis do not increase the OOD detection performance compared with Maximum Score Probability or even the Entropic Score. However, we point out that producing more energy inefficient inferences is one drawback of adopting such inference methods.

In summary, rather than concurrent, the training and inference methods are orthogonal and complementary. Moreover, we see no reason not to train the models with a loss designed to support OOD, regardless of subsequently applying an OOD inference method.

E. Additional Studies

For additional analyses regarding logits, probabilities, and entropies, please see Supplementary Material F. For information regarding the similarity of the SoftMax loss and the IsoMax loss training metrics, see the Supplementary Material G.

V. Conclusion

We proposed a baseline OOD detection approach based on the maximum entropy principle. The proposed IsoMax loss works as a SoftMax loss drop-in replacement that produces accurate predictions in addition to inferences that are fast and energy- and computation-efficient. OOD detection is performed using the rapid entropic score. Furthermore, it is also turnkey, as no additional procedures other than a straightforward neural network training is required, and no hyperparameters are tuned. Neither is required collection of extra/outlier data.

The direct replacement of the SoftMax loss by the IsoMax loss significantly improves the baseline neural networks' OOD detection performance. Hence, we conclude that the general low OOD detection performance of current deep networks is due to SoftMax loss drawbacks, i.e., anisotropy and overconfidence, rather than the models' limitations.

Our OOD detection approach produces high OOD detection performance without relying on extra techniques (e.g., input preprocessing, adversarial training, hyperparameter validation, feature ensemble, additional/outlier/background data, and others) that, unfortunately, present inconvenient requirements and undesired side effects. Nevertheless, when the cited requirements and side effects are not a concern for a particular real-world application, the mentioned (or novel) techniques may be combined with IsoMax loss to possibly achieve higher OOD detection performance. This combined solution may be particularly needed for more challenging large-size image datasets such as ImageNet.

Summarily, the three most original and relevant contributions of this paper are the following: First, the theoretical arguments, intuitions, and insights that motivate neural network design with high entropy outputs. Second, "the entropy maximization trick" that allows it. Third, the experimental demonstration that the entropy of neural networks with high entropy posterior probability distributions represents a high-performance OOD detection score.

We intend to apply our OOD detection approach to models using large-size images used in large-scale real-world applications in future works, as we believe our approach scales to these cases as well. As the principle of maximum entropy is a general concept, we also plan to adapt our approach to machine learning areas other than deep learning. Another promising approach could be using recent data augmentation techniques [68], [69] or strategies based on pretrained models [43]. Additionally, despite requiring auxiliary/outlier/background data, to further increase the OOD detection performance of our solution, we plan to incorporate loss enhancement (regularization) techniques such as outlier exposure [26], [27], background samples [28] and energy-based training and score techniques [29].

References

[1] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *International Conference on Learning Representations*, 2018.

[2] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," *Neural Information Processing Systems*, 2018.

[3] M. Hein, M. Andriushchenko, and J. Bitterwolf, “Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem,” *IEEE International Conference on Computer Vision and Pattern Recognition*, 2018.

[4] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult, “Towards open set recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757–1772, 2013.

[5] W. J. Scheirer, L. P. Jain, and T. E. Boult, “Probability models for open set recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2317–2324, 2014.

[6] A. Bendale and T. Boult, “Towards open world recognition,” *IEEE International Conference on Computer Vision and Pattern Recognition*, 2015.

[7] E. Rudd, L. P. Jain, W. J. Scheirer, and T. Boult, “The extreme value machine,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 762–768, 2018.

[8] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” *International Conference on Learning Representations*, 2017.

[9] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, “Deep learning for anomaly detection: A review,” *ACM Comput. Surv.*, vol. 54, no. 2, 2021.

[10] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A discriminative feature learning approach for deep face recognition,” *European Conference on Computer Vision*, 2016.

[11] W. Liu, Y. Wen, Z. Yu, and M. Yang, “Large-margin softmax loss for convolutional neural networks.” *International Conference on Machine Learning*, 2016.

[12] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” *International Conference on Machine Learning*, 2017.

[13] E. T. Jaynes, “Information theory and statistical mechanics,” *Physical Review*, vol. 106, pp. 620–630, 1957.

[14] ——, “Information theory and statistical mechanics. II,” *Physical Review*, vol. 108, pp. 171–190, 1957.

[15] T. M. Cover and J. A. Thomas, “Elements of information theory,” *Wiley Series in Telecommunications and Signal Processing*, 2006.

[16] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, “The computational limits of deep learning,” *CoRR*, vol. abs/2007.05558, 2020.

[17] E. Techapanurak, M. Suganuma, and T. Okatani, “Hyperparameter-free out-of-distribution detection using cosine similarity,” *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020.

[18] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, “Generalized ODIN: Detecting out-of-distribution image without learning from out-of-distribution data,” *IEEE International Conference on Computer Vision and Pattern Recognition*, 2020.

[19] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. J. Goodfellow, A. Madry, and A. Kurakin, “On evaluating adversarial robustness,” *CoRR*, vol. abs/1902.06705, 2019.

[20] E. Wong, L. Rice, and J. Z. Kolter, “Fast is better than free: Revisiting adversarial training,” *International Conference on Learning Representations*, 2020.

[21] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. P. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, “Adversarial training for free!” *Neural Information Processing Systems*, 2019.

[22] A. Raghunathan, S. M. Xie, F. Yang, J. C. Duchi, and P. Liang, “Adversarial training can hurt generalization,” *CoRR*, vol. abs/1906.06032, 2019.

[23] Q. Yu and K. Aizawa, “Unsupervised out-of-distribution detection by maximum classifier discrepancy,” *International Conference on Computer Vision*, 2019.

[24] A. Vyas, N. Jammalamadaka, X. Zhu, D. Das, B. Kaul, and T. L. Willke, “Out-of-distribution detection using an ensemble of self supervised leave-out classifiers,” *European Conference on Computer Vision*, 2018.

[25] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” *Neural Information Processing Systems*, 2017.

[26] D. Hendrycks, M. Mazeika, and T. Dietterich, “Deep anomaly detection with outlier exposure,” *International Conference on Learning Representations*, 2019.

[27] A.-A. Papadopoulos, M. R. Rajati, N. Shaikh, and J. Wang, “Outlier exposure with confidence control for out-of-distribution detection,” *CoRR*, vol. abs/1906.03509, 2019.

[28] A. R. Dhamija, M. Günther, and T. Boult, “Reducing network agnostophobia,” *Neural Information Processing Systems*, 2018.

[29] W. Liu, X. Wang, J. D. Owens, and Y. Li, “Energy-based out-of-distribution detection,” *CoRR*, vol. abs/2010.03759, 2020.

[30] A. Kendall and Y. Gal, “What uncertainties do we need in bayesian deep learning for computer vision?” *Neural Information Processing Systems*, 2017.

[31] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl, “Leveraging uncertainty information from deep neural networks for disease detection,” *Scientific Reports*, vol. 7, 2017.

[32] A. Malinin and M. Gales, “Predictive uncertainty estimation via prior networks,” *Neural Information Processing Systems*, 2018.

[33] V. Kuleshov, N. Fenner, and S. Ermon, “Accurate uncertainties for deep learning using calibrated regression,” *International Conference on Machine Learning*, 2018.

[34] A. Subramanya, S. Srinivas, and R. V. Babu, “Confidence estimation in deep neural networks via density modelling,” *CoRR*, vol. abs/1707.07013, 2017.

[35] A. Shafaei, M. Schmidt, and J. J. Little, “A less biased evaluation of out-of-distribution sample detectors,” *British Machine Vision Conference*, 2019.

[36] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green AI,” *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[37] D. Macêdo, T. I. Ren, C. Zanchettin, A. L. I. Oliveira, and T. B. Ludermir, “Entropic out-of-distribution detection,” *International Joint Conference on Neural Networks*, 2021.

[38] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” *Neural Information Processing Systems*, 2014.

[39] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” *IEEE International Conference on Computer Vision and Pattern Recognition*, 2015.

[40] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical networks for few-shot learning,” *Neural Information Processing Systems*, 2017.

[41] T. DeVries and G. W. Taylor, “Learning confidence for out-of-distribution detection in neural networks,” *CoRR*, vol. abs/1802.04865, 2018.

[42] J. Tack, S. Mo, J. Jeong, and J. Shin, “CSI: novelty detection via contrastive learning on distributionally shifted instances,” *Neural Information Processing Systems*, 2020.

[43] C. S. Sastry and S. Oore, “Detecting out-of-distribution examples with gram matrices,” *International Conference on Machine Learning*, vol. 119, pp. 8491–8501, 2020.

[44] A. Dubey, O. Gupta, R. Raskar, and N. Naik, “Maximum-entropy fine grained classification,” *Neural Information Processing Systems*, 2018.

[45] G. Pereyra, G. Tucker, J. Chorowski, Łukasz Kaiser, and G. Hinton, “Regularizing neural networks by penalizing confident output distributions,” *International Conference on Learning Representations*, 2017.

[46] D. Miller, A. Rao, K. Rose, and A. Gersho, “A maximum entropy approach for optimal statistical classification,” *IEEE Workshop on Neural Networks for Signal Processing*, 1995.

[47] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra, “A maximum entropy approach to natural language processing,” *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[48] J. Shawe-Taylor and D. Hardoon, “Pac-bayes analysis of maximum entropy classification,” *International Conference on Artificial Intelligence and Statistics*, 2009.

[49] E. T. Jaynes, “The relation of bayesian and maximum entropy methods,” *Maximum-Entropy and Bayesian Methods in Science and Engineering: Foundations*, pp. 25–29, 1988.

[50] J. Pearl, “Probabilistic reasoning in intelligent systems - networks of plausible inference,” *Morgan Kaufmann*, 1989.

[51] J. Williamson, “Objective bayesian nets,” *We Will Show Them! Essays in Honour of Dov Gabbay, Volume Two*, pp. 713–730, 2005.

[52] ——, “Philosophies of probability,” *Philosophy of Mathematics: Handbook of the Philosophy of Science*, 2009.

[53] ——, “In defence of objective bayesianism,” *Oxford University Press*, vol. 23, no. 2, pp. 255–258, 2013.

[54] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” *International Conference on Artificial Intelligence and Statistics*, 2010.

[55] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” *International Conference on Computer Vision*, 2016.

[56] F. Wang, J. Cheng, W. Liu, and H. Liu, “Additive margin softmax for face verification,” *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[57] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “ArcFace: Additive angular margin loss for deep face recognition,” *IEEE International Conference on Computer Vision and Pattern Recognition*, 2019.

[58] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

[59] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[60] D. Lee and Y. Cheon, "Soft labeling affects out-of-distribution detection of deep neural networks," *CoRR*, vol. abs/2007.03212, 2020.

[61] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *IEEE International Conference on Computer Vision and Pattern Recognition*, 2017.

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *European Conference on Computer Vision*, 2016.

[63] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Science Department, University of Toronto*, 2009.

[64] Y. Netzer and T. Wang, "Reading digits in natural images with unsupervised feature learning," *Neural Information Processing Systems*, 2011.

[65] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "ImageNet: A large-scale hierarchical image database," *IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.

[66] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "LSUN: construction of a large-scale image dataset using deep learning with humans in the loop," *CoRR*, vol. abs/1506.03365, 2015.

[67] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *Empirical Methods in Natural Language Processing*, 2014.

[68] S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," *Neural Information Processing Systems*, 2019.

[69] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," *International Conference on Computer Vision*, 2019.

[70] T. Mensink, J. J. Verbeek, F. Perronnin, and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2624–2637, 2013.

[71] K. Musgrave, S. J. Belongie, and S. Lim, "A metric learning reality check," *CoRR*, vol. abs/2003.08505, 2020.

[72] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

David Macêdo received his M.Sc. Computer Science degree and B.Sc. Electronic Engineer degree summa cum laude from the Universidade Federal de Pernambuco (UFPE), Brazil. He was a visiting researcher at the Montreal Institute for Learning Algorithms (MILA), Université de Montréal (UdeM), Canada. He co-created and is currently a collaborator professor of the Deep Learning course of the Computer Science Master and Doctorate Programs at the Center for Informatics (CIn), Universidade Federal de Pernambuco (UFPE), Brazil. He is a professor at Faculdade Nova Roma, Brazil. He has authored dozens of deep learning papers published in international peer-reviewed journals and conferences and is reviewer of ICLR, ICML, and IEEE. He is pursuing a Ph.D. degree in computer science at UFPE. His research interests include Deep Learning, Trustworthy Artificial Intelligence, Computer Vision, Natural Language Processing, and Speech Processing.

Tsang Ing Ren received the B.Sc. degree in electronic engineering from the Universidade Federal de Pernambuco, Recife, Brazil, and the Ph.D. degree in physics from the University of Antwerp, Antwerp, Belgium. He is currently an associate professor with the Center for Informatics, Universidade Federal de Pernambuco. His research interests include Machine Learning, Image Processing, Computational Photography, Computer Vision, and Deep Learning.

Cleber Zanchettin is an Associate Professor with the Center for Informatics, Federal University of Pernambuco, Brazil, and Visiting Associate Professor at Northwestern University, EUA. Dr. Zanchettin received the D.Sc. degree in Computer Science from the Center for Informatics, Federal University of Pernambuco, Brazil in 2008. He has authored more than 100 scientific publications in journals and conferences and one edited book. His research interests include pattern recognition, machine learning, deep learning, and image analysis.

Adriano L. I. Oliveira obtained his B.Sc. degree in electrical engineering and M.Sc. and Ph.D. degrees in computer science from the Universidade Federal de Pernambuco, Brazil, in 1993, 1997 and 2004, respectively. In 2011, he joined the Center for Informatics at the Universidade Federal de Pernambuco as an assistant professor. He was a visiting professor at École de technologie supérieure, Université du Quebec, Montreal, Canadá, from 2018 to 2019. He was an assistant professor at the Universidade Federal Rural de Pernambuco from 2009 to 2011 and at the Pernambuco State University from 2002 to 2009. He has published over 110 articles in scientific journals and conferences and one book. He is a senior member of the IEEE. His current research interests include neural networks, machine learning, pattern recognition, data mining, and applications of these techniques to time series analysis and forecasting, information systems, software engineering, and biomedicine.

Teresa Ludermir received her Ph.D. degree in artificial neural networks from the Imperial College London, London, U.K., in 1990. She was a lecturer with Kings College London, London, from 1991 to 1992. She joined the Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil, in 1992, where she is currently a professor and the Head of the Computational Intelligence Group. She has authored over 300 articles in scientific journals and conferences and three books on neural networks. Her current research interests include weightless neural networks, hybrid neural systems, and applications of neural networks. Ludermir organized two of the Brazilian Symposiums on Neural Networks.

Supplementary Material

Appendix A
SoftMax Loss Anisotropy

Let $\boldsymbol{x}$ represent the input applied to a neural network and $\boldsymbol{f_\theta}(\boldsymbol{x})$ represent the high-level feature vector produced by it. For this work, the underlying structure of the network does not matter. Considering k to be the correct class for a particular training example $\boldsymbol{x}$, we can write the SoftMax loss associated with this specific training sample as:

$$\mathcal{L}_S(\hat{y}^{(k)}|\boldsymbol{x}) = -\log\left(\frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{f_\theta}(\boldsymbol{x})+b_k)}{\sum\limits_j \exp(\boldsymbol{w}_j^\top \boldsymbol{f_\theta}(\boldsymbol{x})+b_j)}\right) \tag{13}$$

In Equation (13), $\boldsymbol{w}_j$ and b_j represent the weights and biases associated with class j, respectively. The sum has N terms, where N is the number of classes. From a geometric perspective, the term $\boldsymbol{w}_j^\top \boldsymbol{f_\theta}(\boldsymbol{x})+b_j$ represents a hyperplane in the high-level feature space. It divides the feature space into two subspaces that we call positive and negative subspaces. The deeper inside the positive subspace the feature vector $\boldsymbol{f_\theta}(\boldsymbol{x})$ of a particular example is located, the more likely the example is believed to belong to the considered class.

Therefore, training neural networks using SoftMax loss does not incentivize the agglomeration of representations of examples associated with a particular class into a limited region of the hyperspace, as it produces *separable features* rather than *discriminative features* [10, Fig. 1]. The immediate consequence is the propensity of neural networks trained with SoftMax loss to make high confidence predictions on examples that stay in regions far away from the training examples, which explains their unsatisfactory OOD detection performance [3]. Indeed, the SoftMax loss is based on affine transformations, which are essentially internal products. Consequently, the last layer representations of such networks tend to align in the direction of the weight vector, producing locally preferential directions in space and subsequently anisotropy.

The SoftMax loss anisotropy is usually corrected by using metric learning on neural network pretrained features [2], [4]–[7], [70]. For example, the high OOD detection performance of the Mahalanobis approach [2] indicates that deploying locally isotropic spaces around class prototypes improves the OOD detection performance. In such solutions, a mapping from the extracted features to a novel embedding space is constructed, and class prototypes are produced. The distance may be predefined (e.g., Euclidean distance) or learned (e.g., Mahalanobis distance).

However, approaches based on feature extraction and metric learning present drawbacks [71]. First, they are not turnkey, as additional procedures are required after neural network training. Additionally, they usually present hyperparameters to tune, typically requiring unrealistic access to design-time OOD or adversarial samples. *Therefore, a possible option to build a seamless approach to OOD detection is to design an isotropic (exclusively distance-based) loss that works as a SoftMax loss drop-in replacement.*

Appendix B
Nonsquared Euclidean Distance

We need to choose a distance that allows IsoMax to work as a SoftMax drop-in replacement. Hence, the loss needs to learn *both* high-level features *and prototypes* using *exclusively* SGD and *end-to-end* backpropagation, as *no additional offline procedures are allowed*. We also require the training using IsoMax loss to be *as consistent and stable as the typical SoftMax loss neural network training*.

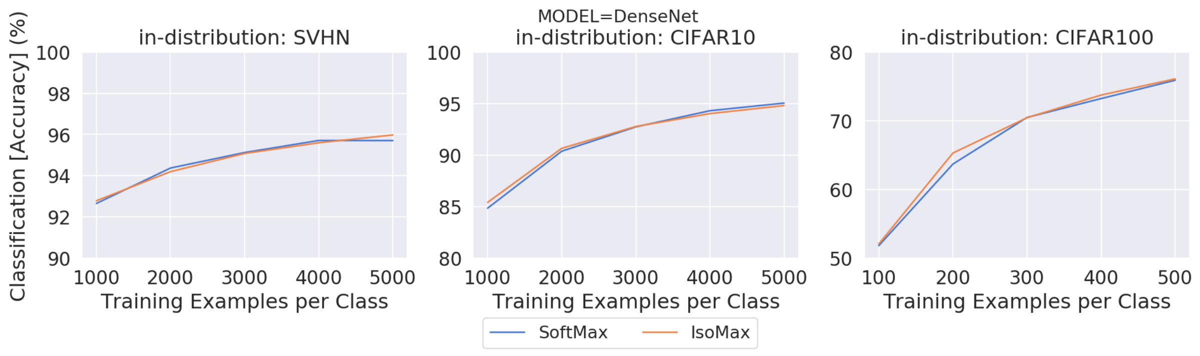
The covariance matrix makes it hard to use the Mahalanobis distance to train a neural network directly. Therefore, we decide to use *Euclidean* distance. We have reasons to prefer the *nonsquared* Euclidean distance rather than the *squared* Euclidean distance. First, the *nonsquared* Euclidean distance is a real metric that obeys the Cauchy–Schwarz inequality while the *nonsquared* Euclidean distance is not. Using a metric that follows the Cauchy–Schwarz inequality is essential because of our previous geometric considerations. Additionally, using the *squared* Euclidean distance is actually equivalent to using a linear model with a particular parameterization [40], [70], which we prefer to avoid to increase the representative power of the proposed loss.

Moreover, we experimentally observed that training neural networks using *exclusively* SGD and *end-to-end* backpropagation is more stable and consistent when using *nonsquared* Euclidean distance rather than *squared* Euclidean distance. Indeed, *squared* Euclidean distance-based logits are harder to *seamlessly* train than *nonsquared* Euclidean distance-based logits because numeric calculus instabilities are much more likely to occur when performing calculations and derivations with values of the order of $\mathcal{O}(e^{-d^2})$ than $\mathcal{O}(e^{-d})$. Finally, even in the cases in which we were eventually able to *seamlessly* train neural networks using the *squared* Euclidean distance, we observed lower OOD detection performance than using the *nonsquared* Euclidean distance-based alternative. Therefore, we choose the *nonsquared* Euclidean distance.

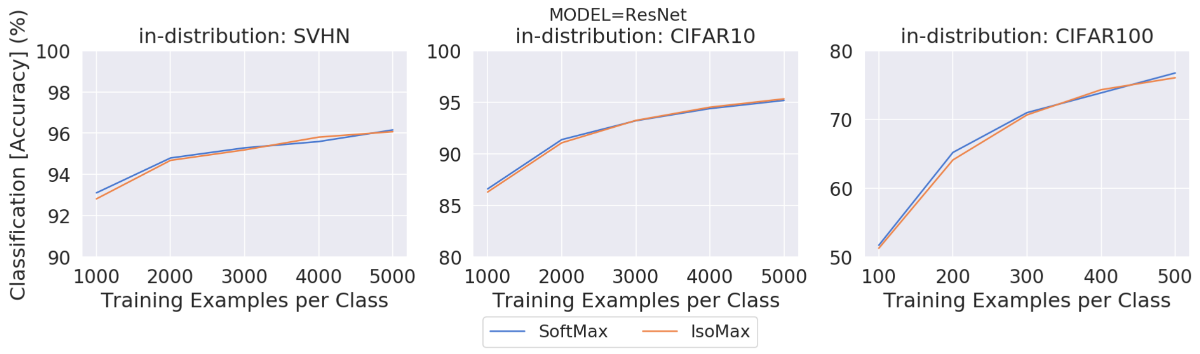
Appendix C
Entropic Score Discussion

Out-of-distribution detection approaches typically define a score to be used after inference to evaluate whether an example should be considered OOD. In a seminal work, [72] demonstrated that the entropy presents the optimum measure of the randomness of a source of symbols. More broadly, we understand entropy as a measure of the uncertainty we have about a random variable. Considering that the uncertainty in classifying a specific sample should be an optimum metric to evaluate whether a particular example is OOD.

During IsoMax training, the embedding-prototype distances are affected. On the one hand, the distances from embeddings to the correct class prototype are reduced to increase classification accuracy. On the other hand, the distances from embeddings to the wrong class prototypes are increased. Consequently, based on the Equation (10), the probabilities of in-distribution examples increase. Therefore, it is reasonable to expect that samples with lower entropy more likely belong to the in-distribution.



(a)



(b)

Fig. 3. IsoMax loss presents test accuracy similar to SoftMax loss (no classification accuracy drop) for different numbers of training examples per class on several datasets and models. Simultaneously, IsoMax usually produces a much higher OOD detection performance (Table II).

From a practical perspective, using this predefined mathematical expression as score avoids the need to train an additional regression model to detect OOD samples that is otherwise required, for example, in the Mahalanobis approach.

Even more important, since no regression model needs to be trained, there is no need for unrealistic access to OOD or adversarial samples for hyperparameter validation. Since the entropic score is a predefined mathematical expression rather than a trainable model, it is available as soon as the neural network training finishes, avoiding additional training of extra models in a post-processing phase.

Appendix D
Differentiation from the Scaled Cosine Approach

An approach based on cosine similarity is proposed in [17]. However, different from IsoMax loss, this solution presents *significant classification accuracy drop* (the authors even suggest using two models, one for classification and the other for OOD detection) and increases the total number of parameters by requiring an additional layer.

Additionally, it is not an isotropic loss, as the cosine similarity is used rather than a distance. Moreover, the authors explicitly say that they do not have an explanation of why their solution works. Finally, it does not work as a SoftMax loss drop-in replacement, as we need to change the optimizer to avoid applying weight decay to the last layer.

Appendix E
Robustness Study

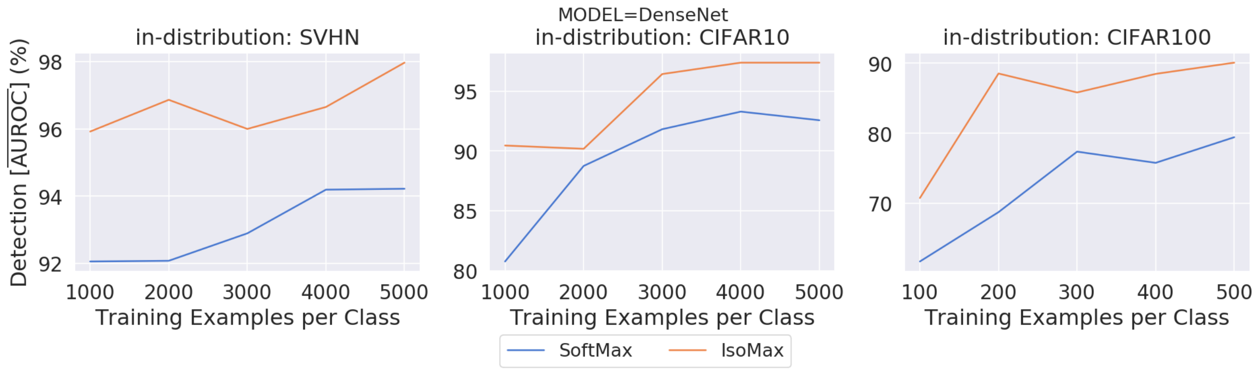
Fig. 3 shows that the classification accuracy of models trained with the IsoMax loss is similar to the models trained with the SoftMax loss regardless of the dataset or architecture. This is also true for a varying number of training examples per class.

Fig. 4 presents the OOD detection performance of SoftMax and IsoMax losses in many models (DenseNet and ResNet), and datasets (SVHN, CIFAR10, and CIFAR100) for a range of training samples per class. For each in-distribution, the AUROC and TNR@TPR95 results were averaged over all possible combinations of OOD detection test sets. The entropic score was used in all cases. It shows that IsoMax loss consistently and notably overcomes the OOD detection performance of SoftMax loss for virtually all combinations of datasets, training examples per class, metrics, and models.

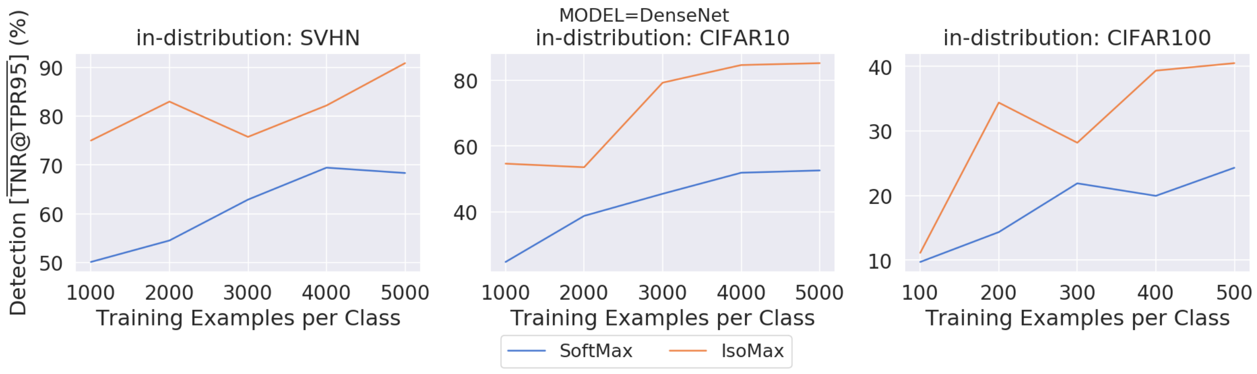
Appendix F
Additional Analyses

Fig. 5 shows that the in-distribution *interclass* logits are more distinguishable from out-distribution logits when using IsoMax loss, which explains its increased OOD detection performance compared with the SoftMax loss since there are far more *interclass* logits than *intraclass* logits. Distances are calculated from class prototypes.

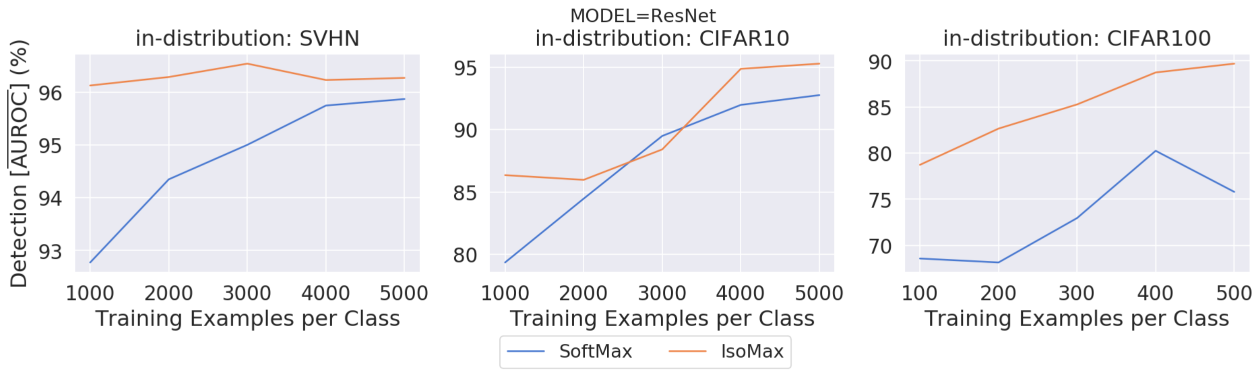
The unimodal nature of the out-distribution can also help to explain how IsoMax can distinguish between in-distributions



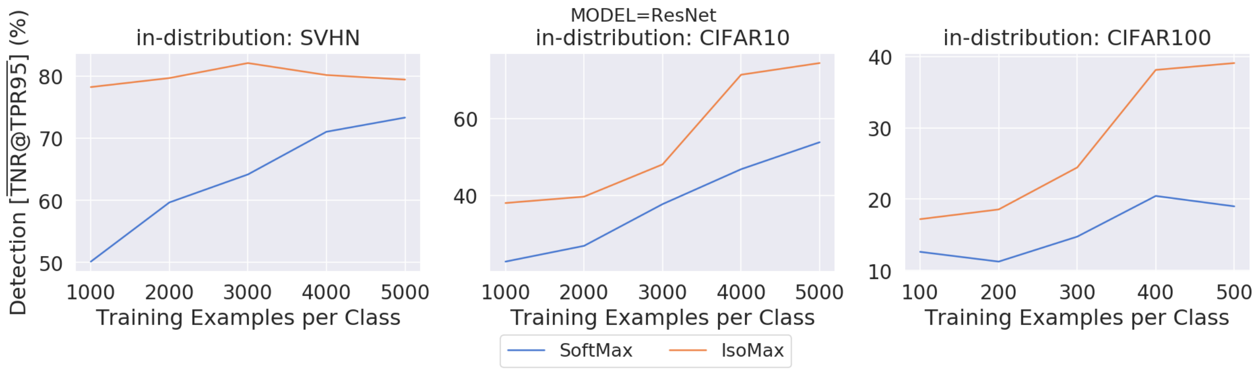
(a)



(b)

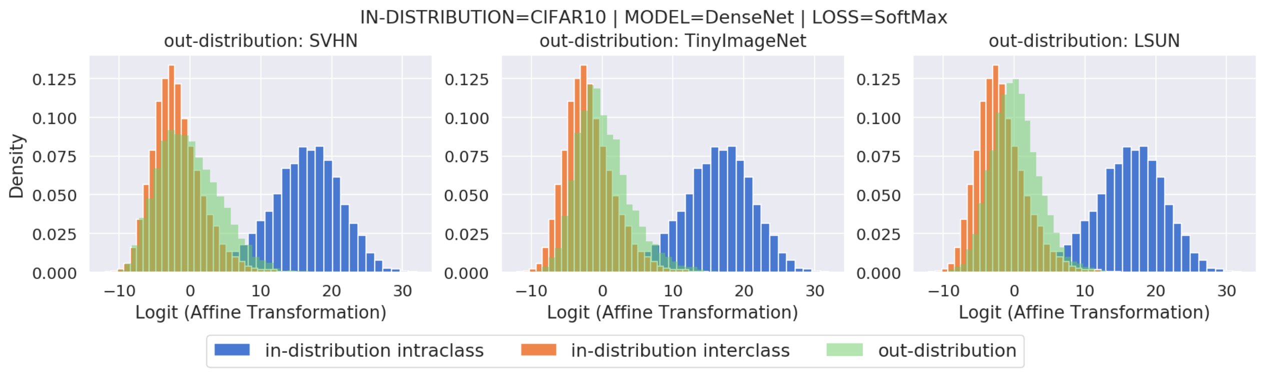


(c)

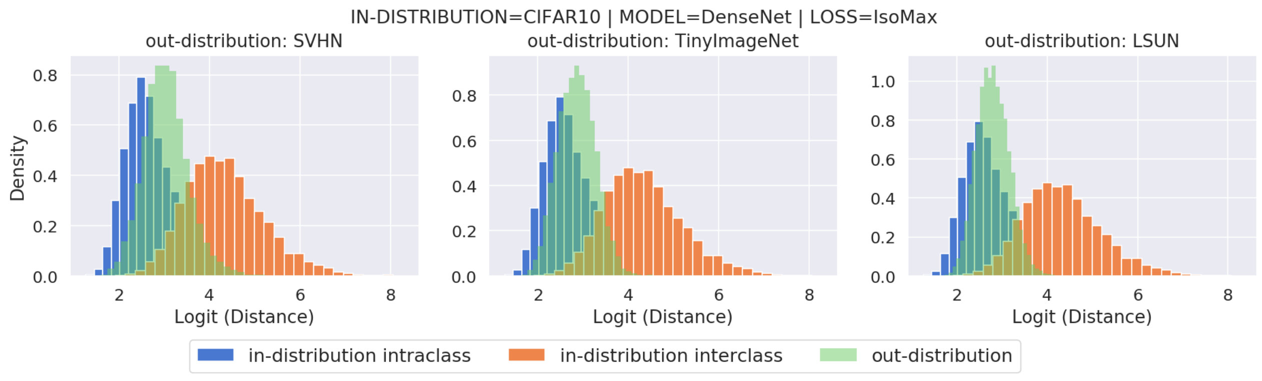


(d)

Fig. 4. Robustness study: IsoMax loss consistently presents higher OOD detection performance than does SoftMax loss for different numbers of training examples per class on several datasets, models, and metrics. The entropic score was used for both SoftMax and IsoMax losses. For each in-distribution, we calculated the mean AUROC and TNR@TPR95 considering all possible OOD detection test sets for extra details).



(a)



(b)

Fig. 5. Logits: (a) In SoftMax loss, out-distribution logits mimic in-distribution interclass logits. (b) In IsoMax loss, out-distribution logits mimic in-distribution intraclass logits, which facilitates OOD detection, as there are many more interclass logits than intraclass logits, and the entropic score takes into consideration the information provided by all network outputs rather than just one. Distances are calculated from class prototypes.

and out-distributions. Indeed, for out-distributions examples, we do not distinguish between intraclass and interclass logits, as there is no such thing that we can call the correct class in such cases. Therefore, on the one hand, we can see that the out-distributions logits are all very closed clustered (green distributions), producing even higher entropies than in-distributions examples. On the other hand, intraclass logits (blue distributions) are apart from interclass logits (orange distributions), generating high but slightly smaller entropies.

SoftMax intraclass and interclass logits are much farther apart than the IsoMax intraclass and interclass logits. Consequently, in disagreement with the maximum entropy principle, the former produces much lower entropies than the latter for in-distribution data.

Fig. 6 shows that networks trained with SoftMax loss exhibit high maximum probabilities. Sometimes this is true even for OOD samples. For networks trained with IsoMax loss, OOD samples usually present lower maximum probabilities compared with in-distribution samples. Furthermore, it also shows that the networks trained with SoftMax loss are overconfident.

We reemphasize that this work aims to perform OOD detection. Unlike uncertainty/calibration estimation/calibration approaches, we do not claim or intend that the probabilities produced are calibrated. To agree with the principle of maximum entropy, rather than “plausible probabilities”, we need to produce the lowest possible probabilities that are, nevertheless, capable of providing the highest classification accuracy possible. We may even question the meaning of calibrated probabilities in an open set environment such as OOD tasks, where we do not even know how often the system will deal with OOD examples.

Additionally, Fig. 6 shows that the entropy works as a high-quality score to distinguish the in-distribution from the out-distribution in neural networks trained with IsoMax loss.

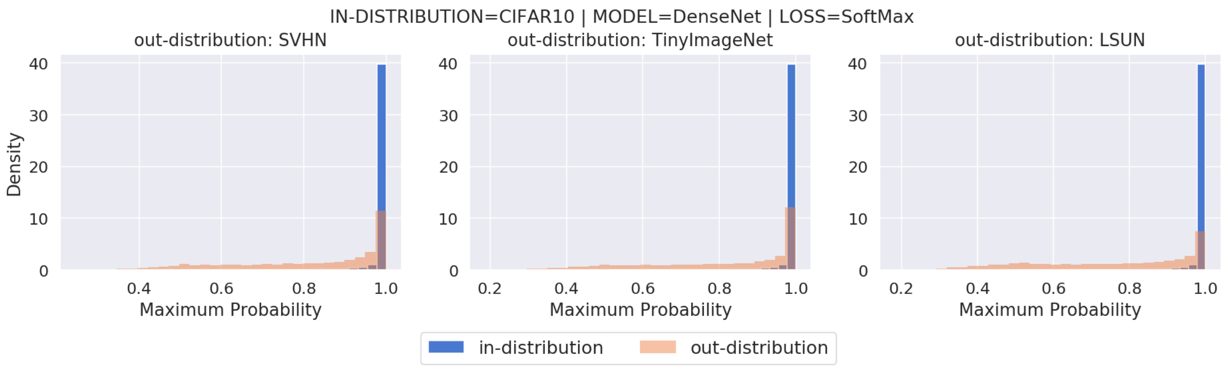
APPENDIX G
TRAINING METRICS

Fig. 7 shows that the training metrics are remarkably similar for SoftMax and IsoMax losses. Regardless of the dataset and model used, the loss values produced are similar. The same is true for the validation accuracy along the network training.

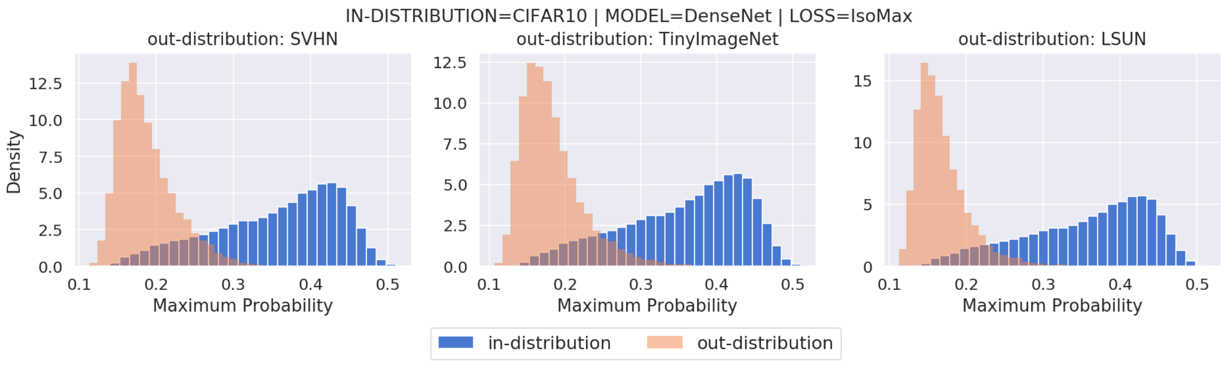
APPENDIX H
SOURCE CODE

The *code* to reproduce the experiments of this paper, as well as future updates, is available online[9].

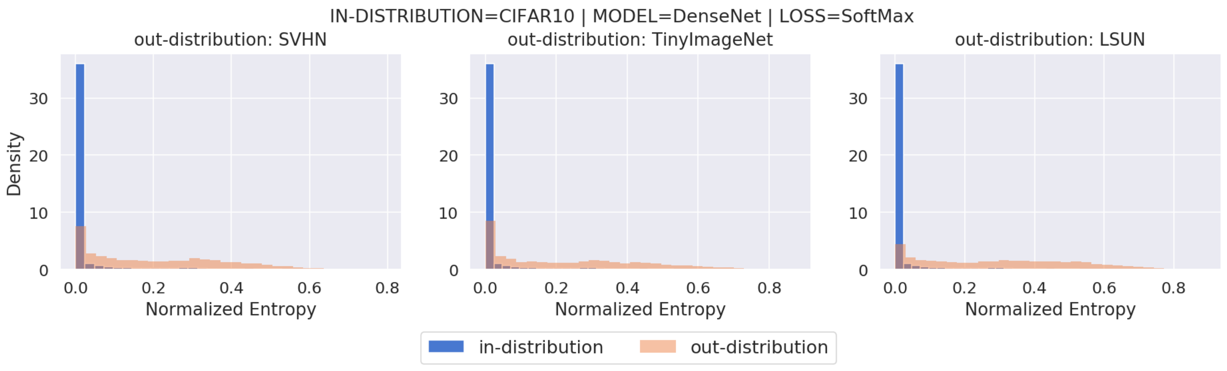
[9]https://github.com/dlmacedo/entropic-out-of-distribution-detection



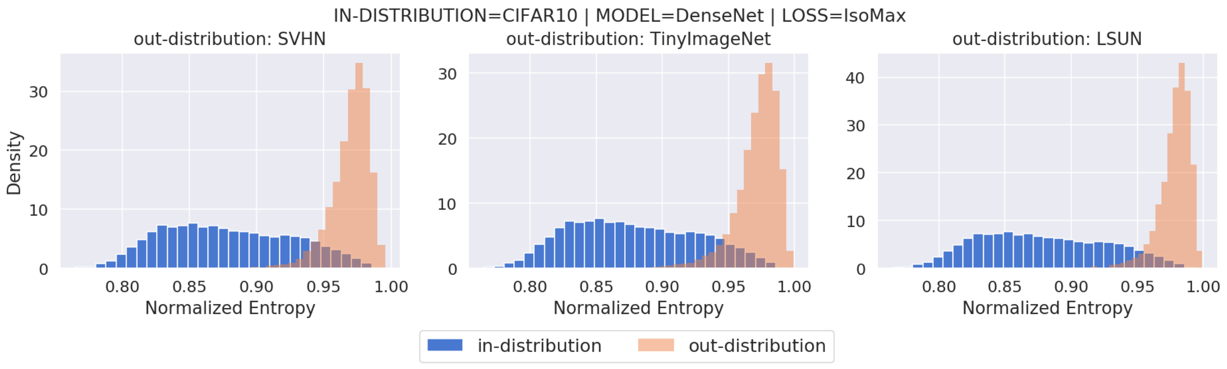
(a)



(b)

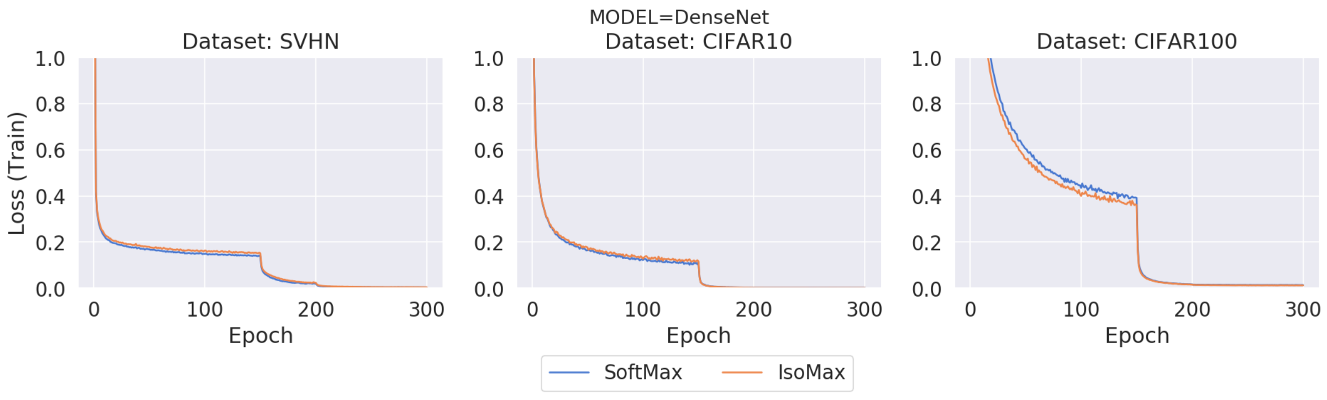


(c)

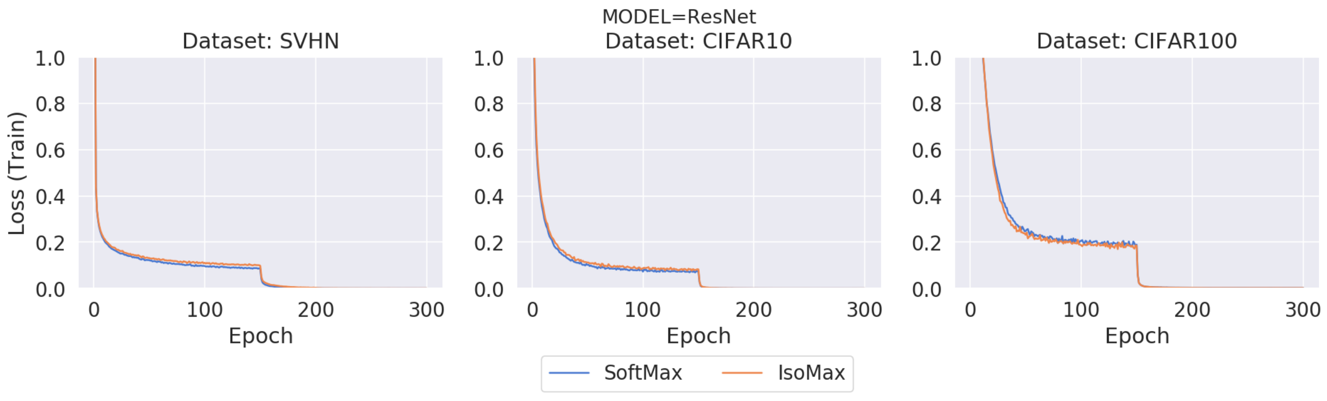


(d)

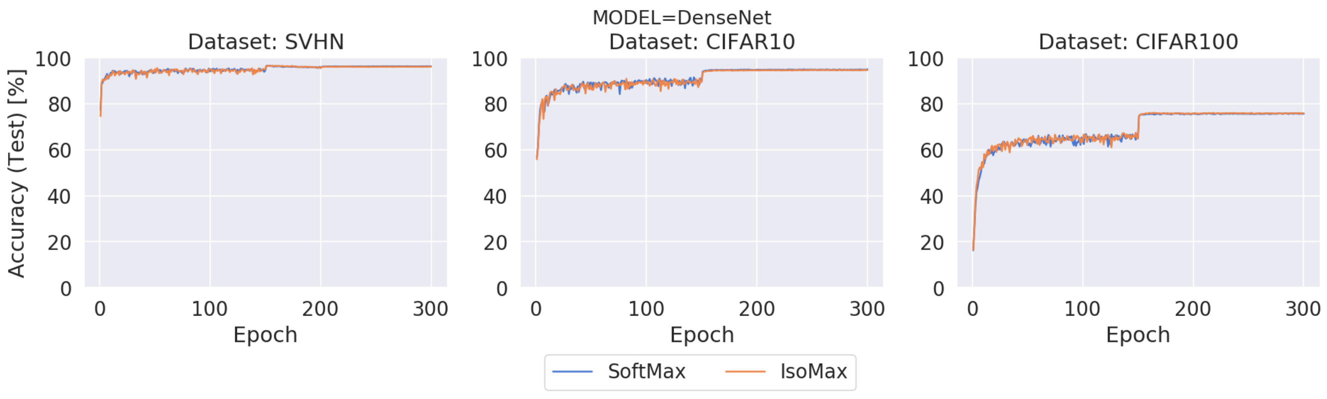
Fig. 6. Probabilities and entropies: (a) SoftMax loss produces high confident predictions for in-distribution samples. SoftMax loss usually provides high maximum probabilities even for OOD samples. (b) IsoMax loss produces less confident predictions for in-distribution samples than SoftMax loss. IsoMax loss commonly produces even lower maximum probability for OOD samples. (c) SoftMax loss provides low entropy (high confidence) for almost all in-distribution samples and even usually for OOD samples. (d) IsoMax loss produces high entropy for out-distributions. More precise separation between the in-distribution and out-distributions is obtained. *Following the principle of maximum entropy, rather than calibrated probabilities, we need to provide predictions as underconfident (high entropy) as possible as long as they fit the data appropriately, i.e., produce no classification accuracy drop.*



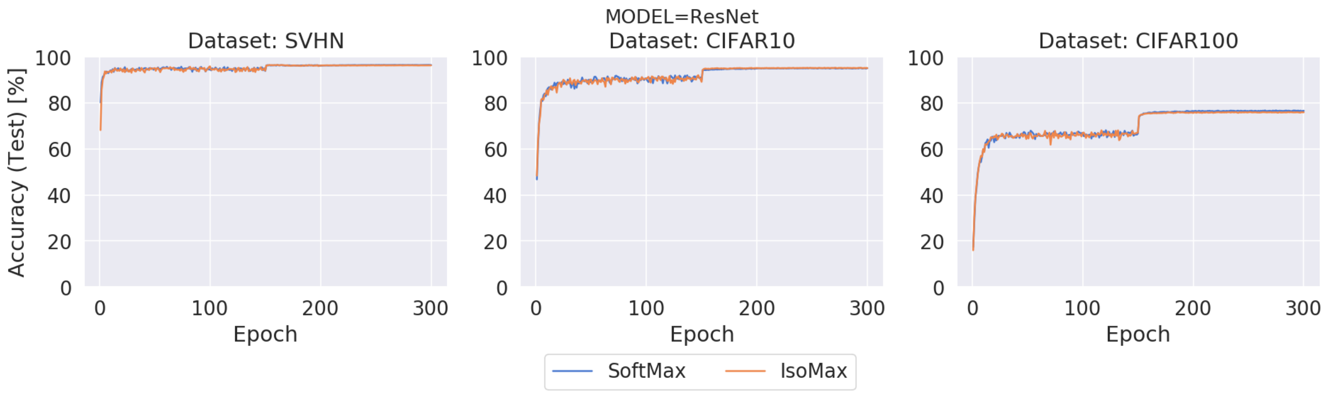
(a)



(b)



(c)



(d)

Fig. 7. Training metrics: (a,b) Training loss values and (c,d) test accuracies for a range of models, datasets, and losses. SoftMax loss and IsoMax loss present remarkably similar metrics throughout training, which confirms that IsoMax loss training is consistent and stable.

APPENDIX C - ENHANCED ISOTROPY MAXIMIZATION LOSS

Enhanced Isotropy Maximization Loss: Seamless and High-Performance Out-of-Distribution Detection Simply Replacing the SoftMax Loss

David Macêdo, *Member, IEEE,* and Teresa Ludermir, *Senior Member, IEEE*

Abstract—Current out-of-distribution detection approaches usually present special requirements (e.g., collecting outlier data and hyperparameter validation) and produce side effects (e.g., classification accuracy drop and slow/inefficient inferences). Recently, entropic out-of-distribution detection has been proposed as a seamless approach (i.e., a solution that avoids all previously mentioned drawbacks). The entropic out-of-distribution detection solution uses the IsoMax loss for training and the entropic score for out-of-distribution detection. The IsoMax loss works as a drop-in replacement of the SoftMax loss (i.e., the combination of the output linear layer, the SoftMax activation, and the cross-entropy loss) because swapping the SoftMax loss with the IsoMax loss requires no changes in the model's architecture or training procedures/hyperparameters. In this paper, we perform what we call an isometrization of the distances used in the IsoMax loss. Additionally, we propose replacing the entropic score with the minimum distance score. Experiments showed that these modifications significantly increase out-of-distribution detection performance while keeping the solution seamless. Besides being competitive with or outperforming all major current approaches, the proposed solution avoids all their current limitations, in addition to being much easier to use because only a simple loss replacement for training the neural network is required.

Index Terms—Out-of-Distribution Detection, Enhanced Isotropy Maximization Loss, Minimum Distance Score

1 Introduction

Neural networks have been used in classification tasks in many real-world applications [1]. In such cases, the system must typically identify whether a given input belongs to any of the classes on which it was trained. [2] called this capability out-of-distribution (OOD) detection and proposed datasets and metrics to allow standardized performance evaluation and comparison. However, current OOD detection solutions still have limitations (e.g., special requirements and side effects) that prevent more general use of OOD detection capabilities in practical real-world applications [3] (Table 1).

First, OOD detection solutions commonly present hyperparameters that usually presume access to out-of-distribution samples to be defined [4], [5], [13], [14], [15]. A consequence of presuming access to OOD samples to validate hyperparameters and using the same distribution to evaluate OOD detection results is producing overestimated performance estimations [16]. To avoid unrealistic access to OOD samples and overestimated performance, [5] proposed validating hyperparameters using adversarial samples. However, this requires the generation of adversarial examples. This procedure also requires determining hyperparameters (e.g., maximum adversarial perturbation) that are typically unknown when dealing with novel datasets. Similar arguments hold for solutions based on adversarial training [6], [14], [17], [18], [19], which also result in higher training time. Approaches based on the generation of adversarial examples or adversarial training may also have limited scalability when dealing with large images (e.g., ImageNet [20]).

Many solutions make use of the so-called *input preprocessing* technique introduced in ODIN [4]. However, the use of the mentioned technique *increases at least four times the inference delay and power consumption* [3], [12] because a combination of a first forward pass, backpropagation operation, and second forward pass is required [4], [5], [8], [15] for a single useful inference. Actually, approaches that may be applied directly to pretrained models and completely avoid training or fine-tuning the model [4], [5], [9] typically produce inefficient inferences and/or additional computational complexity to perform OOD detection [12, Section IV, D]. From a practical perspective, this is a drawback because inferences may be performed thousands or millions of times in the field. Thus, such approaches may be prohibitive (not sustainable) from environmental [21] and real-world cost-based perspectives.

Another harmful and common side effect is the so-called *classification accuracy drop* [8], [10]. In such cases, higher OOD detection performance is achieved at the expense of a drop in the classification accuracy compared with models trained using the usual SoftMax loss (i.e., the combination of the output linear layer, the SoftMax activation, and the

• *David Macêdo and Teresa Ludermir are with the Centro de Informática, Universidade Federal de Pernambuco, Brazil. E-mail: dlm@cin.ufpe.br. David Macêdo was with Montreal Institute for Learning Algorithms (MILA), Université de Montréal (UdeM), Quebec, Canada.*

TABLE 1
Out-of-distribution detection approaches: special requirements and side effects.

Approach	Special Requirement		Side Effect	
	Hyperparameter Tuning	Additional Data	Inefficient Inference or OOD Detection	Classification Accuracy Drop
ODIN [4]	Required	Not Required	Present	Not Present
Mahalanobis [5]	Required	Not Required	Present	Not Present
ACET [6]	Required	Not Required	Not Present	Present
Outlier Exposure [7]	Not Required	Required	Not Present	Not Present
Generalized ODIN [8]	Required	Not Required	Present	Present
Gram Matrices [9]	Not Required	Not Required	Present	Not Present
Scaled Cosine [10]	Not Required	Not Required	Not Present	Present
Energy-based [11]	Required	Required	Not Present	Not Present
Entropic (Seamless) [3], [12] IsoMax + Entropic Score	**Not Required**	**Not Required**	**Not Present**	**Not Present**
Entropic (Seamless) [ours] Enhanced IsoMax + Distance Score	**Not Required**	**Not Required**	**Not Present**	**Not Present**

cross-entropy loss [22])[1]. For example, in some situations, the solution proposed in [10] produces a classification accuracy drop of more than two percent when compared with SoftMax loss training of the same neural network. From a practical perspective, this situation is not desirable because the detection of out-of-distribution samples may be a rare event, and classification is usually the primary aim of the designed system [23]. The proposal also requires changing the training of the last layer by removing its weight decay to work properly. Therefore, this solution does not work as a SoftMax loss drop-in replacement.

Generalized ODIN [8] uses the *in-distribution* validation set to avoid the need to access OOD samples to determine the hyperparameters required by the solution. However, considering that CIFAR10 and CIFAR100 do not have separate sets for validation and testing, the results presented in the paper may be overestimated because the validation sets used to define the hyperparameters were reused for OOD detection performance estimation. A more realistic OOD detection performance estimation could have been achieved by removing the in-distribution validation set from in-distribution training data, which would probably produce an even higher *classification accuracy drop*. For example, *Generalized ODIN produces a nearly 3% classification accuracy drop* for ResNet34 trained on CIFAR100. Retraining the neural network using a very unusual fine-tuned regularization (dropout with $p = 0.7$), as suggested by the authors, does not actually solve the problem *because a classification accuracy drop of approximately 1% persists and the OOD detection performance plunges approximately 10%*.

The Generalized ODIN changes the architecture by adding a fully connected layer to the end of the network. We believe that this fact may explain the mentioned overfitting. We imagine that this problem may be even worse in datasets with fewer training examples. Additionally, the solution proposed in [8] is expensive and not environmentally friendly because it uses *input preprocessing* and thus produces slow and energy-inefficient inferences [3], [12].

Recently, many OOD detection approaches have used additional/outlier/background data [7], [11], [24]. In addition to requiring collecting additional data, these approaches require double the GPU memory size for training. The Gram matrices solution performs calculations on values produced by the model during inference [9] to perform OOD detection. This inference is efficient; however, the OOD detection has a high computational cost.

In some cases, an ensemble of classifiers is used [25]. For deep ensembles, [17] proposed an ensemble of same-architecture models trained with different random initial weights. Some proposals required structural changes in the model to perform OOD detection [26], and certain trials used uncertainty or confidence estimation/calibration techniques [27], [28], [29], [30], [31]. However, Bayesian neural networks that are used in most of these are typically more difficult to implement and require much more computational resources. Computational constraints also typically require approximations that compromise the performance, which is also affected by the prior distribution used [17]. For example, MC-dropout uses pretrained models with dropout activated during the test time. An average of many inferences is used to perform a single decision [32].

The *entropic* out-of-distribution detection approach, which is composed of the IsoMax loss for training and the entropic score for OOD detection, avoids all mentioned special requirements and side effects [3], [12]. Indeed, no hyperparameter tuning is required because *the entropic scale is a global constant kept equal to ten for all combinations of datasets and models*. Even if we call the entropic scale hyperparameter, the IsoMax loss does not involve hyperparameter *tuning* because the same constant value of entropic scale is used in all cases. This result is possible because Macêdo et al. showed in [3, Fig. 3] and in [12, Section IV, A] that *the OOD detection performance exhibits a well-behaved dependence on the entropic scale regardless of the dataset and model*. No

[1]In this paper, we consider that an approach does not present classification accuracy drop if it never presents a classification accuracy more than one percent lower than the correspondent softmax-loss-trained model.

Algorithm 1 PyTorch pseudocode for the enhanced IsoMax loss: Implementation.

```
class IsoMaxPlusLossFirstPart(nn.Module):
    """This part replaces the model classifier output layer nn.Linear()"""
    def __init__(self, num_features, num_classes):
        super(IsoMaxPlusLossFirstPart, self).__init__()
        self.num_features = num_features
        self.num_classes = num_classes
        self.prototypes = nn.Parameter(torch.Tensor(num_classes, num_features))
        nn.init.normal_(self.prototypes, mean=0.0, std=1.0)
        self.distance_scale = nn.Parameter(torch.Tensor(1))
        nn.init.constant_(self.distance_scale, 1.0)

    def forward(self, features):
        distances = torch.abs(self.distance_scale) * torch.cdist(
            F.normalize(features), F.normalize(self.prototypes),
            p=2.0, compute_mode="donot_use_mm_for_euclid_dist")
        logits = -distances
        return logits

class IsoMaxPlusLossSecondPart(nn.Module):
    """This part replaces the nn.CrossEntropyLoss()"""
    def __init__(self, entropic_scale = 10.0):
        super(IsoMaxPlusLossSecondPart, self).__init__()
        self.entropic_scale = entropic_scale

    def forward(self, logits, targets):
        """Probabilities and logarithms are calculated separately and sequentially"""
        """Therefore, nn.CrossEntropyLoss() must not be used to calculate the loss"""
        distances = -logits
        probabilities_for_training = nn.Softmax(dim=1)(-self.entropic_scale * distances)
        probabilities_at_targets = probabilities_for_training[range(distances.size(0)), targets]
        loss = -torch.log(probabilities_at_targets).mean()
        return loss
```

additional/outlier/background data are necessary. Models trained using IsoMax loss produce inferences as fast and energy-efficient as the inferences produced by softmax-loss-trained networks. OOD detection requires only a speedy entropy calculation. Finally, no classification accuracy drop is observed.

For real-world large-scale applications, we require efficient inference and high-performance solutions. By *efficient inference*, we mean solutions that do not increase the inference delay, computation, and energy consumption when compared with current softmax-loss-trained neural networks. By *high performance*, we mean solutions that do not present a classification accuracy drop and *simultaneously* exhibit much higher out-of-distribution detection when compared with usual SoftMax loss trained neural networks. In this paper, we are only interested in studying and comparing inference-efficient high-performance out-of-distribution detection solutions.

Starting from the IsoMax loss, we added the following contributions. First, we normalize both the prototypes and the features. Second, we change the initialization of the prototypes. Third, we add the *distance scale*, which is a *learnable scalar parameter* that multiplies the *feature-prototype distances*. We call the combination of these three modifications the *isometrization* of the *feature-prototype distances*. We call the proposed modified version of IsoMax the *enhanced* isotropy maximization loss or the *enhanced* IsoMax loss (IsoMax+ loss). Fourth, we use the *minimum feature-prototype distance* as a score to perform OOD detection. Considering that the minimum feature-prototype distance is calculated to perform the classification, *the OOD detection task presents essentially zero computational cost* because we simply reuse this value as a score to perform OOD detection. Fifth, in addition to experimental evidence, we provide insights into why a combination of training using IsoMax+ and performing OOD detection using the *minimum distance score* produces a substantial performance increase in OOD detection.

Our approach keeps the solution seamless (i.e., avoids the previously mentioned special requirements and side effects) while significantly increasing the OOD detection performance. Similar to IsoMax loss, IsoMax+ works as a *SoftMax loss drop-in replacement* because no procedures other than regular neural network training are required.

2 Background

Many approaches have been proposed to manage out-of-distribution detection. We may roughly classify them into three classes: training/fine-tuning methods and inference methods. Training/fine-tuning methods are used to train from scratch or to fine-tune pretrained models. Inference methods are applied to pretrained models (i.e., no training or fine-tuning is allowed), regardless of the training method used. Thus, training/fine-tuning and inference methods are complementary rather than competitors.

2.1 Training/Fine-tuning Methods

The first and most common training method is simply training with SoftMax loss and using the maximum probability score (MPS) [2] for OOD detection. Despite not presenting high OOD detection performance, this approach is seamless (i.e., no hyperparameters to tune, no additional data are necessary, efficient inference and OOD detection, no classification accuracy drop).

Techapanurak et al. proposed the Scaled Cosine approach [10] by adding a layer to the output of the neural

Algorithm 2 PyTorch pseudocode for the enhanced IsoMax loss: Interface.

```
class Model(nn.Module):
    def __init__(self):
    (...)
    #self.classifier = nn.Linear(num_features, num_classes)
    self.classifier = losses.IsoMaxPlusLossFirstPart(num_features, num_classes)

model = Model()
#criterion = nn.CrossEntropyLoss()
criterion = losses.IsoMaxPlusLossSecondPart()

outputs = model(inputs)
# outputs are equal to logits, which in turn are equivalent to negative distances
score = outputs.max(dim=1)[0] # this is the minimum distance score for detection
# the minimum distance score is the best option for the IsoMax+ loss
```

network to learn a scale to be multiplied by the cosine similarity that replaces the affine transformation in the SoftMax loss. The solution requires changing the training of the last layer by removing its weight decay to work properly. Therefore, this method does not work as a SoftMax loss drop-in replacement. Additionally, the solution is not seamless because *it produces a classification accuracy drop higher than two percent in some cases*.

Hsu et al. developed the Generalized ODIN solution [8]. Despite having hyperparameters, this solution exclusively uses *in-distribution* validation data to fine-tune the hyperparameters, preventing access to out-of-distribution data that is typically unavailable. Similar to ODIN, the solution uses *input preprocessing*. The solution is not seamless because, despite preventing access to out-of-distribution data, it requires hyperparameter tuning. This solution also produces inefficient inferences and decreases classification accuracy in some cases.

Considering that we can train a model from scratch or fine-tune pretrained models to the proposed custom data, training methods are typically useful in practice. However, except for the pure SoftMax loss and the IsoMax loss variants, other training methods (e.g., Scaled Cosine, Generalized ODIN) decrease classification accuracy, probably due to the fact that those approaches add a fully connected layer to the end of the neural network, which makes it more likely to overfit. Additionally, in some cases, even inefficient inferences are produced [8]. To our knowledge, the pure SoftMax and IsoMax loss variants are the only seamless training methods currently available.

2.1.1 Additional-Data Techniques

Training/fine-tuning methods may be enhanced by additional-data techniques. The additional data are from an unlabeled third distribution different from both the in-distribution and the out-of-distribution. This fact is important because out-of-distribution data is often unknown in practice. The additional data are usually called outlier data or background samples in the literature [7], [11], [24].

2.2 Inference Methods

Liang et al. proposed ODIN [4], which is essentially a combination of *input preprocessing* and *temperature calibration*. Input preprocessing consists of increasing the SoftMax score of any given input in a procedure that is inspired by adversarial attacks. Additionally, input preprocessing uses out-of-distribution samples to perform *temperature calibration*.

Lee at al. proposed the Mahalanobis approach [5], which consists of training an ad hoc generative classifier on the features extracted from intermediate layers of a pretrained model. OOD detection uses a regression model based on the Mahalanobis distances that are calculated using many layer activations (*feature ensemble*). Additionally, this approach uses *input preprocessing* to maximize performance. To avoid overestimating the OOD detection performance, this approach uses adversarial examples rather than out-of-distribution samples to validate hyperparameters because in real-world applications, we typically do not know out-of-distribution data.

Sastry et al. developed the Gram matrices method [9], which identifies inconsistencies between activity patterns and predicted classes. This method characterizes activity patterns by Gram matrices and may be used to perform OOD detection.

First, we may believe that inference methods should be preferred due to avoiding training or fine-tuning models. However, in the real world, we typically have custom data that require us to train from scratch or at least fine-tune the proposed model. In such cases, we have no good reason not to train/fine-tune the model with a *seamless training method* to start from a better baseline, regardless of planning to subsequently using an inference method. We emphasize that *training and inference methods are complementary rather than competitors*.

Additionally, all known inference methods produce inference or require expensive additional computation to perform OOD detection. Thus, not using a training method yields higher computational costs and more energy-inefficient inferences during the thousands or millions of times the system will operate in the field. Regardless, inference methods may be applied after a training method to improve performance if the previously mentioned drawback is not an issue in a particular application.

3 Enhanced IsoMax and the Distance Score

In this section, we present the enhanced IsoMax for training and the minimum distance score for inference. By combining those methods, we develop a seamless, scalable, and high-performance out-of-distribution detection approach.

3.1 Enhanced Isotropy Maximization Loss

We consider an input $\boldsymbol{x}$ applied to a neural network that performs a parametrized transformation $\boldsymbol{f_\theta}(\boldsymbol{x})$. We also consider $\boldsymbol{p}^j_{\boldsymbol{\phi}}$ to be the *learnable prototype* associated with class j. Additionally, let the expression $\|\boldsymbol{f_\theta}(\boldsymbol{x}){-}\boldsymbol{p}^j_{\boldsymbol{\phi}}\|$ represent the *nonsquared Euclidean distance* between $\boldsymbol{f_\theta}(\boldsymbol{x})$ and $\boldsymbol{p}^j_{\boldsymbol{\phi}}$. Finally, we consider $\boldsymbol{p}^k_{\boldsymbol{\phi}}$ as the learnable prototype associated with the correct class for the input $\boldsymbol{x}$. Thus, we write the IsoMax loss [3] using the equation below:

$$\mathcal{L}_{\text{IsoMax}} = -\log^\dagger \left(\frac{\exp(-E_s \|\boldsymbol{f_\theta}(\boldsymbol{x}) - \boldsymbol{p}^k_{\boldsymbol{\phi}}\|)}{\sum\limits_j \exp(-E_s \|\boldsymbol{f_\theta}(\boldsymbol{x}) - \boldsymbol{p}^j_{\boldsymbol{\phi}}\|)} \right) \tag{1}$$

In the above equation, E_s represents the entropic scale. From Equation (1), we observe that the distances from IsoMax loss are given by the expression $\mathcal{D} = \|\boldsymbol{f_\theta}(\boldsymbol{x}) - \boldsymbol{p}^j_{\boldsymbol{\phi}}\|$. During inference, probabilities calculated based on these distances are used to produce the negative entropy, which serves as a score to perform OOD detection. However, because the features $\boldsymbol{f_\theta}(\boldsymbol{x})$ are not normalized, examples with low norms are unjustifiably favored to be considered OOD examples because they tend to produce high entropy. Additionally, because the weights $\boldsymbol{p}^j_{\boldsymbol{\phi}}$ are not normalized, examples from classes that present prototypes with low norms are unjustifiably favored to be considered OOD examples for the same reason. Thus, we propose replacing $\boldsymbol{f_\theta}(\boldsymbol{x})$ with its normalized version given by $\widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} = \boldsymbol{f_\theta}(\boldsymbol{x}) / \|\boldsymbol{f_\theta}(\boldsymbol{x})\|$. Additionally, we propose replacing $\boldsymbol{p}^j_{\boldsymbol{\phi}}$ with its normalized version given by $\widehat{\boldsymbol{p}^j_{\boldsymbol{\phi}}} = \boldsymbol{p}^j_{\boldsymbol{\phi}} / \|\boldsymbol{p}^j_{\boldsymbol{\phi}}\|$. The expression $\|\boldsymbol{v}\|$ represents the 2-norm of a given vector $\boldsymbol{v}$.

However, while the distances in the original IsoMax loss may vary from zero to infinity, the distance between two normalized vectors is always equal to or lower than two. To avoid this unjustifiable and unreasonable restriction, we introduce the *distance scale* d_s, which is a *scalar learnable parameter*. Naturally, we require the distance scale to always be positive by taking its absolute value $|d_s|$.

Feature normalization makes the solution isometric regardless of the norm of the features produced by the examples. The distance scale is class independent because it is a *single* scalar value learnable during training. The weight normalization and the class independence of the distance scale make the solution isometric regarding all classes. The final distance is isometric because it produces an isometric treatment of all features, prototypes, and classes. Therefore, we can write the *isometric distances* used by the IsoMax+ loss as $\mathcal{D}_{\mathcal{I}} = |d_s| \, \|\widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}^j_{\boldsymbol{\phi}}}\|$. Returning to Equation (1), we can finally write the expression for the IsoMax+ loss as follows (see Algorithm 1):

$$\mathcal{L}_{\text{IsoMax+}} = -\log^{\dagger\dagger} \left(\frac{\exp(-E_s \, |d_s| \, \|\widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}^k_{\boldsymbol{\phi}}}\|)}{\sum\limits_j \exp(-E_s \, |d_s| \, \|\widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}^j_{\boldsymbol{\phi}}}\|)} \right) \tag{2}$$

Applying the entropy maximization trick (i.e., the removal of the entropic scale E_s for inference) [3], we can write the expression for the IsoMax+ loss probabilities used during inference for performing OOD detection when using the maximum probability score or the entropic score [3]:

$$\mathcal{P}_{\text{IsoMax+}}(y^{(i)}|\boldsymbol{x}) = \frac{\exp(-\,|d_s| \, \|\widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}^i_{\boldsymbol{\phi}}}\|)}{\sum\limits_j \exp(-\,|d_s| \, \|\widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}^j_{\boldsymbol{\phi}}}\|)} \tag{3}$$

Different from IsoMax loss, where the prototypes are initialized to a zero vector, we initialized all prototypes using a normal distribution with a mean of zero and a standard deviation of one. This approach is necessary because we normalize the prototypes when using IsoMax+ loss. The distance scale is initialized to one, and we added no hyperparameters to the solution.

3.2 Minimum Distance Score

Motivated by the desired characteristics of the isometric distances used in IsoMax+, we use the minimum distance as score for performing OOD detection. Naturally, the *minimum distance score* for IsoMax+ is given by:

$$\text{MDS} = \min_j \left(\|\widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}^j_{\boldsymbol{\phi}}}\| \right) \tag{4}$$

In this equation, $|d_s|$ was removed because, after training, it is a scale factor that does not affect the detection decision. Considering that the minimum distance is computed to perform the classification because the predicted class is the one that presents *the lowest feature-prototype distance*, when using the minimum distance score, *the OOD detection exhibits essentially zero latency and computational cost* because we simply reuse the minimum distance that was already calculated for classification purpose.

4 Experiments

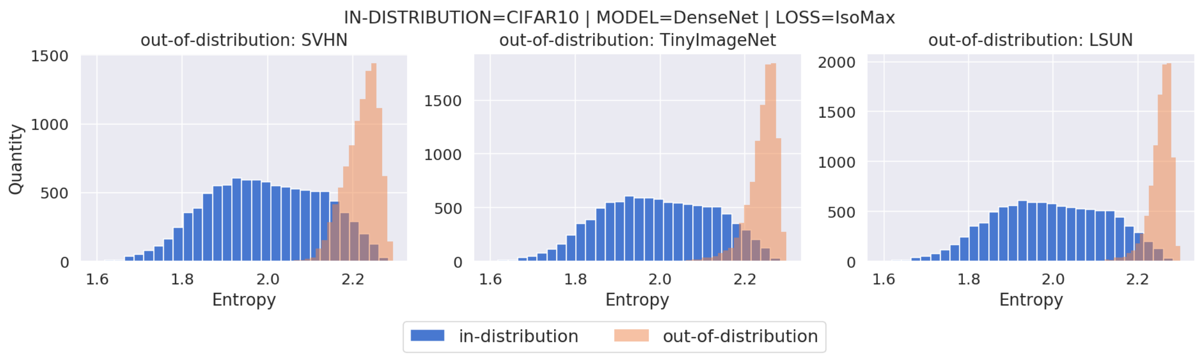
To allow standardized comparison, we used the datasets, training procedures, and metrics that were established in [2] and used in many subsequent OOD detection papers [4], [5], [6]. We trained many 100-layer DenseNetBCs with growth rate $k = 12$ (i.e., 0.8M parameters) [34], 110-layer ResNets (*correct size proper implementation*) [35][2], and 34-layer ResNets (*overparametrized commonly used implementation*) [35][3] on CIFAR10 [36], CIFAR100 [36], SVHN [37], and TinyImageNet

†*The probability (i.e., the expression between the outermost parentheses) and logarithm operations are computed sequentially and separately for higher OOD detection performance [3] (see the source code).*

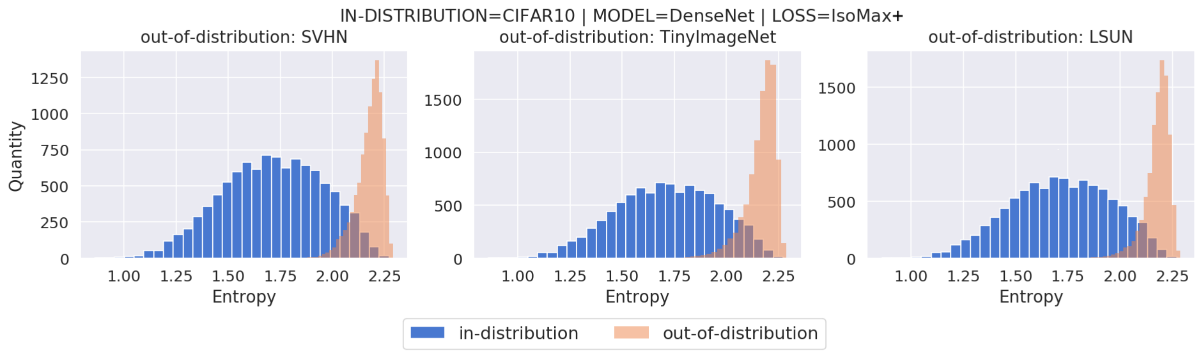
††*Similar to [3], the probability (i.e., the expression between the outermost parentheses) and logarithm operations are computed sequentially and separately for higher OOD detection performance (see Algorithm 1 and the source code).*

[2]https://github.com/akamaster/pytorch_resnet_cifar10

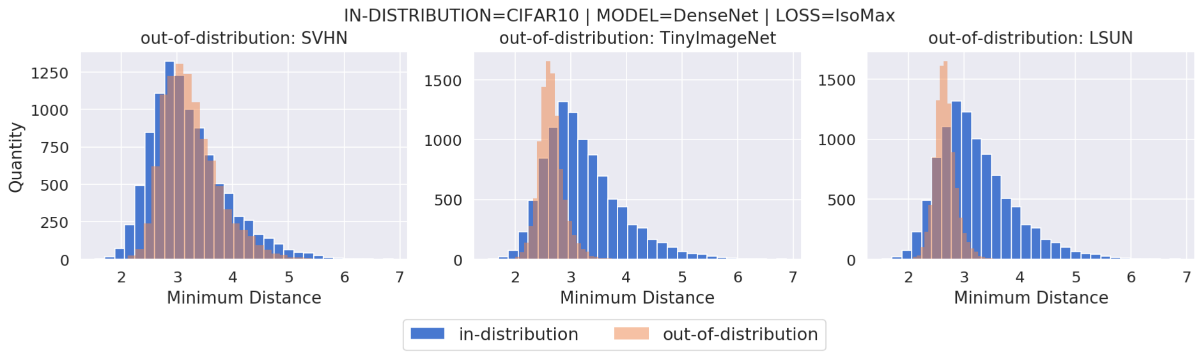
[3]https://github.com/pokaxpoka/deep_Mahalanobis_detector



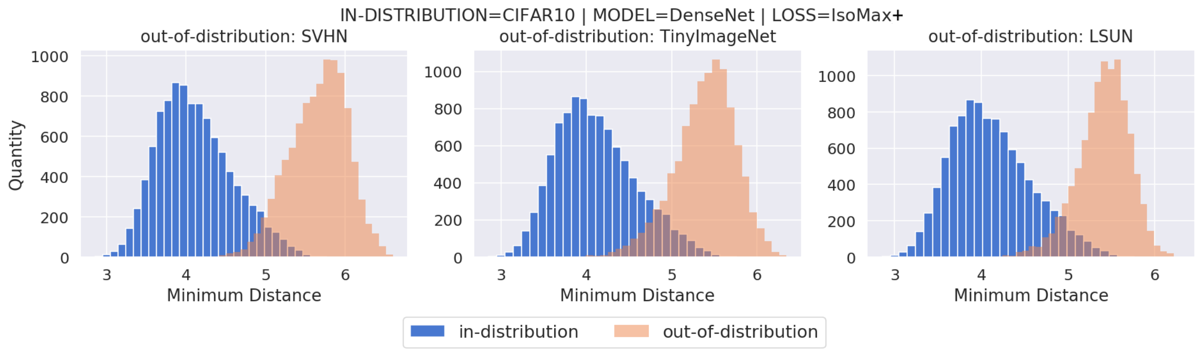
(a)



(b)



(c)



(d)

Fig. 1. In agreement with other studies [33], (a) IsoMax (b) and IsoMax+ produce higher entropy/uncertainty on out-of-distributions than in-distribution. Therefore, the entropic score produces high OOD detection performance in both cases. (c) However, IsoMax does not make the in-distribution closer to the prototypes than out-of-distributions. (d) Concurrently, by introducing *distance isometrization*, IsoMax+ gets in-distribution closer to the prototypes while pushing out-of-distribution data far away, which is what we expect based on the findings of other recent studies [33]. This result also explains why the minimum distance score produces high performance when using IsoMax+, while producing low performance when using IsoMax. See also Table 6.

TABLE 2
Classification accuracy of models trained using SoftMax, IsoMax, and IsoMax+ losses.

Model	Data	Train Accuracy (%) [↑]	Test Accuracy (%) [↑]
		SoftMax Loss / IsoMax Loss / IsoMax+ Loss	
DenseNet100	CIFAR10	99.9±0.1 / 99.9±0.1 / 99.9±0.1	95.3±0.2 / 95.2±0.3 / 95.3±0.1
	CIFAR100	99.9±0.1 / 99.0±0.1 / 99.9±0.1	77.2±0.3 / 77.3±0.4 / 77.0±0.3
	SVHN	97.0±0.2 / 97.8±0.2 / 97.0±0.3	96.5±0.2 / 96.6±0.3 / 96.5±0.2
ResNet110	CIFAR10	99.9±0.1 / 99.9±0.1 / 99.9±0.1	94.4±0.3 / 94.5±0.3 / 94.6±0.2
	CIFAR100	99.5±0.1 / 99.9±0.1 / 99.8±0.1	72.7±0.2 / 74.1±0.4 / 73.9±0.3
	SVHN	99.8±0.1 / 99.9±0.1 / 99.5±0.1	96.6±0.3 / 96.8±0.2 / 96.9±0.3

Besides preserving classification accuracy compared with SoftMax loss- and IsoMax loss-trained networks, IsoMax+ loss-trained models show higher OOD detection performance. Results are shown as means and standard deviations of five different iterations (see Table 3).

TABLE 3
Fair comparison of seamless approaches: No hyperparameter tuning, no additional/outlier/background data, no classification accuracy drop, and no slow/inefficient inferences.

Model	Data (training)	OOD (unseen)	Out-of-Distribution Detection: Seamless Approaches.	
			TNR@TPR95 (%) [↑]	AUROC (%) [↑]
			SoftMax$_{\text{ES}}$ / IsoMax$_{\text{ES}}$ / IsoMax+$_{\text{MDS}}$ (ours)	
DenseNet100	CIFAR10	SVHN	40.4±5.3 / 78.6±9.0 / **97.0±0.7**	89.1±2.2 / 96.2±1.0 / **99.4±0.1**
		TinyImageNet	58.0±3.7 / 83.9±3.6 / **92.6±2.4**	94.0±0.6 / 97.1±0.4 / **98.5±0.3**
		LSUN	64.6±1.7 / 90.3±1.4 / **94.3±1.4**	95.2±0.4 / 98.0±0.3 / **99.1±0.2**
	CIFAR100	SVHN	21.9±2.8 / 29.6±3.7 / **78.2±4.1**	78.4±3.3 / 88.8±2.8 / **96.3±1.3**
		TinyImageNet	24.0±3.0 / 49.3±4.9 / **85.5±2.8**	75.5±2.1 / 90.8±2.1 / **97.5±0.8**
		LSUN	24.9±2.9 / 60.6±6.6 / **78.3±3.9**	77.2±3.2 / 93.0±1.4 / **97.2±1.3**
	SVHN	CIFAR10	85.2±3.3 / 93.4±1.1 / **95.1±0.4**	97.3±0.3 / 98.4±0.2 / **99.1±0.1**
		TinyImageNet	90.8±0.6 / 96.2±0.9 / **98.1±0.3**	98.3±0.2 / 99.0±0.2 / **99.6±0.1**
		LSUN	87.9±0.8 / 94.3±1.7 / **97.7±1.1**	97.8±0.3 / 98.7±0.2 / **99.6±0.2**
ResNet110	CIFAR10	SVHN	35.6±5.0 / 68.7±5.5 / **85.3±3.9**	88.9±0.9 / 94.8±1.5 / **98.1±0.9**
		TinyImageNet	39.0±4.1 / 65.4±5.6 / **76.2±2.7**	88.2±2.1 / 94.3±0.4 / **96.1±0.5**
		LSUN	46.9±5.7 / 80.7±2.2 / **86.9±2.5**	91.2±1.7 / 96.5±0.3 / **97.9±0.7**
	CIFAR100	SVHN	16.6±1.5 / 20.8±5.9 / **41.0±6.4**	73.6±3.7 / 85.3±1.3 / **88.9±1.2**
		TinyImageNet	15.6±2.1 / 22.8±1.9 / **44.7±5.9**	71.5±1.7 / 81.3±2.3 / **88.0±2.9**
		LSUN	16.5±3.3 / 22.9±3.5 / **46.1±4.3**	72.6±4.2 / 83.3±2.0 / **89.1±2.4**
	SVHN	CIFAR10	80.9±3.5 / 84.3±1.3 / **88.4±2.3**	95.1±0.3 / 96.5±0.2 / **97.4±0.3**
		TinyImageNet	84.0±3.2 / 87.4±2.2 / **93.4±1.8**	96.6±0.7 / 96.7±0.6 / **98.5±0.7**
		LSUN	81.4±2.3 / 84.7±2.1 / **90.2±2.2**	96.1±0.3 / 95.8±0.4 / **97.8±0.3**

SoftMax$_{\text{ES}}$ means training using SoftMax loss and performing OOD detection using the entropic score (ES). IsoMax$_{\text{ES}}$ means training using IsoMax loss and performing OOD detection using the entropic score (ES). IsoMax+$_{\text{MDS}}$ means training using IsoMax+ loss and performing OOD detection using minimum distance score (MDS). Results are shown as the mean and standard deviation of five iterations. The best results are shown in bold. See Table 2.

[20] datasets with SoftMax, IsoMax, and IsoMax+ losses using the same procedures (e.g., initial learning rate, learning rate schedule, weight decay).

We used SGD with the Nesterov moment equal to 0.9 with a batch size of 64 and an initial learning rate of 0.1. The weight decay was 0.0001, and we did not use dropout. We trained during 300 epochs for CIFAR10, CIFAR100, and SVHN; and during 90 epochs for TinyImageNet. We used a learning rate decay rate equal to ten applied in epoch numbers 150, 200, and 250 for CIFAR10, CIFAR100, and SVHN; and 30 and 60 for TinyImageNet. The code to reproduce the results and replace the SoftMax loss by the IsoMax+ loss is available at https://github.com/dlmacedo/entropic-out-of-distribution-detection.

We used resized images from the TinyImageNet [20], the Large-scale Scene UNderstanding dataset (LSUN) [38], CIFAR10, and SVHN [37] to create out-of-distribution samples. We added these out-of-distribution images to the validation sets of in-distribution data to form the test sets and evaluate the OOD detection performance.

We evaluated the OOD detection performance using the true negative rate at 95% true positive rate (TNR@TPR95), the area under the receiver operating characteristic curve (AUROC) and the detection accuracy (DTACC), which corresponds to the maximum classification probability over all possible thresholds δ:

$$1 - \min_{\delta} \left\{ P_{\texttt{in}}\left(o\left(\mathbf{x}\right) \leq \delta\right) P\left(\mathbf{x} \text{ is from } P_{\texttt{in}}\right) \right.$$
$$\left. + P_{\texttt{out}}\left(o\left(\mathbf{x}\right) > \delta\right) P\left(\mathbf{x} \text{ is from } P_{\texttt{out}}\right) \right\},$$

where $o(\mathbf{x})$ is the OOD detection score. It is assumed that both positive and negative samples have an equal probability of being in the test set, i.e., $P\left(\mathbf{x} \text{ is from } P_{\texttt{in}}\right) = P\left(\mathbf{x} \text{ is from } P_{\texttt{out}}\right)$. All metrics follow the calculation procedures specified in [5].

TABLE 4
Unfair comparison with approaches that use input preprocessing and produce slow/inefficient inferences in addition to requiring validation using adversarial examples.

Model	Data (training)	OOD (unseen)	Comparison with approaches that use input preprocessing and adversarial validation.	
			AUROC (%) [↑]	DTACC (%) [↑]
			ODIN / Mahalanobis / IsoMax+$_{\text{MDS}}$ (ours)	ODIN / Mahalanobis / IsoMax+$_{\text{MDS}}$ (ours)
DenseNet100	CIFAR10	SVHN	92.1±0.2 / 97.2±0.3 / **99.4±0.1**	86.1±0.3 / 91.9±0.3 / **96.3±0.4**
		TinyImageNet	97.2±0.3 / 97.7±0.2 / **98.5±0.3**	91.9±0.3 / **94.3±0.5** / **93.9±0.6**
		LSUN	98.5±0.3 / 98.6±0.2 / **99.1±0.2**	94.3±0.3 / **95.7±0.4** / **95.3±0.5**
	CIFAR100	SVHN	88.0±0.5 / 91.3±0.4 / **96.3±1.3**	80.0±0.6 / 84.3±0.4 / **90.3±0.5**
		TinyImageNet	85.6±0.5 / **96.7±0.3** / **97.5±0.8**	77.6±0.5 / **91.0±0.4** / **91.3±0.3**
		LSUN	85.7±0.6 / **97.1±1.9** / **97.2±1.3**	77.5±0.4 / **92.5±0.8** / **91.7±0.7**
ResNet34	CIFAR10	SVHN	86.0±0.3 / 95.0±0.3 / **98.0±0.4**	77.1±0.4 / 88.7±0.3 / **93.5±0.4**
		TinyImageNet	92.6±0.3 / **98.3±0.4** / 95.3±0.3	86.5±0.5 / **94.8±0.3** / 90.0±0.4
		LSUN	93.0±0.4 / **98.8±0.3** / 96.3±0.4	86.3±0.4 / **96.8±0.4** / 92.1±0.5
	CIFAR100	SVHN	71.0±0.4 / 84.0±0.6 / **88.0±0.7**	68.0±0.5 / 77.3±0.7 / **82.1±0.4**
		TinyImageNet	83.1±0.3 / 87.3±0.5 / **90.5±0.4**	76.2±0.4 / **84.0±0.4** / **84.2±0.5**
		LSUN	81.0±0.3 / 82.0±0.5 / **88.6±0.6**	75.2±0.3 / 79.3±0.5 / **82.5±0.4**

ODIN and Mahalanobis were applied to models trained using SoftMax loss. These approaches compute at least four times slower and less power efficient inferences [3] because they use input preprocessing. Their hyperparameters were validated using adversarial examples. Additionally, Mahalanobis requires feature extraction for training ad-hoc models to perform OOD detection. Finally, feature ensemble was also used. IsoMax+$_{\text{MDS}}$ (ours) means training using IsoMax+ loss and performing OOD detection using minimum distance score (MDS). No hyperparameter tuning is required when using IsoMax+ loss for training and the MDS for OOD detection. The IsoMax+ loss OOD detection performance shown in this table may be increased further without relying on input preprocessing or hyperparameter tuning by replacing the minimum distance score with the Mahalanobis [5] or the energy score [11]. Results are the mean and standard deviation of five runs. The best results are shown in bold.

TABLE 5
Unfair comparison of outlier exposure-enhanced SoftMax loss with IsoMax loss and IsoMax+ loss without using additional data.

Model	Data (training)	Comparison of IsoMax loss variants without using additional data with outlier exposure-enhanced SoftMax loss.	
		TNR@TPR95 (%) [↑]	AUROC (%) [↑]
		SoftMax$^{\text{OE}}_{\text{ES}}$ / IsoMax$_{\text{ES}}$ / IsoMax+$_{\text{MDS}}$ (ours)	SoftMax$^{\text{OE}}_{\text{ES}}$ / IsoMax$_{\text{ES}}$ / IsoMax+$_{\text{MDS}}$ (ours)
DenseNet100	CIFAR10	**94.4±1.4** / 84.2±4.6 / **94.6±1.5**	98.0±0.3 / 97.1±0.5 / **99.0±0.2**
	CIFAR100	36.4±9.4 / 46.5±5.0 / **80.6±3.6**	83.8±5.6 / 90.8±2.1 / **97.0±1.1**
ResNet110	CIFAR10	**82.4±2.1** / 71.6±4.4 / **82.8±3.0**	**96.8±0.7** / 95.2±0.7 / **97.3±0.7**
	CIFAR100	29.1±4.5 / 22.1±3.7 / **43.9±5.5**	80.5±2.8 / 83.3±1.8 / **88.6±2.1**

SoftMax$^{\text{OE}}_{\text{ES}}$ means training using SoftMax loss enhanced during training using outlier exposure [7], which requires the collection of outlier data, and performing OOD detection using the entropic score. We used the same outlier data used in [7]. We collected the same amount of outlier data as the number of training examples present in the training set used to train SoftMax$^{\text{OE}}$. Despite being possible [12], the IsoMax loss and IsoMax+ loss were not enhanced by outlier exposure to keep the solution seamless. IsoMax$_{\text{ES}}$ means training using IsoMax loss and performing OOD detection using the entropic score. IsoMax+$_{\text{MDS}}$ (ours) means training using IsoMax+ loss and performing OOD detection using minimum distance score (MDS). The values of the performance metrics TNR@TPR95 and AUROC were averaged over all out-of-distribution (SVHN, TinyImageNet, and LSUN). Results are shown as the mean and standard deviation of five iterations. The best results are shown in bold.

TABLE 6
Comparison of IsoMax variants using different scores.

Model	Data (training)	Comparison of IsoMax loss variants using diferent scores.	
		TNR@TPR95 (%) [↑]	AUROC (%) [↑]
		IsoMax$_{\text{ES}}$ / IsoMax$_{\text{MDS}}$ / IsoMax+$_{\text{ES}}$ / IsoMax+$_{\text{MDS}}$ (ours)	IsoMax$_{\text{ES}}$ / IsoMax$_{\text{MDS}}$ / IsoMax+$_{\text{ES}}$ / IsoMax+$_{\text{MDS}}$ (ours)
DenseNet100	CIFAR10	84.2±4.6 / 0.9±0.5 / 89.3±2.3 / **94.6±1.5**	97.1±0.5 / 65.5±6.0 / 97.9±0.3 / **99.0±0.2**
	CIFAR100	46.5±5.0 / 6.2±6.1 / 63.7±8.0 / **80.6±3.6**	90.8±2.1 / 50.1±7.4 / 94.0±1.3 / **97.0±1.1**
ResNet110	CIFAR10	71.6±4.4 / 0.1±0.1 / 74.8±3.5 / **82.8±3.0**	95.2±0.7 / 83.7±4.1 / 95.2±0.6 / **97.3±0.7**
	CIFAR100	22.1±3.7 / 1.6±2.0 / 22.3±5.3 / **43.9±5.5**	83.3±1.8 / 61.9±7.2 / 84.4±0.8 / **88.6±2.1**

IsoMax$_{\text{ES}}$ means training using IsoMax loss and performing OOD detection using the entropic score (ES). IsoMax$_{\text{MDS}}$ means training using IsoMax loss and performing OOD detection using the minimum distance score (MDS). IsoMax+$_{\text{ES}}$ means training using IsoMax+ loss and performing OOD detection using entropic score (ES). IsoMax+$_{\text{MDS}}$ (ours) means training using IsoMax+ loss and performing OOD detection using minimum distance score (MDS). The values of the performance metrics TNR@TPR95 and AUROC were averaged over all out-of-distribution (SVHN, TinyImageNet, and LSUN). Results are shown as the mean and standard deviation of five iterations. The best results are shown in bold. See Fig. 1.

In this paper, we did not directly present a comparison with the approaches that produce *classification accuracy drop* (e.g., [8], [10]) because this is a substantial limitation from a practical perspective [23]. Regardless, it is easy to notice that IsoMax+ presents a similar OOD detection performance of Generalized ODIN (difference lower than the margin of error alternatively in favor of one or another) while simultaneously avoiding classification accuracy drop, hyperparameter tuning, and inefficient inferences. Moreover, IsoMax+ is also much easier to implement and to use.

For example, on the one hand, when using Generalized ODIN, we need to exclude some components of the solution from receiving weight decay, which requires optimization procedure modifications. On the other hand, just a loss replacement is needed in our proposal. Additionally, Generalized ODIN has many flavors. Consequently, it is hard to know a prior which one will perform best for a given dataset and model without training all of them from scratch using the same data and model.

5 Results and Discussion

In this section, we present the results and discussion. We initially show that the enhanced IsoMax produces classification accuracies that are comparable to those of the SoftMax loss function. We then show that the enhanced IsoMax produces much higher OOD detection performance than the IsoMax Loss and the SoftMax Loss.

5.1 Classification Accuracy

Table 2 shows classification accuracies. We see that IsoMax+ loss never exhibits *classification accuracy drop* compared to SoftMax loss or IsoMax loss regardless of the dataset and model. The IsoMax loss variants achieve more than one percent better accuracy than the SoftMax loss when using ResNet110 on the CIFAR100 dataset.

5.2 Out-of-Distribution Detection

Table 3 summarizes the results of the *fair* OOD detection comparison. We report the results using the entropic score for SoftMax loss (SoftMax$_{\text{ES}}$) and IsoMax loss (IsoMax$_{\text{ES}}$) because this score always overcame the maximum probability score in these cases. For IsoMax+, we report the values using the minimum distance score (IsoMax+$_{\text{MDS}}$) because this method overcame the maximum probability and the entropic score in this situation.

All approaches are accurate (i.e., exhibit no *classification accuracy drop*); fast and power-efficient (i.e., inferences are performed without *input preprocessing*); and no hyperparameter tuning was performed. Additionally, no additional/outlier/background data are required. IsoMax+$_{\text{MDS}}$ always overcomes IsoMax$_{\text{ES}}$ performance, regardless of the model, dataset, and out-of-distribution data.

The minimum distance score produces high OOD detection performance when combined with IsoMax+, which shows that the isometrization of the distances works appropriately in this case. However, the same minimum distance score produced low OOD detection performance when combined with the original IsoMax loss. Fig. 1, and Table 6 provide an explanation for this fact.

Table 4 summarizes the results of an *unfair* OOD detection comparison because the methods have different requirements and produce distinct side effects. ODIN [4] and the Mahalanobis[4] [5] approaches require adversarial samples to be generated to validate hyperparameters for each combination of dataset and model. These approaches also use *input preprocessing, which makes inferences at least four times slower and less energy-efficient* [3], [12]. Validation using adversarial examples may be cumbersome when performed from scratch on novel datasets because hyperparameters such as optimal adversarial perturbations may be unknown in such cases. IsoMax+$_{\text{MDS}}$ does not have these requirements, and does not produce the side effects.

Additionally, IsoMax+$_{\text{MDS}}$ achieves higher performance than ODIN by a large margin. In addition to the differences between the entropy maximization trick and temperature calibrations present in [3], [12], we emphasize that training with entropic scale affects the learning of all weights, while changing the temperature during inference affects only the last layer. Thus, the fact that the proposed solution overcomes ODIN by a safe margin is evidence that the *entropy maximization trick produces much higher OOD detection performance than temperature calibration*, even when the latter is combined with input preprocessing. Moreover, the entropy maximization trick does not require access to validation data to tune the temperature.

In addition to being seamless and avoiding the drawbacks of the Mahalanobis approach, IsoMax+$_{\text{MDS}}$ typically overcomes it in terms of AUROC and produces similar performance when considering the DTACC. We emphasize that the IsoMax+ loss OOD detection performance presented in Table 4 may increase further without requiring input preprocessing or hyperparameter tuning by replacing the minimum distance score with the Mahalanobis score [5].

Table 5 *unfairly* compares the performance of the proposed approach with the outlier exposure solution. Similar to IsoMax variants, the outlier exposure approach does not require hyperparameter tuning and produces efficient inferences. However, it does require collecting outlier data, while our approach does not. We emphasize that outlier exposure may also be combined with IsoMax loss variants to further increase the OOD detection performance [12]. In the table, we present the IsoMax loss variants *without outlier exposure* to show that the outlier exposure-enhanced SoftMax loss typically achieves worse OOD detection than IsoMax+$_{\text{MDS}}$ *even without using outlier exposure*.

5.3 Loss Landscapes

Fig. 2 shows the 3D surfaces and 2D contours of the losses. Considering that IsoMax outperforms SoftMax and IsoMax outperforms IsoMax, we concluded that less steep 3D inclination provides increased out-of-distribution detection performances. In other words, less concentration of 2D contours produces improved OOD detection performance.

[4]Considering that the proposed approach easily outperforms the vanilla Mahalanobis method when applied to SoftMax loss trained models (i.e., training a Mahalanobis distance-based classifier using features extracted from the neural network and using the Mahalanobis distance as score), throughout this paper, we use the term Mahalanobis approach to refer to the full Mahalanobis approach.

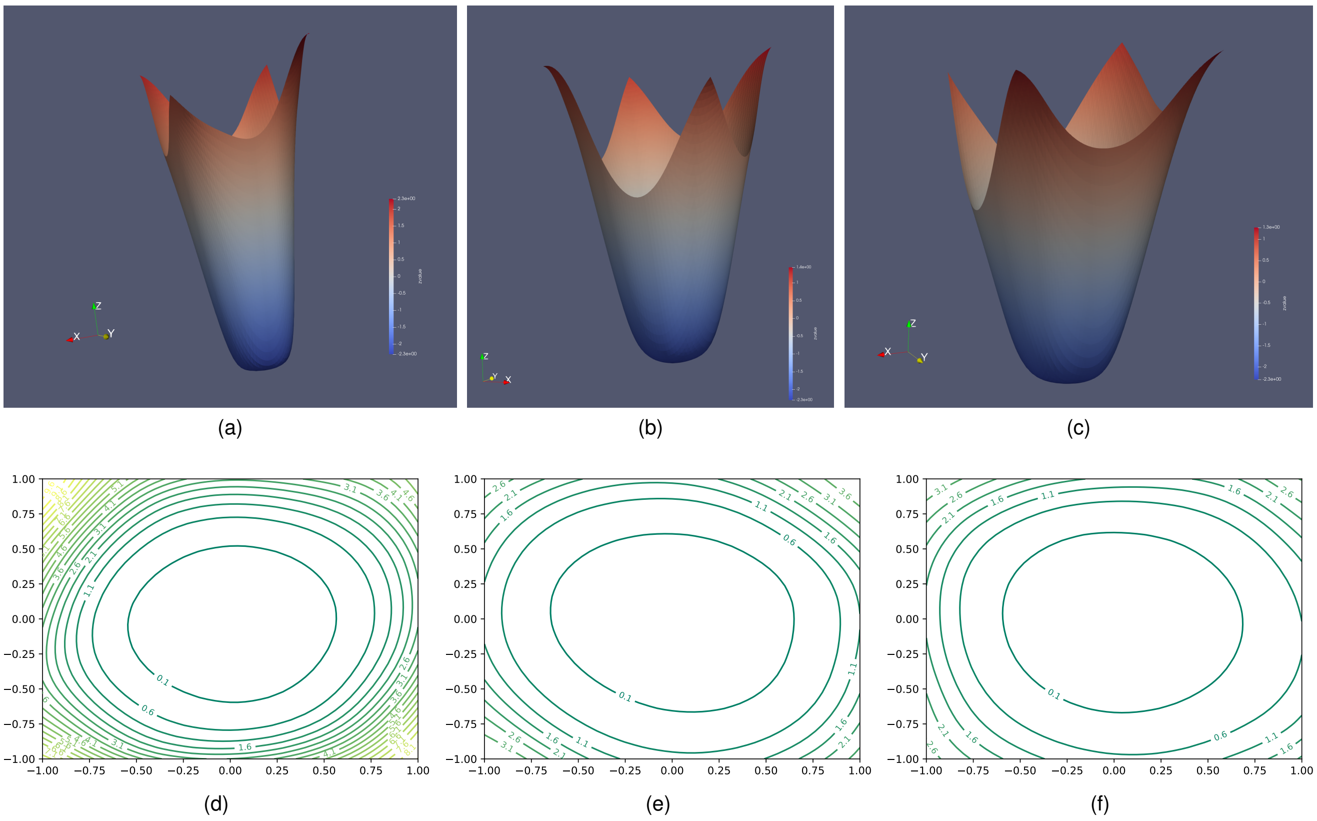


Fig. 2. 3D loss surfaces and 2D loss contours as proposed in [39]. (a) SoftMax, (b) IsoMax, and (c) IsoMaxPlus.

6 Conclusion

In this paper, we improved the IsoMax loss function by making many important modifications to it. Additionally, we proposed the *zero computational cost* minimum distance score. Experiments showed that these modifications achieve higher OOD detection performance while maintaining the desired benefits of IsoMax loss (i.e., absence of hyperparameters to tune, no reliance on additional/outlier/background data, fast and power-efficient inference, and no *classification accuracy drop*).

Similar to IsoMax loss, after training using the proposed IsoMax+ loss, we may apply inference-based approaches (e.g., ODIN, Mahalanobis, Gram matrices, outlier exposure, energy-based) to the pretrained model to eventually increase the overall OOD detection performance even more. Thus, the IsoMax+ loss is a replacement for SoftMax loss but not for OOD methods that may be applied to pretrained models, which may be used to improve even more the OOD detection performance of IsoMax+ loss pretrained networks.

There is no drawback in training a model using IsoMax+ loss instead of SoftMax loss or IsoMax loss, regardless of planning to subsequently use an inference-based OOD detection approach to further increase OOD detection performance. Therefore, instead of competitors, the OOD detection approaches that may be applied to pretrained models are actually complementary to the proposed approach [3], [12]. Hence, IsoMax+ loss constitutes a better baseline than SoftMax loss or IsoMax loss to construct future OOD detection methods.

In future work, we plan to verify whether our approach works satisfactorily when dealing with text and audio datasets. We also intend to verify the performance of our approach using transformer-based models [40], regardless of being pretrained or fine-tuned using IsoMax+ loss.

References

[1] T. DeVries and G. W. Taylor, “Leveraging uncertainty estimates for predicting segmentation quality,” *CoRR*, vol. abs/1807.00502, 2018.

[2] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” *International Conference on Learning Representations*, 2017.

[3] D. Macêdo, T. I. Ren, C. Zanchettin, A. L. I. Oliveira, and T. B. Ludermir, “Entropic out-of-distribution detection,” *International Joint Conference on Neural Networks*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9533899

[4] S. Liang, Y. Li, and R. Srikant, “Enhancing the reliability of out-of-distribution image detection in neural networks,” *International Conference on Learning Representations*, 2018.

[5] K. Lee, K. Lee, H. Lee, and J. Shin, “A simple unified framework for detecting out-of-distribution samples and adversarial attacks,” *Neural Information Processing Systems*, 2018.

[6] M. Hein, M. Andriushchenko, and J. Bitterwolf, “Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem,” *Computer Vision and Pattern Recognition*, 2018.

[7] D. Hendrycks, M. Mazeika, and T. Dietterich, “Deep anomaly detection with outlier exposure,” *International Conference on Learning Representations*, 2019.

[8] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, “Generalized ODIN: Detecting out-of-distribution image without learning from out-of-distribution data,” *Computer Vision and Pattern Recognition*, 2020.

[9] C. S. Sastry and S. Oore, "Detecting out-of-distribution examples with gram matrices," *International Conference on Machine Learning*, 2020.

[10] E. Techapanurak, M. Suganuma, and T. Okatani, "Hyperparameter-free out-of-distribution detection using cosine similarity," *Asian Conference on Computer Vision (ACCV)*, November 2020.

[11] W. Liu, X. Wang, J. D. Owens, and Y. Li, "Energy-based out-of-distribution detection," *Neural Information Processing Systems*, 2020.

[12] D. Macêdo, T. I. Ren, C. Zanchettin, A. L. I. Oliveira, and T. B. Ludermir, "Entropic out-of-distribution detection: Seamless detection of unknown examples," *IEEE Transactions on Neural Networks and Learning Systems.*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9556483

[13] S. Liang, Y. Li, and R. Srikant, "Principled detection of out-of-distribution examples in neural networks," *CoRR*, vol. abs/1706.02690, 2017.

[14] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," *International Conference on Learning Representations*, 2018.

[15] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *CoRR*, vol. abs/1802.04865, 2018.

[16] A. Shafaei, M. Schmidt, and J. J. Little, "A less biased evaluation of out-of-distribution sample detectors," *British Machine Vision Conference*, 2019.

[17] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Neural Information Processing Systems*, 2017.

[18] D. Li, D. Chen, J. Goh, and S. Ng, "Anomaly detection with generative adversarial networks for multivariate time series," *CoRR*, vol. abs/1809.04758, 2018.

[19] M. Kliger and S. Fleishman, "Novelty detection with GAN," *CoRR*, vol. abs/1802.10560, 2018.

[20] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "ImageNet: A large-scale hierarchical image database," *Computer Vision and Pattern Recognition*, 2009.

[21] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[22] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks." *International Conference on Machine Learning*, 2016.

[23] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. J. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *CoRR*, vol. abs/1902.06705, 2019.

[24] A. R. Dhamija, M. Günther, and T. E. Boult, "Reducing network agnostophobia," *Neural Information Processing Systems*, 2018.

[25] A. Vyas, N. Jammalamadaka, X. Zhu, D. Das, B. Kaul, and T. L. Willke, "Out-of-distribution detection using an ensemble of self supervised leave-out classifiers," *European Conference on Computer Vision*, 2018.

[26] Q. Yu and K. Aizawa, "Unsupervised out-of-distribution detection by maximum classifier discrepancy," *International Conference on Computer Vision*, 2019.

[27] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *Neural Information Processing Systems*, 2017.

[28] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl, "Leveraging uncertainty information from deep neural networks for disease detection," *Scientific Reports*, vol. 7, 2017.

[29] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," *Neural Information Processing Systems*, 2018.

[30] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," *International Conference on Machine Learning*, 2018.

[31] A. Subramanya, S. Srinivas, and R. V. Babu, "Confidence estimation in deep neural networks via density modelling," *CoRR*, vol. abs/1707.07013, 2017.

[32] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," *International Conference on Machine Learning*, 2016.

[33] J. Z. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax-Weiss, and B. Lakshminarayanan, "Simple and principled uncertainty estimation with deterministic deep learning via distance awareness," *Neural Information Processing Systems*, 2020.

[34] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *Computer Vision and Pattern Recognition*, 2017.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *European Conference on Computer Vision*, 2016.

[36] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Science Department, University of Toronto*, 2009.

[37] Y. Netzer and T. Wang, "Reading digits in natural images with unsupervised feature learning," *Neural Information Processing Systems*, 2011.

[38] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "LSUN: construction of a large-scale image dataset using deep learning with humans in the loop," *CoRR*, vol. abs/1506.03365, 2015.

[39] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Neural Information Processing Systems*, 2018.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Neural Information Processing Systems*, 2017.

APPENDIX D - DISTINCTION MAXIMIZATION LOSS

Distinction Maximization Loss: Efficiently Improving Classification Accuracy, Uncertainty Estimation, and Out-of-Distribution Detection Simply Replacing the Loss and Calibrating

David Macêdo[1,2], Cleber Zanchettin[1], Teresa Ludermir[1]
[1]Centro de Informática, Universidade Federal de Pernambuco, Brasil
[2]Montreal Institute for Learning Algorithms, University of Montreal, Canada
`dlm@cin.ufpe.br`, `cz@cin.ufpe.br`, `tbl@cin.ufpe.br`

Abstract

Building robust deterministic neural networks remains a challenge. On the one hand, some approaches improve out-of-distribution detection at the cost of reducing classification accuracy in some situations. On the other hand, some methods simultaneously increase classification accuracy, uncertainty estimation, and out-of-distribution detection at the expense of reducing the inference efficiency and requiring training the same model many times to tune hyperparameters. In this paper, we propose training deterministic neural networks using our DisMax loss, which works as a drop-in replacement for the usual SoftMax loss (i.e., the combination of the linear output layer, the SoftMax activation, and the cross-entropy loss). Starting from the IsoMax+ loss, we create each logit based on the distances to all prototypes rather than just the one associated with the correct class. We also introduce a mechanism to combine images to construct what we call fractional probability regularization. Moreover, we present a fast way to calibrate the network after training. Finally, we propose a composite score to perform out-of-distribution detection. Our experiments show that DisMax usually outperforms current approaches simultaneously in classification accuracy, uncertainty estimation, and out-of-distribution detection while maintaining deterministic neural network inference efficiency and avoiding training the same model repetitively for hyperparameter tuning. The code to reproduce the results is available.[1]

1 Introduction

Deep neural networks have been used for classification in many applications. However, improving the robustness of such systems remains a significant challenge. Classification accuracy, uncertainty estimation, and out-of-distribution (OOD) detection comprise three essential points regarding measuring the robustness of deep learning approaches.

On the one hand, some OOD detection approaches do not address uncertainty estimation or produce diminished classification accuracy in some cases [24, 7]. These solutions also require changing the training of the last layer by removing its weight decay to work correctly. Therefore, they do not work as straightforward drop-in replacements for the SoftMax loss (i.e., the combination of the linear output layer, the SoftMax activation, and the cross-entropy loss [15]). On the other hand, some recent approaches that address both OOD detection and uncertainty estimation require hyperparameter tuning and reduce the inference efficiency compared to pure deterministic neural

[1]https://github.com/dlmacedo/distinction-maximization-loss

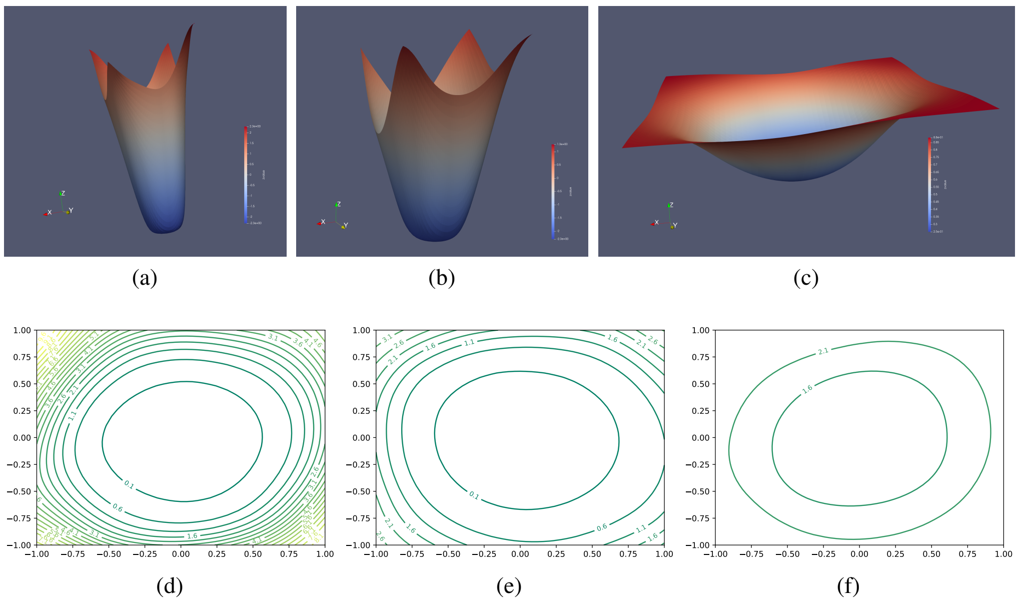


Figure 1: **Loss Surface Study.** 3D loss surfaces and 2D loss contours as proposed in [12]. Loss landscapes for ResNet34 trained on CIFAR10. (a, d) SoftMax; (b, e) IsoMax+; and (c, f) DisMax. Considering that IsoMax+ outperforms SoftMax and DisMax outperforms IsoMax+, a less steep 3D inclination (i.e. a lower 2D contour concentration) provides increased robustness.

networks [10, 25, 14]. Therefore, simultaneously increasing classification accuracy, OOD detection, and uncertainty estimation performances while maintaining inference efficiency poses a challenge, mainly if we also desire to avoid training the same architecture many times to tune hyperparameters.

Recently, so-called IsoMax loss variants have been proposed [17, 16, 18]. They increase the OOD detection performance without reducing the inference efficiency compared to pure deterministic deep neural networks trained using the usual SoftMax loss. Moreover, they do not require repetitive training of the same model for hyperparameter tuning. However, they increase neither the classification accuracy nor uncertainty estimation.

Contributions In this paper, starting from IsoMax+ loss [18], we construct the Distinction Maximization (DisMax) loss. Our main contributions are the following. First, we create the *enhanced* logits (logits+) by using *all* feature-prototype distances rather than just the feature-prototype distance to the correct class. Second, we introduce the *fractional* probability regularization (FPR) by minimizing the Kullback–Leibler (KL) divergence between the output probability distribution associated with a *compound* image and a target probability distribution containing *fractional* rather than integer probabilities. We call DisMax dagger (DisMax†) the variant of our loss without using the FPR. Third, we construct a *composite* score for OOD detection that combines three components: the maximum logit+, the *mean* logit+, and the entropy of the network output. Fourth, we present a simple and fast temperature-scaling procedure that allows DisMax trained models to produce a high-performance uncertainty estimation.

Like IsoMax+, DisMax works as a drop-in replacement for SoftMax loss. Moreover, only a *single* neural network training is required to use the proposed solution, as it avoids hyperparameter tuning. Furthermore, the trained models keep deterministic neural network inference efficiency. Finally, we show experimentally that to obtain improved robustness, we need to construct losses with less steep 3D landscapes, as showed in Fig. 1.

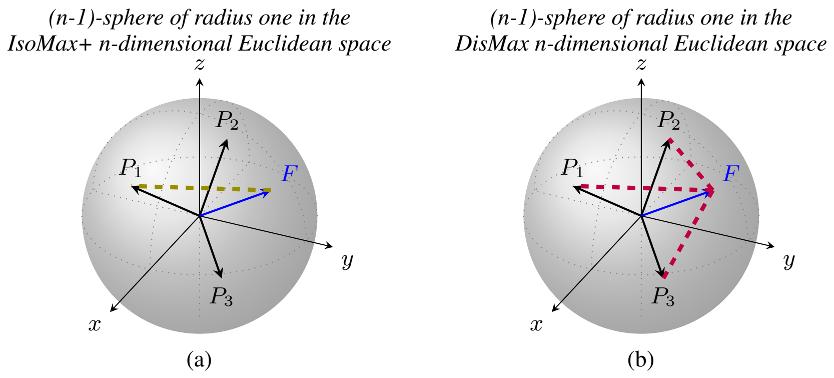

(n-1)-sphere of radius one in the IsoMax+ n-dimensional Euclidean space
(n-1)-sphere of radius one in the DisMax n-dimensional Euclidean space
z
P_2
P_1
F
y
P_3
x
(a)
(b)

Figure 2: **All-Distances-Aware Logits, Enhanced Logits, or Logits+.** The illustration presents the difference between IsoMax+ [18] and DisMax with respect to *logit formation*. P_1, P_2, and P_3 represent prototypes of classes 1, 2, and 3, respectively. F denotes the feature associated with a given image. Like all current losses, IsoMax+ constructs *each* logit associated with F considering its distance from a *single* prototype (olive dashed line). In contrast, DisMax loss builds *each* logit associated with F considering its distance from *all* prototypes (purple dashed lines). In this paper, we use the terms *all-distances-aware* logits, *enhanced* logits, or logits+ indistinctly.

2 Distinction Maximization Loss

All-Distances-Aware Logits In IsoMax loss variants (e.g., IsoMax and IsoMax+), logits are formed from distances and are commonly used to calculate the score to perform OOD detection. Hence, it is essential to build logits that contain semantic information relevant to separating in-distribution (ID) from OOD during inference. IsoMax+ uses the *isometric* distances [18]. In IsoMax+, the logits are simply the negatives of the isometric distances. We have two motivations to add the *mean* isometric distance considering *all* prototypes to the isometric distance associated with *each* class to construct what we call *all-distances-aware* logits, *enhanced* logits, or logits+.

First, considering that IsoMax+ is an isotropic loss, the pairwise distances between the prototypes and ID examples are forced to become increasingly smaller. Therefore, after training, it is reasonable to believe that ID feature-prototype distances are, on average, smaller than the distances from the prototypes to OOD samples, which were not forced to be closer to the prototypes. Hence, adding the mean distance to the logits used in IsoMax+ can help distinguish between ID and OOD more effectively. Second, taking *all* feature-prototype distances to compose the logits makes them a more stable source of information to perform OOD detection (Fig. 2).

$$\overset{\text{all-distances-aware logit for the j-th class}}{\boxed{L_+^j}} = -\left(\underbrace{\boxed{D_I^j}}_{\text{isometric distance to the j-th class prototype}} + \overbrace{\boxed{\frac{1}{N}\sum_{n=1}^{N} D_I^n}}^{\text{mean isometric distance to all prototypes}} \right) \tag{1}$$

$$\overset{\text{predicted probability distribution}}{\boxed{\mathcal{P}_{\mathsf{DisMax}}(y^{(i)}|\boldsymbol{x})}} = \frac{\exp(\overset{\text{all-distances-aware logit for the i-th class}}{\boxed{L_+^i}}/T)}{\sum\limits_{j=1}^{N}\exp(\underbrace{\boxed{L_+^j}}_{\text{all-distances-aware logit for the j-th class}}/T)} \tag{2}$$

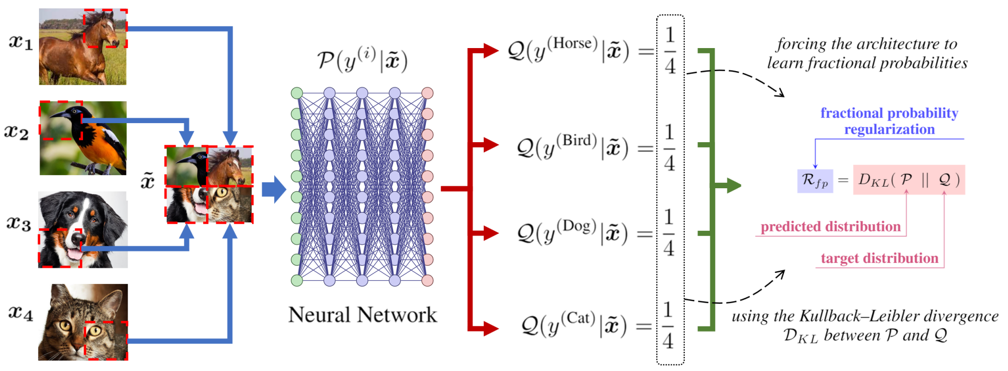

x_1
x_2
x_3
x_4
$\tilde{x}$
$\mathcal{P}(y^{(i)}|\tilde{x})$
Neural Network
$\mathcal{Q}(y^{(\text{Horse})}|\tilde{x}) = \frac{1}{4}$
$\mathcal{Q}(y^{(\text{Bird})}|\tilde{x}) = \frac{1}{4}$
$\mathcal{Q}(y^{(\text{Dog})}|\tilde{x}) = \frac{1}{4}$
$\mathcal{Q}(y^{(\text{Cat})}|\tilde{x}) = \frac{1}{4}$
forcing the architecture to learn fractional probabilities
fractional probability regularization
$\mathcal{R}_{fp} = \mathcal{D}_{KL}(\mathcal{P} \,||\, \mathcal{Q})$
predicted distribution
target distribution
using the Kullback–Leibler divergence $\mathcal{D}_{KL}$ between $\mathcal{P}$ and $\mathcal{Q}$

Figure 3: **Fractional Probability Regularization.** We use images composed of patches of four randomly selected training examples. The KL divergence regularization term forces the network to predict *fractional* probabilities on compound images.

Therefore, we consider an input $\boldsymbol{x}$ and a network that performs a transformation $\boldsymbol{f_\theta}(\boldsymbol{x})$. We also consider $\boldsymbol{p}_{\boldsymbol{\phi}}^j$ to be the learnable prototype associated with class j. Moreover, considering that $\|\boldsymbol{v}\|$ represents the 2-norm of a vector $\boldsymbol{v}$, and $\widehat{\boldsymbol{v}}$ represents the 2-norm normalization of $\boldsymbol{v}$, we can write the *isometric distance* relative to class j as $D_I^j = |d_s| \, \|\widehat{\boldsymbol{f_\theta}(\boldsymbol{x})} - \widehat{\boldsymbol{p}_{\boldsymbol{\phi}}^j}\|$, where $|d_s|$ represents the absolute value of the *learnable* scalar called distance scale [18]. Finally, we can write the proposed *enhanced* logit for class j using the equation (1). N is the number of classes. Probabilities are given by the equation (2), where T is the temperature. For the rest of this paper, distance means *isometric* distance.

Fractional Probability Regularization We often train neural networks using *unitary* probabilities. Indeed, the usual cross-entropy loss forces a *probability equal to one* on a given training example. Consequently, we commonly train neural networks by providing a tiny proportion of points in the learning manifold. Hence, we propose what we call the *fractional* probability regularization (FPR). The idea is to force the network to learn more diverse points in the learning manifold. Consequently, we confront target and predicted probability distributions also on *fractional* probability values rather than only *unitary* probability manifold points.

$$\overbrace{\mathcal{Q}_{\text{Target}}(y^{(i)}|\tilde{\boldsymbol{x}})}^{\text{desired probability distribution for the compound image}} = \underbrace{\tfrac{1}{4}\sum_{m=1}^{4} \delta[y^{(i)} - y^{(j^m)}]}_{\text{sum one quarter to the probability of each class in the compound image}} \tag{3}$$

$$\overbrace{\mathcal{L}_{\text{DisMax}}}^{\text{loss function}} = -\log^{*}\left(\frac{\exp(E_s \, \overbrace{L_+^k}^{\text{enhanced logit for the correct class k}})}{\sum_j \exp(E_s \, \underbrace{L_+^j}_{\text{enhanced logit for the j-th class}})} \right) + \underbrace{D_{KL}(\mathcal{P}_{\text{DisMax}}(y|\tilde{\boldsymbol{x}}) \,||\, \mathcal{Q}_{\text{Target}}(y|\tilde{\boldsymbol{x}}))}_{\text{fractional probability regularization}} \tag{4}$$

[*]The probability (i.e., the expression between the outermost parentheses) and logarithm operations are computed sequentially and separately for optimal OOD detection performance [17].

Therefore, our batch is divided into two halves. In the first half, we use the regular *unitary* probability training. For the second batch, we construct images specifically composed of patches of four others (Fig. 3). We construct our target probability distribution Q for those images by adding a quarter probability for each class corresponding to a patch of the compound image. Finally, we minimize the KL divergence regularization between our predicted and target probability distributions in the second half. These procedures do not increase training memory size requirements. Considering δ the Kronecker delta function, equation (3) represents the FPR in math terms. By combining the *enhanced* logits and the FPR, equation (4) presents the mathematical expression for the DisMax loss.

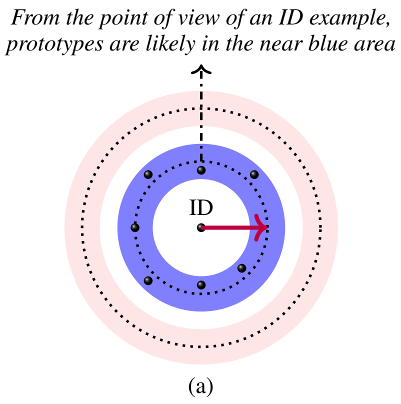
We recognize a similarity between CutMix [27] and FPR: both are based on the combination of images to create compound data. However, we identify many differences. CutMix combines two images, while FPR combines four images. Moreover, the combination procedure is entirely different. In CutMix, a portion of an image is replaced by a patch of *variable* size, format, and position that comes from another image. In FPR, patches of the *same* size, format, and *predefined* positions from four different images are combined into a single one. This simplification introduced by FRP allowed us to combine four images instead of only two. Indeed, trying to extend CutMix by replacing portions of an image with patches from three others while allowing *random* sizes, shapes, and positions produces *patch superpositions*, making *the calculation of the pairwise ratio of the areas of the superposed patches extremely hard*. Therefore, the mentioned simplification made it possible to simultaneously combine four rather than only two images in addition to avoiding the beta distribution and the α hyperparameter.

While CutMix is applied randomly to some batches with probability p, FPR is applied to half of each batch, avoiding loss or gradient oscillations. CutMix neither creates a target distribution containing *fractional* probabilities nor forces the predicted probabilities to follow it by minimizing the KL divergence between them. Indeed, CutMix does *not* use the KL divergence at all. CutMix calculates the regular cross-entropy loss of the compound image considering the labels of the original images and takes a linear interpolation between the resulting loss values weighted by the ratio of the areas of the patch and the remaining image. While CutMix operates on losses, FPR operates directly on probabilities *before* calculating loss values. The concept of *fractional* probabilities is not even present in CutMix. Unlike CutMix, the mentioned procedure can be easily expanded to combine even more than four images. Finally, CutMix *increases the training time* and *presents hyperparameters* [27]. As FPR is an *integral part of the DisMax loss*, it is *transparent* for the user of the provided code.

Max-Mean Logit Entropy Score For OOD detection, we propose a score composed of three parts. The first part is the maximum logit+. The second part is the *mean* logit+. Incorporating the mean value of the logits into the score is an *independent procedure relative to the logit formation*. It can be applied regardless of the type of logit (e.g., usual or enhanced) used during training. Finally, we subtract the entropy calculated considering the probabilities of the neural network output. We call this *composite* score *Max-Mean Logit Entropy Score* (MMLES). It is given by equation (5). We call MMLES a *composite* score because it is formed by the sum of many other scores.

$$\overbrace{\mathcal{S}_{\text{MMLES}}}^{\text{max-mean logit entropy score}} = \underbrace{\max_j (L_+^j)}_{\text{maximum logit+}} + \underbrace{\frac{1}{N}\sum_{n=1}^{N} L_+^n}_{\text{mean logit+}} - \overbrace{\mathcal{H}(\mathcal{P}_{\text{DisMax}})}^{\text{entropy}} \tag{5}$$

Temperature Calibration Unlike the usual SoftMax loss, the IsoMax loss variants produce *under-estimated* probabilities *to obey the Maximum Entropy Principle* [17, 19, 16, 18]. Therefore, *we need to perform a temperature calibration after training to improve the uncertainty estimation*. To find an optimal temperature, we used the L-BFGS-B algorithm with approximate gradients and bounds equal to 0.001 and 100 [1, 29, 21] for expected calibration error (ECE) minimization. The mentioned calibration procedure takes only a few seconds using the provided code.


From the point of view of an ID example, prototypes are likely in the near blue area
ID
(a)

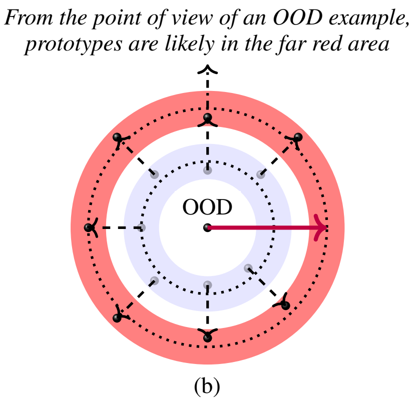

From the point of view of an OOD example, prototypes are likely in the far red area
OOD
(b)

Figure 4: **Max-Mean Logit Entropy Score (MMLES).** In addition to the maximum logit and the negative entropy, the MMLES incorporates the *mean* logit+. We empirically observed that the prototypes are generally closer to ID samples than OOD samples, which is true *regardless of whether the ID sample belongs to the class of the considered prototype*. Hence, incorporating this *all-distances-aware* information increases the OOD detection performance (see Table 1 and Fig. 5).

3 Experiments

To allow standardized comparison, we used the datasets, training procedures, and metrics that were established in [6] and used in many subsequent papers [13, 11, 5]. We trained many 100-layer DenseNetBCs with growth rate $k = 12$ (i.e., 0.8M parameters) [8], 34-layer ResNets [4], and 28-layer WideResNets (widening factor $k = 10$) [28] on the CIFAR10 [9] and CIFAR100 [9] datasets with SoftMax, IsoMax+, and DisMax losses using the same procedures (e.g., initial learning rate, learning rate schedule, weight decay). We used DisMax† for DenseNetBC100 trained on CIFAR10 because this is a very small model, and the mentioned dataset has too many examples per class; therefore, no augmentation is needed. For all other cases, we used DisMax.

We used stochastic gradient descent (SGD) with the Nesterov moment equal to 0.9 with a batch size of 64 and an initial learning rate of 0.1. The weight decay was 0.0001, and we did not use dropout. We trained during 300 epochs. We used a learning rate decay rate equal to ten applied in epoch numbers 150, 200, and 250. We used resized images from TinyImageNet [2], the Large-scale Scene UNderstanding dataset (LSUN) [26], CIFAR10, CIFAR100, and SVHN [23] to create out-of-distribution samples. We added these out-of-distribution images to the validation sets of the ID data to form the test sets and evaluate the OOD detection performance. We evaluated the accuracy (ACC) to assess classification performance. Like IsoMax+, when using DisMax, *we only train once*, as no hyperparameter tuning is required. We evaluated the OOD detection performance using the area under the receiver operating characteristic curve (AUROC), the area under the precision-recall curve (AUPR), and the true negative rate at a 95% true positive rate (TNR@TPR95). We used the expected calibration error (ECE) [22, 3, 20] for uncertainty estimation performance. The results are the mean and standard deviation of five runs. Two methods are considered to produce the same performance if their mean performance difference is less than the sum of the error margins.

Ablation Study Table 1 shows that logits+ and FPR often improve the accuracy and OOD detection performance compared to IsoMax+ even when using MDS. Moreover, it also shows that replacing MDS with the *composite* score MMLES often increases OOD detection. These conclusions are essentially true regardless of the model, in-distribution, and (near, far, and very far) out-of-distribution.

Finally, we performed experiments *combining IsoMax+ with CutMix* by training ResNet34 models on CIFAR10 and CIFAR100. However, adding CutMix to IsoMax+ did *not* increase the OOD detection performance significantly. Sometimes, it even *decreased*. Therefore, *DisMax easily outperformed IsoMax+ even when the latter was combined with CutMix.*

Table 1: **Ablation Study.** MPS means Maximum Probability Score (i.e., the standard for SoftMax loss). MDS indicates Minimum Distance Score (i.e., the standard for IsoMax+ loss). MMLES means Max-Mean Logit Entropy Score (i.e., the standard for DisMax loss for (very) far OOD detection). We used MPS for near OOD detection for DisMax, as this score provided the best results in this particular case. We emphasize that the MPS for DisMax is based on *logits+ rather than usual logits*. The best performances are bold. All results can be reproduced using the provided code.

CIFAR10

Model	Method	Score	Out-of-Distribution Detection			
			Near	Far		Very Far
			CIFAR100	TinyImageNet	LSUN	SVHN
		MPS,MDS MPS/MMLES	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]
DenseNetBC100 (small size)	SoftMax (baseline) [6]	MPS	39.5±2.1	53.1±7.8	62.1±6.2	41.2±3.6
	IsoMax+ [18]	MDS	**57.3±1.1**	86.9±0.4	91.4±0.3	96.4±0.6
	DisMax† (ours)	MDS	**56.2±0.5**	**88.0±0.3**	**92.2±0.3**	97.5±0.3
		MPS/MMLES	54.2±1.0	**89.0±1.0**	**92.4±1.1**	**98.3±0.3**
ResNet34 (medium size)	SoftMax (baseline) [6]	MPS	40.0±1.6	46.4±4.9	53.6±4.7	44.1±9.3
	IsoMax+ [18]	MDS	55.1±1.3	71.0±6.4	81.5±4.4	82.4±8.8
	DisMax (ours)	MDS	**60.4±0.7**	**92.0±1.5**	97.2±0.4	**91.1±2.9**
		MPS/MMLES	**60.0±0.5**	**93.3±1.1**	**98.0±0.3**	**91.2±2.7**
WideResNet2810 (big size)	SoftMax (baseline) [6]	MPS	44.9±0.7	53.4±3.3	59.2±3.6	50.1±5.2
	IsoMax+ [18]	MDS	61.5±0.4	80.2±4.2	87.4±3.0	**96.3±1.4**
	DisMax (ours)	MDS	60.2±1.3	**98.4±0.4**	**99.4±0.1**	**93.8±1.7**
		MPS/MMLES	**62.9±0.5**	**98.6±0.2**	**99.5±0.1**	92.8±2.0

CIFAR100

Model	Method	Score	Out-of-Distribution Detection			
			Near	Far		Very Far
			CIFAR10	TinyImageNet	LSUN	SVHN
		MPS,MDS MPS/MMLES	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]	TNR@95TPR (%) [↑]
DenseNetBC100 (small size)	SoftMax (baseline) [6]	MPS	17.6±1.1	18.1±1.7	18.7±2.0	19.8±2.9
	IsoMax+ [18]	MDS	17.2±0.7	71.6±6.5	66.8±9.4	**67.1±3.0**
	DisMax (ours)	MDS	16.6±0.6	97.7±0.3	98.5±0.4	57.9±3.6
		MPS/MMLES	**22.1±1.1**	**99.0±0.5**	**99.4±0.3**	**66.6±2.6**
ResNet34 (medium size)	SoftMax (baseline) [6]	MPS	19.4±0.5	20.6±2.4	21.3±3.4	17.1±5.0
	IsoMax+ [18]	MDS	18.0±0.7	43.3±4.3	41.5±5.7	43.6±3.5
	DisMax (ours)	MDS	20.8±0.4	79.9±1.5	81.5±1.4	43.7±1.6
		MPS/MMLES	**22.0±0.5**	**85.4±1.7**	**86.4±1.3**	**48.5±2.0**
WideResNet2810 (big size)	SoftMax (baseline) [6]	MPS	21.8±0.7	26.7±5.9	28.7±6.7	15.8±5.5
	IsoMax+ [18]	MDS	19.0±0.7	66.9±3.9	67.9±3.3	61.8±1.9
	DisMax (ours)	MDS	22.4±0.2	92.3±1.3	95.2±0.4	56.8±1.8
		MPS/MMLES	**24.6±0.3**	**96.3±1.2**	**97.8±0.9**	**65.6±1.2**

Classification, Efficiency, Uncertainty, and OOD Detection Results Table 2 compares DisMax with major approaches such as Scaled Cosine [24], GODIN [7], Deep Ensemble [10], DUQ [25], and SNGP [14] regarding classification accuracy, inference efficiency, uncertainty estimation, and (near, far, and very far) out-of-distribution detection. Unlike other approaches, DisMax is as inference efficient as a trivially trained neural network using the usual SoftMax loss. Moreover, using DisMax, *we only train once* the neural network, as no hyperparameter tuning is needed. Furthermore, DisMax often outperforms other approaches simultaneously in all evaluated metrics.

Additional Analyses Fig. 5 show the distribution of *mean* logits+ under some scenarios. We see that prototypes are, on average, usually closer to in-distribution examples than out-of-distribution examples, which explains why the *mean* enhanced logit improves OOD detection performance when combined with the maximum logit+ and the negative entropy to compose the MMLES. In other words, even prototypes *that are not associated with the class of a given in-distribution example* are usually closer to it than they are to out-of-distribution examples.

Table 2: **Classification, Efficiency, Uncertainty, and OOD Detection Results.** In this table, efficiency represents the inference speed (i.e., the inverse of the inference delay) calculated as a percentage of the performance of a single deterministic neural network trivially trained. For a fair comparison, we also calibrated the temperature of the SoftMax loss and IsoMax+ loss approaches using the same procedure that we defined for DisMax loss. Considering that input preprocessing can be applied indistinctly to improve the OOD detection performance of all methods compared [7] (at the cost of making their inferences approximately four times less efficient [16]), unless explicitly mentioned otherwise, all results are presented without using input preprocessing. The results worse than the baseline or most of the other approaches are shown in red. The methods that present the best performances are bold. Results for Scaled Cosine are from Scaled Cosine paper [24]. Results for GODIN are from GODIN paper [7]. Results for Deep Ensemble, DUQ, and SNGP are from SNGP paper [14]. All other results can be reproduced using the provided code.

CIFAR10

Model	Method	Classification	Inference	Uncertainty Estimation	Out-of-Distribution Detection			
					Near	Far		Very Far
					CIFAR100	TinyImageNet	LSUN	SVHN
		ACC (%) [↑]	Efficiency (%) [↑]	ECE [↓]	AUPR (%) [↑]	AUROC (%) [↑]	AUROC (%) [↑]	AUPR (%) [↑]
DenseNetBC100 (small size)	SoftMax (baseline) [6]	**95.2±0.1**	**100.0**	**0.0043±0.0008**	86.2±0.5	92.9±1.6	94.7±0.9	93.7±3.3
	Scaled Cosine [24]	94.9±0.1	**100.0**	-	-	**98.8±0.3**	**99.2±0.2**	-
	GODIN with preprocessing[1] [7]	95.0±0.1	26.0	-	-	**99.1±0.1**	**99.4±0.1**	-
	IsoMax+ [18]	**95.1±0.1**	**100.0**	**0.0043±0.0012**	**90.4±0.3**	97.6±0.9	98.3±0.5	99.7±0.1
	DisMax† (ours)	**95.1±0.1**	**100.0**	**0.0045±0.0021**	**90.0±0.2**	98.0±0.5	98.4±0.3	**99.9±0.1**
ResNet34 (medium size)	SoftMax (baseline) [6]	95.6±0.1	**100.0**	**0.0060±0.0013**	85.3±0.4	89.7±2.8	92.4±1.6	94.9±1.0
	GODIN [7]	95.1±0.1	**100.0**	-	-	95.6±0.5	97.6±0.2	-
	IsoMax+ [18]	95.5±0.1	**100.0**	0.0133±0.0177	**90.1±0.3**	95.1±1.0	96.9±0.6	**98.7±0.6**
	DisMax (ours)	**96.7±0.2**	**100.0**	**0.0058±0.0008**	**90.3±0.2**	**98.3±0.3**	**99.5±0.1**	**99.1±0.3**
WideResNet2810 (big size)	SoftMax (baseline) [6]	96.2±0.1	**100.0**	**0.0038±0.0005**	87.5±0.3	92.6±0.9	94.0±0.7	95.3±0.9
	Deep Ensemble [10]	96.6±0.1	10.3	0.0100±0.0010	88.8±1.0	-	-	96.4±1.0
	DUQ[2] [25]	94.7±0.1	45.0	0.0340±0.0020	85.4±1.0	-	-	97.3±1.0
	SNGP[2] [14]	95.9±0.1	62.5	0.0180±0.0010	90.5±1.0	-	-	99.0±1.0
	Scaled Cosine [24]	95.7±0.1	**100.0**	-	-	97.7±0.7	98.6±0.3	-
	IsoMax+ [18]	96.0±0.1	**100.0**	0.0107±0.0166	**91.8±0.1**	96.6±0.6	97.7±0.4	**99.7±0.3**
	DisMax (ours)	**97.0±0.1**	**100.0**	**0.0043±0.0008**	90.1±0.3	**99.7±0.1**	**99.9±0.1**	**99.3±0.3**

CIFAR100

Model	Method	Classification	Inference	Uncertainty Estimation	Out-of-Distribution Detection			
					Near	Far		Very Far
					CIFAR10	TinyImageNet	LSUN	SVHN
		ACC (%) [↑]	Efficiency (%) [↑]	ECE [↓]	AUPR (%) [↑]	AUROC (%) [↑]	AUROC (%) [↑]	AUPR (%) [↑]
DenseNetBC100 (small size)	SoftMax (baseline) [6]	77.3±0.4	**100.0**	0.0155±0.0026	71.3±0.8	71.8±2.2	73.1±2.4	87.5±1.5
	Scaled Cosine [24]	75.7±0.1	**100.0**	-	-	97.8±0.5	97.6±0.8	-
	GODIN with preprocessing[1] [7]	75.9±0.1	24.0	-	-	98.6±0.2	98.7±0.0	-
	IsoMax+ [18]	76.9±0.3	**100.0**	**0.0108±0.0017**	71.3±0.4	95.1±1.1	94.2±1.7	**97.4±0.6**
	DisMax (ours)	**79.4±0.2**	**100.0**	0.0154±0.0006	**74.4±0.2**	**99.8±0.1**	**99.9±0.1**	96.4±0.8
ResNet34 (medium size)	SoftMax (baseline) [6]	77.7±0.3	**100.0**	0.0268±0.0015	73.3±0.1	79.0±2.1	79.6±1.7	86.3±3.3
	GODIN [7]	75.8±0.2	**100.0**	-	-	91.8±1.1	92.0±0.7	-
	GODIN with dropout[2] [7]	77.2±0.1	**100.0**	-	-	87.0±1.1	87.0±2.2	-
	IsoMax+ [18]	76.5±0.3	**100.0**	0.0190±0.0025	72.1±0.4	89.7±1.0	89.8±1.3	**94.5±0.6**
	DisMax (ours)	**80.6±0.3**	**100.0**	**0.0116±0.0014**	**74.2±0.6**	**97.6±0.5**	**97.7±0.6**	**94.8±1.0**
WideResNet2810 (big size)	SoftMax (baseline) [6]	79.9±0.2	**100.0**	0.0272±0.0032	75.4±0.5	81.7±2.3	82.7±2.2	86.0±2.6
	Deep Ensemble [10]	80.2±0.1	12.3	0.0210±0.0040	78.0±1.0	-	-	88.8±1.0
	DUQ[2] [25]	78.5±0.1	79.9	0.1190±0.0010	73.2±1.0	-	-	87.8±1.0
	SNGP[2] [14]	79.9±0.1	74.9	0.0250±0.0120	**80.1±1.0**	-	-	92.3±1.0
	Scaled Cosine [24]	78.5±0.3	**100.0**	-	-	95.8±0.7	95.2±0.8	-
	IsoMax+ [18]	79.5±0.1	**100.0**	0.0188±0.0016	73.0±0.8	94.2±2.1	94.6±2.0	**96.7±1.7**
	DisMax (ours)	**83.0±0.1**	**100.0**	**0.0143±0.0027**	76.0±1.0	**99.4±0.2**	**99.6±0.1**	**97.0±1.5**

[1]Requires reserving training data for validation and performing hyperparameter tuning.
[2]Requires training the same neural network many times for validating hyperparameters.

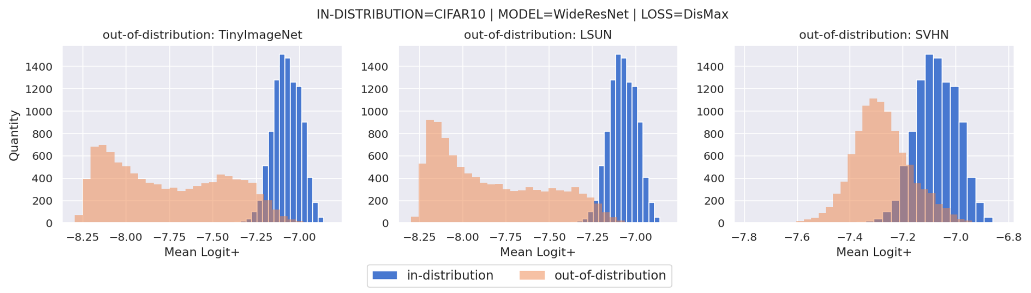


Figure 5: **Additional Analyses.** In the feature space, the mean distance from an in-distribution image to *all* prototypes is usually smaller than the mean distance from an out-of-distribution image to the *all* prototypes. For example, consider a given class present in CIFAR10. This figure shows that even prototypes associated with classes *other than the selected class* are usually closer to images of the assumed class (in-distribution in blue) than images that do not belong to CIFAR10 at all (out-of-distributions in orange). This explains why the *mean value* of logits+ considering all prototypes contributes to the OOD detection performance. Therefore, not only the distance to the nearest prototype is used in the mentioned task.

4 Related Works

In 2019, on the one hand, IsoMax [19] proposed a *non-squared Euclidean distance last layer to address out-of-distribution detection* in an *end-to-end trainable way* (i.e. no feature extraction). On the other hand, Scaled Cosine [24] proposed using a *cosine distance*. Although the scale factor in IsoMax is a *constant* scalar called the entropy scale, Scaled Cosine requires the addition of a *block of layers* to learn the scale factor. This is made up of an exponential function, batch normalization, and a linear layer that has the feature layer as input. Moreover, to present high performance, it is necessary to avoid applying weight decay to this *extra learning block*. We believe that this additional learning block, which adds an ad hoc linear layer in the final of the neural network, may make the solution prone to *overfitting* and explain the classification accuracy drop mentioned by the authors.

In 2020, GODIN [7] cited and was heavily inspired by Scaled Cosine. GODIN kept the *extra learning block* to learn the scale factor and also avoided applying weight decay to it. In addition to the usual affine transformation and cosine distance from Scaled Cosine, it presents a variant that uses a Euclidean distance-based last layer, similar to IsoMax. The major contribution of GODIN was to allow using the input preprocessing introduced in ODIN without the need for out-of-distribution data. However, input preprocessing increases the inference latency (i.e., reduces the inference efficiency) approximately four times [16].

Moreover, SNGP [14] cited, followed, and improved the idea introduced by IsoMax in 2019: A *distance-based output layer* for OOD detection. In a similar direction, DUQ [25] also proposed a modified *distance-based loss to address OOD detection*. However, unlike IsoMax variants (e.g., IsoMax, IsoMax+, and DisMax), SNGP and DUQ produce inferences not as efficient as those produced by a deterministic neural network [14]. Moreover, they require training the neural network many times for hyperparameter tuning.

5 Conclusion

In this work, we proposed DisMax by improving the IsoMax+ with the *enhanced* logits and the *Fractional Probability Regularization*. We also presented a novel *composite* score called MMLES for OOD detection by combining the maximum logit+, the *mean* logit+, and the negative entropy of the network output. We present a *simple and fast temperature scaling procedure* performed after training that makes DisMax produce a high-performance uncertainty estimation. Our experiments showed that the proposed method commonly outperforms the current approaches simultaneously in classification accuracy, inference efficiency, uncertainty estimation, and out-of-distribution detection.

References

[1] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 1995.

[2] Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. ImageNet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition*, 2009.

[3] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *International Conference on Machine Learning*, 2017.

[4] He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. *European Conference on Computer Vision*, 2016.

[5] Hein, M., Andriushchenko, M., and Bitterwolf, J. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *Computer Vision and Pattern Recognition*, 2018.

[6] Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations*, 2017.

[7] Hsu, Y.-C., Shen, Y., Jin, H., and Kira, Z. Generalized ODIN: Detecting out-of-distribution image without learning from out-of-distribution data. *Computer Vision and Pattern Recognition*, 2020.

[8] Huang, G., Liu, Z., Maaten, L. v. d., and Weinberger, K. Q. Densely connected convolutional networks. *Computer Vision and Pattern Recognition*, 2017.

[9] Krizhevsky, A. Learning multiple layers of features from tiny images. *Science Department, University of Toronto*, 2009.

[10] Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Neural Information Processing Systems*, 2017.

[11] Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Neural Information Processing Systems*, 2018.

[12] Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. *Neural Information Processing Systems*, 2018.

[13] Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. *International Conference on Learning Representations*, 2018.

[14] Liu, J. Z., Lin, Z., Padhy, S., Tran, D., Bedrax-Weiss, T., and Lakshminarayanan, B. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Neural Information Processing Systems*, 2020.

[15] Liu, W., Wen, Y., Yu, Z., and Yang, M. Large-margin softmax loss for convolutional neural networks. *International Conference on Machine Learning*, 2016.

[16] Macêdo, D., Ren, T. I., Zanchettin, C., Oliveira, A. L. I., and Ludermir, T. B. Entropic out-of-distribution detection: Seamless detection of unknown examples. *IEEE Transactions on Neural Networks and Learning Systems.*, 2021.

[17] Macêdo, D., Ren, T. I., Zanchettin, C., Oliveira, A. L. I., and Ludermir, T. B. Entropic out-of-distribution detection. *International Joint Conference on Neural Networks*, 2021.

[18] Macêdo, D. and Ludermir, T. Enhanced isotropy maximization loss: Seamless and high-performance out-of-distribution detection simply replacing the softmax loss. *CoRR*, abs/2105.14399, 2021.

[19] Macêdo, D., Ren, T. I., Zanchettin, C., Oliveira, A. L. I., and Ludermir, T. Entropic out-of-distribution detection (first version). *CoRR*, abs/1908.05569v1, 2019.

[20] Minderer, M., Djolonga, J., Romijnders, R., Hubis, F., Zhai, X., Houlsby, N., Tran, D., and Lucic, M. Revisiting the calibration of modern neural networks. *Neural Information Processing Systems*, 2021.

[21] Morales, J. L. and Nocedal, J. Remark on "algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Trans. Math. Softw.*, 2011.

[22] Naeini, M. P., Cooper, G. F., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. *AAAI Conference on Artificial Intelligence*, 2015.

[23] Netzer, Y. and Wang, T. Reading digits in natural images with unsupervised feature learning. *Neural Information Processing Systems*, 2011.

[24] Techapanurak, E., Suganuma, M., and Okatani, T. Hyperparameter-free out-of-distribution detection using cosine similarity. *Asian Conference on Computer Vision*, November 2020.

[25] van Amersfoort, J. R., Smith, L., Teh, Y. W., and Gal, Y. Simple and scalable epistemic uncertainty estimation using a single deep deterministic neural network. *International Conference on Machine Learning*, 2020.

[26] Yu, F., Zhang, Y., Song, S., Seff, A., and Xiao, J. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015.

[27] Yun, S., Han, D., Chun, S., Oh, S. J., Yoo, Y., and Choe, J. Cutmix: Regularization strategy to train strong classifiers with localizable features. *International Conference on Computer Vision*, 2019.

[28] Zagoruyko, S. and Komodakis, N. Wide residual networks. *British Machine Vision Conference*, 2016.

[29] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 1997.