Pós-Graduação em Ciência da Computação

LUÃ LÁZARO J. SANTOS

Squeezed Very Deep Convolutional Neural Networks for Text Classification

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2020

LUÃ LÁZARO J. SANTOS

Squeezed Very Deep Convolutional Neural Networks for Text Classification

Dissertation presented to the Computer Science Graduation Program of the Federal University of Pernambuco, as a requirement to obtain the Master title in Computer Science.

Main research fields: Computational intelligence, Deep Learning, Constrained Systems and Text Classification.

Advisor: Prof. Dr. Cleber Zanchettin

Recife
2020

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

S237s Santos, Luã Lázaro Jesus dos
Squeezed very deep convolutional neural networks for text classification / Luã Lázaro Jesus dos Santos. – 2020.
59 f.: il., fig., tab.

Orientador: Cleber Zanchettin.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2020.
Inclui referências.

1. Inteligência computacional. 2. Redes neurais convolucionais. I. Zanchettin, Cleber (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2020 - 148

Luã Lázaro Jesus dos Santos

“Squeezed Very Deep Convolutional Neural Networks for Text Classification”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 14/02/2020.

BANCA EXAMINADORA

_____

Prof. Dr. Adriano Lorena Inácio Oliveira
Centro de Informática/UFPE

_____

Prof. Dr. Byron Leite Dantas Bezerra
Escola Politécnica de Pernambuco

_____

Prof. Dr. Cleber Zanchettin
Centro de Informática/UFPE
(**Orientador**)

I dedicate this dissertation to all my family, friends, and professors who gave me all the necessary support to get here.

ACKNOWLEDGEMENTS

My academic life was a long journey with lots of good and also bad moments, but in all these moments, I was never only by myself. Several people stayed on this road with me.

I want to acknowledge the professors Ricardo Prudêncio and Flávia Barros, who supported me in the first part of my master's. You both help me not only with my academic field but also with my personal happens and health. In such a tricky moment, you both behaved as if you were my parents. I will always be grateful for all the kindness offered to me. I have to say that you both are one of the most amazing people I knew on this road. Thank you for making me strong enough for not giving my dreams up.

The second part of my master was conducted for a person who became a strong example to me. During my Deeplearning classes, I had the opportunity to know better the professor Cleber Zanchettin. I have to say that you not only inspired me to give my best, as you provided the best support I could have. I want to thank you for all that advice and attention, and also for sharing with me your vast experience. This relationship made me grow substantially in both my academic and moral side and allowed me to perform this fantastic research work.

I want to acknowledge my family, but especially my mother Iracema, my aunt Jane and Lili, and my uncle Josildo. You all do not have an idea of how important you were during all this time. Since I was a child, my family experiences were never as I dreamed. I have never had a family like in the movies nor a peaceful childhood. I grew up, got mature, and could make my own decisions. I could rebuild myself. You all were my safety island during the period that could have been the most lonely of my life. But it was not. Uncle Josildo and aunt Lili care of me as a son, providing me not only material stuff, like food and clothes, but also those things that do not have a price as a warm hug, sincere advice, and a "god blessed you, son". My aunt Jane was my true confidant. Thank you, Aunt Jane, for all the deep conversations we had, and for all the experiences you shared with me, they guided and helped me to stay strong even during the worst storms.

Mun, you are the strongest woman I have ever know. You are all I could have dreamed of a mother. We indeed took a long time far from each other. We missed each other so much. But I am so gratified to have you again with me. I am incredibly proud of being your son, of having you as my mother. God works in mysterious ways mun, thank you for all that your love has been teaching me. I feel you with me in every step I make, I always felt. I love you, mun.

I also want to acknowledge my lead Thi and my manager, Juan. Moving from Recife to São Paulo for working, was a big challenge. I grew up in Recife, and my master's degree program was still ongoing. Thank you for all the support given to me. You both lighten

this pathway.

I want to talk about my heart-friends a little. I knew them in different ways. But all of them profoundly impacted my history. Andrea, It was a fantastic surprise getting close to you. How many things we had in common, and how many times we thought: "-Oh my gosh, we complete each other so well!". We spent a lot of time together during this master's degree program. Sometimes it was very hard, but lock at how many times we laughed! We published a fantastic article and that trip was unforgettable. Thank you for all the knowledge you share with me and for all the partnership.

Jil, we knew each other since the beginning of our graduation, and our personal histories were very similar. We indeed oscillate between moments were we stay very close and very far. But in all this road we could cultivate good feelings. We protected and supported each other as much as we could. Today we are almost married (okay, not that much haha), but we are now living together that good dreams we always wanted to. I am so happy about what we are building, thank you for always have been taking care of me.

Meua, I could write an entire chapter telling you how important you are in my trajectory and how I love you. I'm not able to say for how long we are together because I do not think that we met each other in this life. Our mutual kindness, care, and love transcend every imaginable barrier. The most significant part of me cames from our reflection. I am not able to say how many times you have inspired me and how many times you give me the power I needed to stand up. I am just thankful for every second we spent together and proud of the giant woman you are. Also, thank you for giving me one more mother. Mrs. Rose, you have a special room in my heart. I just wanted to finish this part by saying: Let the sessions began! We will always be together, we have always been. Love you.

Finally, I would like to acknowledge you, Marison, Ivison, and Bárbara. Thank you for every hug, every funny moment, for every positive vibration. I really take you all with me inside my heart and in my best memories. Every one of you taught me how to be a better person, how to feel light. Thank you all for being part of my life.

ABSTRACT

Embedding artificial intelligence on constrained platforms has become a trend since the growth of embedded systems and mobile devices, experimented in recent years. Although constrained platforms do not have enough processing capabilities to train a sophisticated deep learning model, like Convolutional Neural Network (CNN), they are already capable of performing inference locally by using a previously trained embedded model. This approach enables numerous advantages such as more privacy, smaller response latency, and no real-time network dependence. Still, the use of a local CNN model on constrained platforms is restricted by its storage size and processing power. Most of the research in CNN has focused on increasing network depth to improve accuracy. In the text classification area, deep models were proposed with excellent performance but relying on large architectures with thousands of parameters, and consequently, they require high storage size and processing. One of the models with much renown is the Very Deep Convolutional Neural Networks (VDCNN). In this dissertation, it is proposed an architectural modification in the VDCNN model to reduce its storage size while keeping its performance. In this optimization process, the impacts of using Temporal Depthwise Separable Convolutions and Global Average Pooling in the network are evaluated regarding parameters, storage size, dedicated hardware dependence, and accuracy. The proposed Squeezed Very Deep Convolutional Neural Networks (SVDCNN) model is between 10x and 20x smaller than the original version, depending on the network depth, maintaining a maximum disk size of 6MB. Regarding accuracy, the network experiences a loss between 0.1% and 1.0% in the accuracy performance while obtains lower latency over non-dedicated hardware and higher inference time ratio compared to the baseline model.

Keywords: Convolutional Neural Networks. Text Classification. Embedded Platform. Constrained models.

RESUMO

Embarcar inteligência artificial em plataformas com restrições de desempenho tem se tornado uma tendência desde o crescimento no uso de sistemas embarcados e dispositivos móveis, presenciado nos últimos anos. Apesar de sistemas com restrições de desempenho não terem capacidade de processamento suficiente para treinar modelos complexos, como as Redes Neurais Convolucionais (RNC), eles já são capazes de realizar sua inferência utilizando um modelo embarcado previamente treinado. Essa abordagem oferece diversas vantagens, tais como maior privacidade, menor latência de resposta e a não dependência de conexão com a internet em tempo real. De todo modo, o uso de um modelo de RNC em dispositivos com restrições de desempenho é condicionado ao seu tamanho de armazenamento e poder de processamento. Muitas das pesquisas em RNC tem focado em aumentar a profundidade da rede para melhorar sua acurácia. No campo de classificação de texto, modelos profundos apresentam excelente performance, mas se baseiam em arquiteturas grandes, com milhares de parêmetros, e consequentemente, alto requisito de armazenamento e processamento. Um dos modelos com bastante destaque é o *Very Deep Convolutional Neural Networks* (VDCNN). Nesta dissertação, é proposta a modificação da estrutura do modelo VDCNN para reduzir seu tamanho de armazenamento mantendo sua performance. Neste processo de otimização, são avaliados os impactos do uso de *Depthwise Separable Convolutions* e *Global Average Pooling* na arquitetura da rede, considerando a quantidade de parâmetros, tamanho de armazenamento, dependência de hardware dedicado e acurácia. O modelo proposto, *Squeezed Very Deep Convolutional Neural Networks* (SVDCNN), é entre 10 e 20 vezes menor do que sua versão original, dependendo da profundidade da rede utilizada, mantendo um tamanho de armazenamento máximo de 6MB. Com relação à acurácia, o modelo experimenta uma perda entre 0.1% e 1.0% na perfomance de classificação enquanto obtém menor latência em hardware não-dedicado e maior quociente de tempo de inferência comparado com o modelo base.

Palavras-chaves: Redes Neurais Convolucionais. Classificação de Texto. Plataformas Embarcadas. Modelos restritos em performance.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DSC	Depthwise Separable Convolution
FC	Fully Connected
FPGA	Field Programmable Gate Array
GAP	Global Average Pooling
GPU	Graphics Processing Unit
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
NLP	Natural Language Processing
OS	Operational Systems
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SSC	Spatial Separable Convolution
SVDCNN	Squeezed Very Deep Convolutional Neural Networks
TDSC	Temporal Depthwise Separable Convolution
TSC	Temporal Standard Convolution
VDCNN	Very Deep Convolutional Neural Networks

CONTENTS

1 INTRODUCTION

The general trend in the success of deep learning approaches has been developing models with increasing layers and parameters. Deeper neural networks have achieved high-quality results in different tasks, such as image classification, detection, and segmentation. Deep models can also learn hierarchical feature representations from the input (ZEILER; FERGUS, 2014b). On the other hand, in the Natural Language Processing (NLP) field, the belief that compositional models can also be successfully applied to text-related tasks is more recent.

The increasing availability of text data motivates the research for models able to improve accuracy, over massive datasets, in different language tasks. Following the image classification Convolutional Neural Network (CNN) tendency, the research in text classification has placed effort into developing deeper networks. The first CNN based approach for text was a shallow network with one layer (KIM, 2014). Following this work, deeper architectures were proposed (CONNEAU et al., 2017; ZHANG; ZHAO; LECUN, 2015). Conneau et al. (CONNEAU et al., 2017) was the first to propose Very Deep Convolutional Neural Networks (VDCNN) applied to text classification. According to the authors, VDCNN accuracy increases with depth. The approach with 29 layers presents excellent performance for text classification (CONNEAU et al., 2017).

However, regardless of making networks deeper to improve accuracy, little effort has been made to fit text classification models under constrained platforms, like embedded systems and mobile devices. Although constrained platforms do not have enough processing capabilities to train a sophisticated deep learning model, they are capable of processing predictions locally. In this approach, the model is trained on a proper environment for deep learning tasks, and then the trained model is transferred to the target constrained platform so that it performs the inference step.

Still, the model storage size is a limiting factor for constrained platforms (IANDOLA et al., 2016; HOWARD et al., 2017; GONG et al., 2014). Concerning embedded systems, Field Programmable Gate Array (FPGA) are a clear example: they usually offer less than 10MB of on-chip memory and no off-chip memory or storage. Regarding mobile devices, the model size impacts directly on the final application size. At the same time, there is a constant effort for developing smaller sized applications on both entry-level and cutting-edge devices. Relative to entry-level devices, there are Operational Systems (OS) with several restrictions for maximum application size. For example, on Android GO OS, applications should be less than 40MB on the device, while games should be less than 65MB on the device (GOOGLE, 2020).

Moreover, embedding smaller local models provides additional benefits, even to platforms that do not have storage constraints, such as cutting-edge smartphones and devices.

For example, the final size of an application impacts the amount of mobile data needed for its download and update. Additionally, there are other relevant benefits while embedding local models such as:

- No real-time network dependence: The user can use the application even with no internet connection;
- Lower latency speed: The inference is performed locally (not on a server) so it can be faster and more reliable by avoiding network requests;
- Scalability: No need to worry about scaling or distributed computing, once each user device performs the processing;
- Privacy: No personal data needs to leave the device since all processing is performed locally;

Therefore, the rising number of embedded systems and mobile devices, in addition to the increasing appeal for embedding artificial intelligence, motivates the research for smaller models.

1.1 GOALS

The main objective of this dissertation is to propose a very deep CNN model able to be locally embedded on constrained platforms while keeping accurate results. Thus, this work presents the Squeezed Very Deep Convolutional Neural Networks (SVDCNN), a very deep text classification model that requires significantly fewer parameters compared to the state-of-the-art CNN. Therefore, the main goals of this work are listed in details as follows:

- Reduce the number of parameters and consequently the storage size of the model;
- Keep the same accuracy performance or present minimal degradation;
- Examine the dedicated hardware dependency of the proposed model;
- Perform experiments to validate the proposed architectural modification;

To achieve the abovementioned goals, the use of Temporal Depthwise Separable Convolutions and Global Average Pooling techniques was investigated in the VDCNN network proposed by Conneaut et al. (CONNEAU et al., 2017).

1.2 DISSERTATION-RELATED PUBLICATIONS

- Duque A.B., Santos L.L.J., Macêdo D., Zanchettin C. (2019) Squeezed Very Deep Convolutional Neural Networks for Text Classification. In: Tetko I., Kůrková V., Karpov P., Theis F. (eds) Artificial Neural Networks and Machine Learning – ICANN 2019: Theoretical Neural Computation. ICANN 2019. Lecture Notes in Computer Science, vol 11727. Springer, Cham.

1.3 DOCUMENT ORGANIZATION

The remain sections are strutured as follow.

- Chapter 2 presents a historical review since the beginnings of Artificial Neural Networks until the modern Convolutional Neural Networks and their lightweight versions. Lastly, an overview of Deep Neural Networks for text classification tasks is performed, presenting the main related works, and the importance of the model used as base of this research.
- Chapter 3 presents the VDCNN, the model used as the base to the proposed SVDCNN model. Its architecture is deeply explained, as well as the author's experience with the application of shortcut connections.
- Chapter 4 presents an overview of Separable Convolutions. Its two types, Depthwise and Spatial, are introduced with discussions of applicability, benefits, and drawbacks.
- Chapter 5 presents the Global Average Pooling technique, reviewing its first apparitions and architecture besides its applicability, benefits, and drawbacks as well.
- Chapter 6 introduces the proposed SVDCNN, a text classification model that presents state-of-the-art results and holds significantly less trainable parameters so that it can be locally embedded in constrained platforms. Also, the two architectures modifications implemented are discussed in detail.
- Chapter 7 describes how the experiments were performed, detailing the parameters, base system, and datasets utilized.
- Chapter 8 presents the results obtained by the SVDCNN model compared to the VDCNN and Char-CNN models. These results are exhibited in terms of the number of parameters, storage size, shortcut connections performance, inference time, and accuracy.
- Chapter 9 finishes this research work presenting the conclusions obtained, as well as the contributions and directions for future works.

2 HISTORY REVIEW AND RELATED WORKS

2.1 ARTIFICIAL NEURAL NETWORKS

The first directions to artificial neural networks took place in the 1940s when the neurophysiologist Warren McCulloch and the young mathematician Walter Pitts modeled a simplistic neural network manipulating electrical circuits. In their published paper, they described how the brain neurons' behavior could be replicated. Later in 1949, Donald Hebb came up with one of the more crucial researches for the Artificial Neural Network (ANN) field advent, the Synaptic Plasticity theory. In its research, namely The Organization of Behavior, he showed that each time a set of brain cells fire together, they strengthen the synapses, so that the connection between them is improved (MCCULLOCH; PITTS, 1988). This theory profoundly influenced the formulation of the first train rules of the ANN models.

The 1950s started, and the IBM researcher Nathanial Rochester, driven by the increase of computational power, attempted to simulate the first neural network, but, unfortunately, he did not succeed. Still, the efforts for boosting both artificial intelligence and neural network researches did not cease. In 1956, the Dartmouth Summer Research Project in Artificial Intelligence fomented these areas, and in 1957, John von Neumann proposed to simulate simple neuron functions via telegraph relays or vacuum tubes.

The turn of the '50s was brilliant to the ANN field, beginning with the neuro-biologist of Cornell, Frank Rosenblatt (ROSENBLATT, 1958). In order to understand the processing associated with the operation of a fly's eye, Rosenblatt started to work on the Perceptron. Its implementation was hardware-based and enabled the classification of continuous values, given as input, between two classes. To perform the classification, the Perceptron calculates the weighted-sum of the inputs and then subtracts a threshold. Figure 1 shows the proposed Perceptron architecture. The result points to one of the two possible classes. This scheme became known as Rosenblatt update weight rule, and it is currently the oldest neural network architecture in usage.

After, in 1959, the models ADALINE and MADALINE were published by Bernard Widrow and Marcian Hoff. The models' names are acronyms inspired by the usage of Multiple ADAptive LINear Elements. ADALINE model is capable of recognizing binary patterns so that it can predict the next bit of a sequence of bits. It can be useful, for example, when decoding streaming bits from a phone line. MADALINE, on the other hand, can eliminate phone lines' echoes by using an adaptive filter. It was the first neural network employed in a real-world problem (WIDROW; HOFF, 1960).

Almost three decades passed, and although the Widrow-Hoff rule has enabled to solve several problems, a way to train neural networks with more than two layers was needed

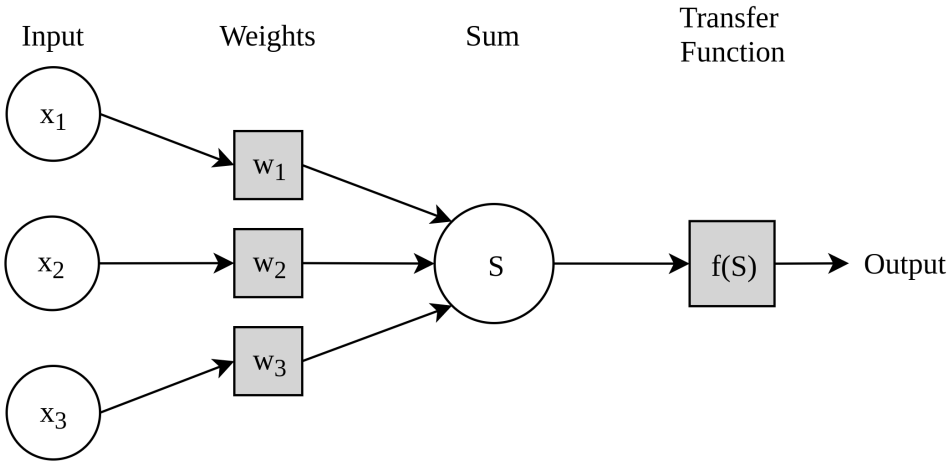

Figure 1 – Perceptron architecture proposed by Rosenblatt.

to step forward. In 1986, three different autonomous research groups came up with similar methods to distribute pattern recognition errors inside the network. These methods were later unified as what is known as backpropagation. The backpropagation rule required thousands of iterations to learn, being in practice slower than the Widrow-Hoff rule but generated more accurate results. One of the research groups mentioned above included David Rumelhart, a previous Stanford's psychology department member who published a functional version of the Multilayer Perceptron (MLP) (RUMELHART; HINTON; WILLIAMS, 1986). Also, in this study, the authors proved mathematically that an MLP could implement any continuous function as long as there were enough neurons in the network.

With the advent of MLP networks, deeper neural network architectures started to be investigated. In 1989 the first Convolutional Neural Network (CNN) was prosed by Yann LeCun and propelled the Deep Learning field (LECUN et al., 1989). After nine long years of research, LeCun came with LeNet5, a network architecture that could effectively extract similar features from multiple locations of an image through the usage of convolutions with few learnable parameters (LECUN et al., 1998). Also, the research directed by LeCun stated that the features are distributed over the whole image. The LeNet5 architecture was composed of a sequence of three layers: convolution, pooling, and non-linearity. Moreover, this architecture introduced map subsample by spatial average and sparse connection matrix among the layers to avoid the high computational cost. The LeNet5 was a turning point for the deep learning field, and truly inspired many of the modern deep learning architectures for image-related tasks.

Although LetNet5 opened the directions for the Deep Learning field, the computational power was a substantial limitation to train deeper models. At that time, Graphics Processing Unit (GPU) has not been managed to work as a general-purpose computing

tool, and even the Central Processing Unit (CPU) did not have relevant processing capabilities. Hereupon, the years between 1998 and 2010 were tepid, with the researchers not presenting much progress. Nevertheless, with the popularization of smartphones and digital cameras, an increasing amount of data was being generated, and in 2010 the first deep network, implemented in GPU, was published by Dan Claudiu Ciresan and Jurgen Schmidhuber (CIRESAN et al., 2010). That network had forward and backward operations performed by an NVIDIA GTX 280 GPU and also included up to 9 layers.

With the rise of the computational power pulled by the usage of the GPU as a general-purpose computing tool, and with the increasing availability of data, deep learning models became a prolific direction for image classification tasks. In 2012, a more in-depth and broader version of the Lenet5 network was published by Alex Krizhevsky (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Known as AlexNet, this network won the challenging ImageNet competition by a considerable margin, showing the extraordinary potential of deep learning models. After the enormous AlexNet success, the scientific community turned their eyes to deep learning models, and day by day, increasingly powerfull models for image-related tasks were proposed. Some of them are VGG, NiN and GoogleLeNet. (SIMONYAN; ZISSERMAN, 2015; LIN; CHEN; YAN, 2013; SZEGEDY et al., 2015).

Indeed, the increase in computational power allowed the creation of even more profound models. Usually, the addition of extra hidden layers boosts the network capability of learning more sophisticated functions, and consequently, its prediction capacities. Nevertheless, the more profound a model is, the more difficult it is to back-propagate the error concerning the weights. The gradients tend to be increasingly smaller as they move backward in the network so that the first layers train very fast while the last ones train very slow or they are not even able to train. As a result, it was not possible to get extra performance by adding extra layers: the training process becomes longer, and the accuracy performance is degraded. This barrier is named the Vanish Gradient problem, and it was the major problem to train Deep Neural Network models about 10 years ago.

An significant step to overcome the Vanish Gradient problem was given in 2011. Glorot et al. demonstrated that the ReLU activation function enabled better training for deeper models (GLOROT; BORDES; BENGIO, 2011). Instead of offering increasingly smaller gradients as the Sigmoid and Tanh activation functions, which have a maximum slope of 0.25, the ReLu offers a stable gradient output due to its gradient slope equal to 1. Another solution to the Vanish Gradient Problem was also proposed by Kaiming He et al. in the Deep Residual Learning for Image Recognition paper. This paper presented the skip connection, a pathway that connected the output of one layer with the input of a previous layer. This approach make the gradient easier back-propagation thorough the layers and has became a profitable direction to improve the accuracy of very deep CNN models (HE et al., 2016a).

2.2 DEEP NEURAL NETWORKS FOR NLP

Deep neural networks have been successfully achieving meaningful results on different text-related tasks as transduction, synthesis, and classification. The Transformer model, proposed by (VASWANI et al., 2017), was the first transduction model that excluded the use of RNNs sequentially aligned, convolutions, and encoding levels, thus the Transformer model computes its representations between input and output entirely through self-attention. In the text synthesis field, Reed et al. proposed a Generative adversarial network (GAN) architecture which enables compelling text to image synthesis of bird and flower images from human-written descriptions (REED et al., 2016). The GAN architecture was first proposed by Goodfellow et al., and it works by learning the data distribution in order to generate artificial instances (GOODFELLOW et al., 2014). Further, GAN models can also be applied to several other tasks as frame prediction, image generation, and voice syntheses (LOTTER; KREIMAN; COX, 2015; ISOLA et al., 2016; GAO; SINGH; RAJ, 2018).

Concerning text classification, the default approach has been the usage of Recurrent Neural Network (RNN). This approach is a particular case of Recursive Neural Networks, where the contexts are combined using the weights sequentially, from left to right (SOCHER et al., 2011). Thereby, it is possible to capture the word's relations naturally, considering the text direction. The most common approach for RNN is the Long Short-Term Memory (LSTM) model (HOCHREITER; SCHMIDHUBER, 1997; SUNDERMEYER; NEY; SCHLUTER, 2015; TAI; SOCHER; MANNING, 2015), which is a variety of RNN that preserves long term dependency more effectively in comparison to its base model.

Although RNN models have been a safe direction to text classification tasks, the successful results obtained by Convolutional Neural Networks on the image classification field raised the curiosity about its applicability in the text classification field. Differently from earlier classification approaches, where feature extraction and classification were processed as two entirely different tasks, CNN deal to perform these two steps at once, as one single unified task (LECUN et al., 1998). Another positive direction has been the sequential applying of pooling layers, which leads to improving the hierarchical representation extraction (ZEILER; FERGUS, 2014a).

The usage of CNN instead of RNN started to be investigated. Its usage in text classification tasks has some peculiarities. Contrasting with the traditional bi-dimensional representation of images, texts are one-dimensionally represented. Due to this property, temporal convolutions are employed. Furthermore, in order to train the networks, it is necessary to generate a numerical representation from the text so that the characters' discrete space can be mapped to a numerical continuous vector. This representation, namely embedding, is low dimensional, and it is usually obtained through the application of a lookup table generated from a given dictionary (CONNEAU et al., 2017).

A new approach for text classification tasks consisted of a shallow neural network working on the word level and using only one convolutional layer followed by a one-

dimensional max polling layer (KIM, 2014). The author reported results on six small datasets, highly focused on the Stanford Sentiment Treebank. Later, Zhang et al. (ZHANG; ZHAO; LECUN, 2015) proposed the first CNN to a character level (Char-CNN), which allowed them to train up to 6 convolutional layers, followed by three fully connected classification layers. Char-CNN uses convolutional kernels of size 3 and 7, as well as simple max-pooling layers. Also, the author introduced the usage of large-scale datasets.

Char-CNN model was able to achieve state-of-the-art results when it was proposed, but there was still a gap between the CNN models for text and image classification. While the networks in computer vision used up to 150 layers (HE et al., 2016a; HE et al., 2016b), there was a belief that it was not possible to improve the accuracy by increasing the network depth for text classification tasks. However, Conneau proposed the Very Deep Convolutional Neural Networks (VDCNN), also on a character level (CONNEAU et al., 2017), presenting improvements compared to Char-CNN. Conneau has shown that deeper models can increase text classification accuracy, indeed. VDCNN has up to 49 layers and uses only small kernel convolutions and pooling operations. The proposed architecture relies on the VGG and ResNet philosophy (HE et al., 2016a; SIMONYAN; ZISSERMAN, 2015): The number of feature maps and the temporal resolution is modeled, so their product is constant. This approach makes it easier to control the memory footprint of the network. Just like Zhang, Conneau CNN utilized standard convolutional blocks and fully connected layers to combine convolution information(ZHANG; ZHAO; LECUN, 2015; CONNEAU et al., 2017). This architecture decision increases the number of parameters and the storage size of the models. However, size and speed were not the focus of this work.

The idea of developing smaller and more efficient CNN models with minimal accuracy degradation, is a less explored research direction in the NLP field. However, it has already been a trend for computer vision applications (HOWARD et al., 2017; IANDOLA et al., 2016; SANTOS et al., 2018). Most approaches consist of compressing pre-trained networks or training small networks (HOWARD et al., 2017).

A recent tendency in image deep learning models is replacing standard convolutional blocks with Depthwise Separable Convolution (DSC). DSC were initially introduced by Sifre while participating in the Google Brain internship (SIFRE; MALLAT, 2014). Its first application, though, was seen in the AlexNet model where the usage of DSC enabled a speed-up on convergence time, a significant reduction in the number of parameters, and a small accuracy improvement (LECUN et al., 1998). Since then, DSC have been successfully applied to image classification, and (HOWARD et al., 2017; SANTOS et al., 2018; CHOLLET, 2017) machine translation (KAISER; GOMEZ; CHOLLET, 2017) to reduce the computation in convolutional blocks. Another approach that aims to light-weight the models is the use of a Global Average Pooling (GAP) layer to perform the final classification task, instead of the traditional fully connected layers. This strategy has become a standard architectural decision for newer CNN (HE et al., 2016a; HUANG et al., 2017).

2.3 CONCLUSION AND ADDITIONAL COMMENTS

In recent years, the image classification field has presented surprising accuracy results using increasingly deeper CNN models. Still, there was unbelief that the text classification tasks would not be able to follow this tendency. In this context, the VDCNN model representes a significant mark to the text classification field. This model demonstrated that its possible to obtain increasingly accurate results when the number of layers increases in text classification tasks (CONNEAU et al., 2017) — also, the VDCNN model encourages the research in the application of other successful approaches used in the image classification field, to text-related tasks.

The SVDCNN model comes to step forward in the direction fomented by Conneau. The proposed model goes through another tendency successfully experienced only in the image classification field: the reduction of very deep CNN models with minimal accuracy degradation, aiming constrained platforms. To achieve this goal, the use of Depthwise Separable Convolutions and Global Average polling is performed as well as the usage of skip connection to improve the model's performance.

3 VDCNN MODEL FOR TEXT CLASSIFICATION

This chapter presents the Very Deep Convolutional Neural Networks (VDCNN), the model used as base for the proposed Squeezed Very Deep Convolutional Neural Networks (SVD-CNN) model.

3.1 THE MODEL

The VDCNN is a modular architecture for text classification tasks developed to offer different depth levels (9, 17, 29, and 49). Fig. 2 presents depth-9 architecture configuration. The network begins with a lookup table, which generates the embeddings for the input text and stores them in a 2D tensor of size (f_0, s). The number of input characters (s) is fixed to 1,024 while the embedding dimension (f_0) is 16. The embedding dimension can be seen as the number of RGB channels of an image.

The next layer (3, Temp Convolution, 64) applies 64 temporal convolutions of kernel size 3, so the output tensor has size $64 * s$. Its primary function is to fit the lookup table output with the modular network segment input composed by convolutional blocks.

The following rule is employed to minimize the network's memory footprint: Before each convolutional block doubling the number of feature maps, a pooling layer halves the temporal dimension. This strategy is inspired by the VGG and ResNets philosophy and results in three levels of feature maps: 128, 256, and 512 (see Fig. 2).

The tensor output size (T_{out}) of each convolutional block can be calculated according to the following equation:

$$T_{out} = N_{fm} * \frac{s}{2^p} \tag{3.1}$$

Where N_{fm} is the number of feature maps of the convolutional block, s is the input size, and p is the number of down-sampling operations performed. Since the VDCNN model uses padded input, the s value is constant (1,024). However, if variable-sized input is employed, the convolutional encoder can also provide adequate representation. The author emphasizes that representations of a text as a set of vectors of variable size can be valuable, namely for neural machine translation, in particular when combined with an attention model.

Lastly, for the classification task, the k most valuable features $(k = 8)$ are extracted using k-max pooling, generating a one-dimensional vector which supplies three fully connected layers with ReLU hidden units and softmax outputs. The number of hidden units is 2,048, and they do not use dropout but rather batch normalization after convolutional layers perform the network regularization. The VDCNN also contains shortcut connec-

tions (HE et al., 2016a) for each convolutional block implemented through the usage of 1×1 convolutions.

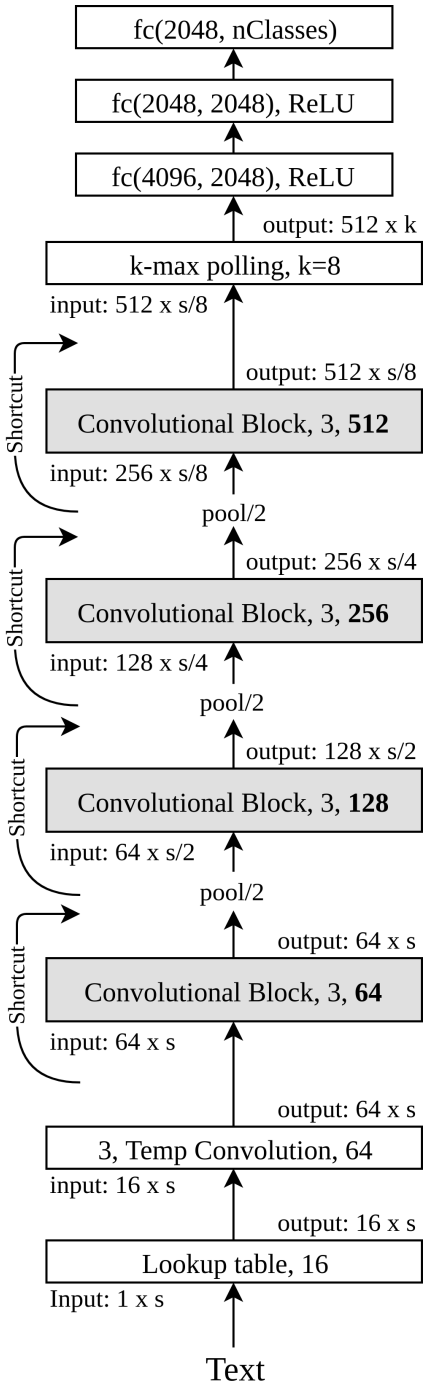

Figure 2 – VDCNN depth-9 architecture configuration.

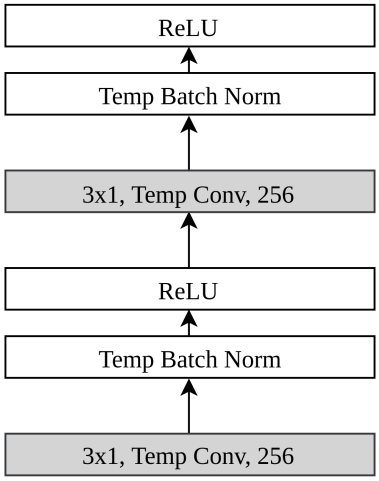

Figure 3 – VDCNN Convolutional Block 256.

3.2 CONVOLUTIONAL BLOCKS

Each Convolutional Block is a sequence of two temporal convolutional layers, each one accompanied by a temporal batch normalization layer (IOFFE; SZEGEDY, 2015) and a ReLU activation. Fig. 3 shows the VDCNN Convolutional Block.

The architecture applied to the convolutional blocks of the VDCNN model goes against the natural trend of Convolutional Networks for Natural Language Processing (NLP) related tasks. Typically, the process of model short-span and long-span relations is made easier when n-grams with different lengths are employed. Therefore, for these tasks, the architectures have been using up to 6 convolutional layers and combining convolutions of different sizes so that it could spawn tokens accordingly.

However, due to the high depth of VDCNN (up to 49 layers), only small convolutions of size 3 were used, so that the number of parameters of the network could be reasonable. Also, with each stacking of 4 layers of these small-sized convolutions, it is possible to obtain a span of 9 tokens, so that the network can build itself a deep hierarchical relation between the different "3-gram features" extracted. Furthermore, Conneau affirms that VDCNN architecture can be seen as a temporal adaptation of the VGG network (SIMONYAN; ZISSERMAN, 2015).

The different network depths are obtained by varying the number of convolutional blocks. As a convention, the depth of a network is given as its total number of convolutions. For instance, the depth-17 architecture configuration has two convolutional blocks of each level of feature maps, which results in 4 convolutional layers for each level (see Table 2). Considering the first convolutional layer of the network, we obtain the depth $2 * (2 + 2 + 2 + 2) + 1 = 17$. Table 2 summarizes the different depth architectures provided by VDCNN model.

Table 2 – Number of convolutional layers for each different VDCNN depth architecture

Depth	9	17	29	49
Convolutional Block 512	2	4	4	6
Convolutional Block 256	2	4	4	10
Convolutional Block 128	2	4	10	16
Convolutional Block 64	2	4	10	16
First Convolutional Layer	1	1	1	1

3.3 SHORTCUT CONNECTIONS

Deeper networks suffer from accuracy degradation after they achieve certain depth maxima, especially if it is a standard Convolutional Neural Network (CNN) (He et al., 2016a). This accuracy reduction is not related to overfitting or the training set but to the optimization process. The deeper the network, the harder it is to backpropagate the gradient through its layers. Also, the Stochastic Gradient Descent (SGD) starts not to be able to converge to a proper loss function minimum.

To study this accuracy degradation problem on text classification tasks, the VDCNN implements optional shortcut connections, seen from the first time on the ResNet model for image classification. These shortcut connections create a quick path that allows the input data x of a convolutional block to be aggregated with its result and passed forward as a residual map.

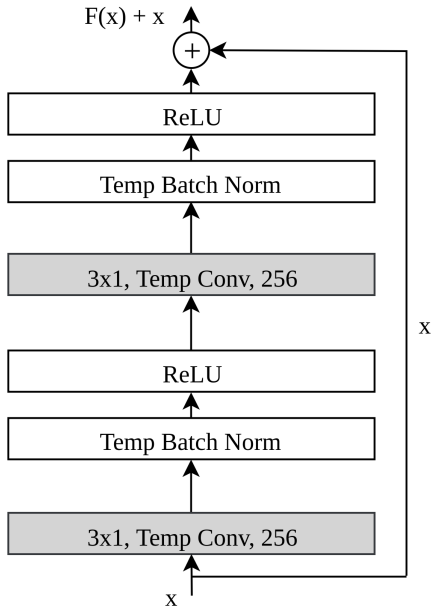

Figure 4 – Shortcut connection schema.

Figure 4 shows a shortcut connection schema. This technique allows the gradients to flow more easily in the network, and do not add almost any extra parameters or computational complexity (He et al., 2016a). Despite the good results presented by the ResNet model, the use of shortcut connections on VDCNN did not bring a consistent accuracy increasing. Concerning the depth-9, depth-17, and depth-29 architecture configurations, its use was not helpful. While to the depth-49, its use allowed better accuracy, but still not able to beat the results delivered by the depth-29 architecture configuration. Table 3 shows the accuracy results reported by the author on the Yelp Review Full data set with or without shortcut connections (CONNEAU et al., 2017).

Table 3 – Accuracy results on the Yelp Full data set for all depths, with or without shortcut connections.

Depth	Without Shortcut	With Shortcut
9	62,37	59,73
17	63,90	60,82
29	64,72	63,99
49	62,59	63,85

4 SEPARABLE CONVOLUTIONS

Separable convolutions have received increasing attention since its apparition in MobileNet (HOWARD et al., 2017). In contrast with the Standard Convolutions, the Separable ones have sensible less trainable parameters, which enables several benefits as being less prone to overfitting and reduce the demanded computation. This chapter starts describing how a Standard Convolution works and follows by explaining the two types of Separable convolutions available, named Depthwise Separable Convolution (DSC) and Spatial Separable Convolution (SSC). The first one had its temporal version applied to the proposed Squeezed Very Deep Convolutional Neural Networks (SVDCNN) model. Additionally, all this chapter uses the same base example, so that the different convolution approaches can be appropriately compared.

4.1 STANDARD CONVOLUTION

In the deep learning field, bi-dimensional images are usually represented by a composition of three channels, namely, height, width, and depth (the RGB component). Still, concerning text-related tasks, the input is one-dimensional, and its depth component length is defined by the embedding process applied. Let us assume that the embedment operation outputs a text representation as a $In_l \times Emb_d$ vector, where In_l is the length of the input and Emb_d is the embedding dimension or channels (as the RGB channels of an image). Hereupon, to make the comprehension easier, consider $In_l = 20$ and $Emb_d = 5$. Out of curiosity, the values used in the VDCNN/SVDCNN models are $In_l = 1024$ and $Emb_d = 16$.

Figure 5 shows the process of performing a convolution of kernel size 3 (3×1 convolution) with no padding and stride 1 in the text. Considering only the length dimension of the text, the convolution of the input text (20×1) with the kernel of size 3 (3×1) generates the output text (18×1).

The 3×1 kernel undergoes scalar multiplication with every three positions of the text tensor, generating one number each time. Thus, the generated text has an 18×1 shape since there is no padding ($20 - 3 + 1 = 18$).

However, because the text representation has five channels, the convolutional kernel also needs to have five channels. Therefore, instead of performing $3 * 1 = 3$ multiplications, we actually do $3 * 5 = 15$ multiplications every time the kernel moves. Also, the scalar matrix multiplication performed on every three positions of the text outputs only 1 number. Consequently, after being submitted to a 3×5 kernel, the 20×5 text will have its shape modified to 18×1. Figure 6 shows the process discussed above.

Finally, to achieve the target number of channels (or depth) n, it is necessary to

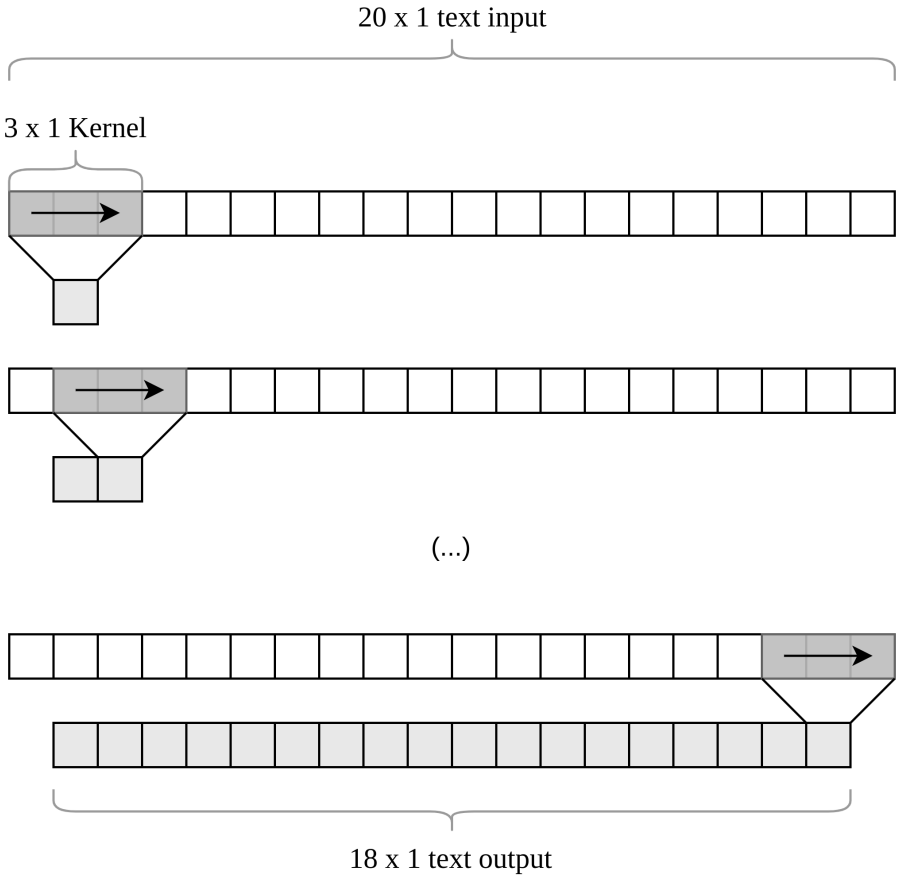

Figure 5 – 3×1 kernel being convoluted with a 20×1 sized text, with no padding and stride 1.

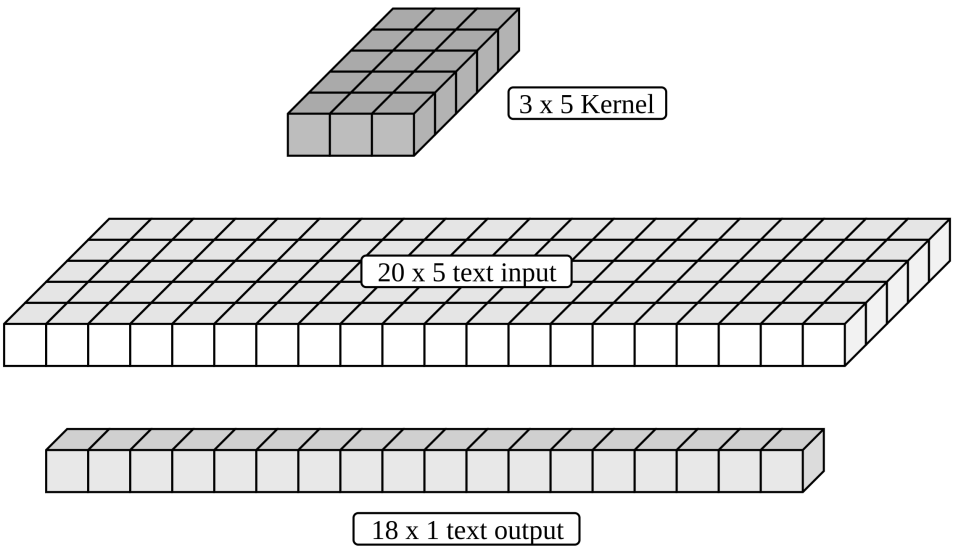

Figure 6 – The process of convolve a 20×5 text with a 3×5 kernel outputs a text with the 18×1 shape.

generate n kernels so that it is possible to produce n 18×1 text representations and aggregate them together. Figure 7 shows this process with the n parameter set to 8.

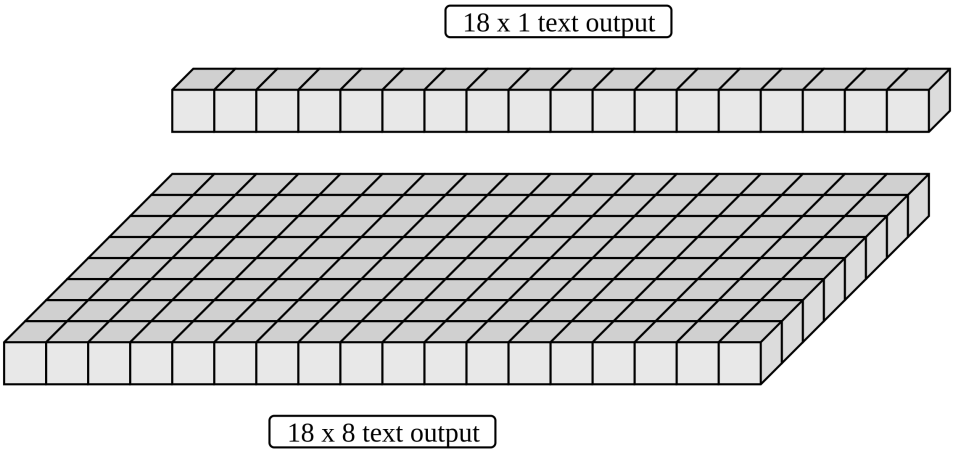

Figure 7 – Aggregation of eight 18×1 outputs, generating a output text with 18×8 shape.

Lastly, the number of multiplication operations performed by a one-dimensional standard convolution (M_{sc}) is defined as follows:

$$M_{sc} = K_s * Out_c * (In_l - K_l + 1) \tag{4.1}$$

Where, K_s is the kernel size, Out_c is the targe number of output channels, K_l is the kernel length, and In_l is the input length. Therefore, the total amount of multiplication performed by the example mentioned above is:

$$M = 3 * 5 * 8 * (20 - 3 + 1) = 2160 \tag{4.2}$$

4.2 DEPTHWISE SEPARABLE CONVOLUTION

A depthwise separable convolution separates the process performed by the standard convolution into two parts: a depthwise convolution and a pointwise convolution. This convolution holds this name because it deals with both spatial and depth dimensions. The spatial dimensions are, for example, the width and the height of an image or kernel, while the depth dimension is the number of channels. Also, the versatility of being able to substitute any standard convolution makes its usage a prevailing direction in the deep learning field.

4.2.1 Depthwise Convolution

The depthwise step convolves the text without altering its depth by the application of 5 kernels of shape 3x1. Figure 8, shows the depthwise convolution operating. Each kernel iterates one channel/depth of the text, performing the scalar products of every three positions of the text vector so that 18x1 texts are generated. Finally, by aggregating all generated text together, the 18x5 text is obtained.

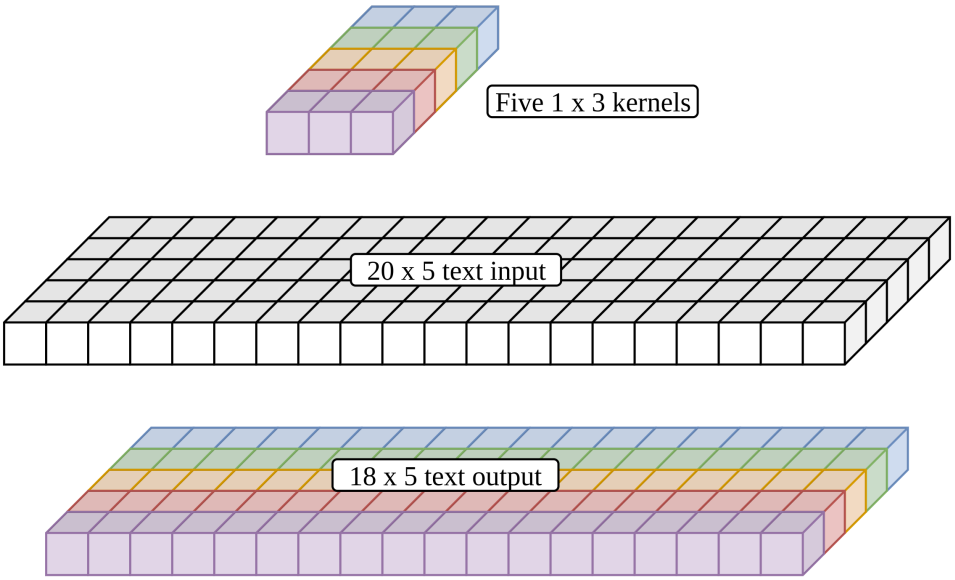

Figure 8 – Depthwise convolution, using 5 kernels to transform a 20×5 text into a 18×5 text.

4.2.2 Pointwise Convolution

As seen in section 4.1, the standard convolution transformed a 20×5 text into an 18×8 text. The depthwise convolution has already transformed the 20×5 text into an 18×5 text, as seen in Figure 8.

In order to transform the 18×5 text into an 18×8 text, the pointwise convolution applies 1x1 kernels. These kernels iterate over every single position of the text vector and have the depth equals to the text depth. Therefore, the convolution of a 1×5 kernel through the 18×5 text generates an 18×1 text. Figure 9, shows the pointwise convolution transforming a text of 5 channels into a text of 1 channel.

Lastly, the process performed to obtain the desired number of channels is pretty similar to the one done by the standard convolution: the aggregation of 18×1 texts, in this case, generated by the application of multiple 5×1 kernels. Visit the Figure 7 once more to review this process.

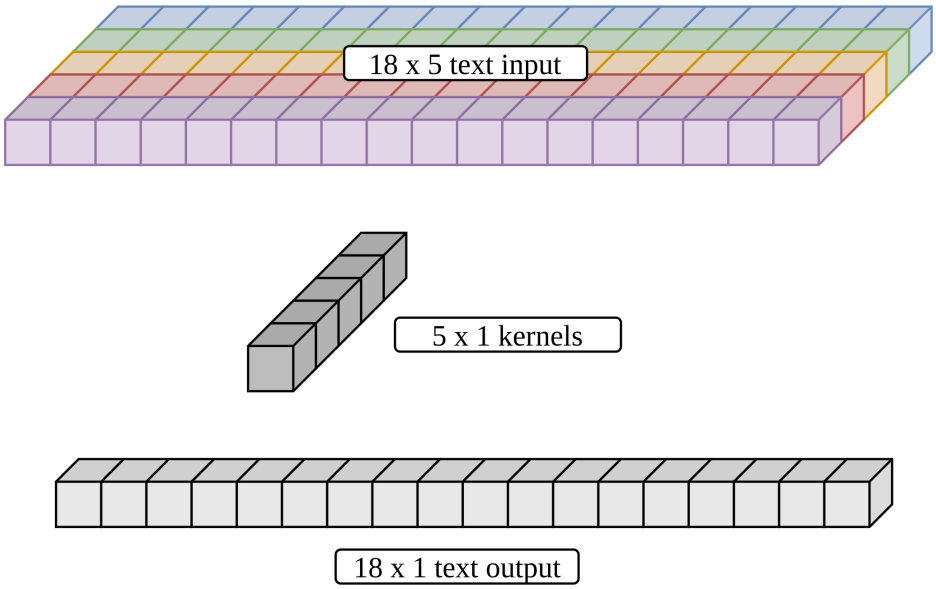

Figure 9 – Pointwise convolution, transforming an text of 5 channels to an text of 1 channel

The number of multiplication operations performed by a 1D depthwise separable convolution (M_{dsc}) is defined as follows:

$$M_{dsc} = In_c * KD_s * (In_l - KD_l + 1) + Out_c * KP_s \tag{4.3}$$

Where, In_c is the number of the input channels, KD_s is the size of the kernel used in the depthwise convolution, In_l is the length of the input, KD_l is the length of the kernel used in the depthwise convolution, Out_c is the target number of output channels, and KP_s is the size of the kernel used in the pointwise convolution. Therefore, the total amount of multiplication performed by the example mentioned above is:

$$M_{dsc} = 5 * 3 \times 1 * (20 - 3 + 1) + 8 * 1 \times 5 = 310 \tag{4.4}$$

4.3 SPATIAL SEPARABLE CONVOLUTION

The spatial separable convolution, as the name suggests, works over the spatial dimensions. Its operation consists of factorizing the standard convolution into two other convolutions with smaller kernels. To keep the equivalence, the multiplication between the two smaller kernels need to result in the original one. For illustration, consider the 4×4 kernel showed in Figure 10.

This approach enables a reduction in the computational complexity since fewer multiplications operations are performed. A standard convolution performs 16 multiplication

4	3	2	1
8	6	4	2
12	9	6	3
16	12	8	4

=

1
2
3
4

x

4	3	2	1

Figure 10 – Separating a 4x4 kernel spatially

operations considering the 4×4 kernel while its analogous spatial separable version performs 8, four of the 4×1 kernel plus four of the 1×4 kernel. Figure 11 shows this process in more details.

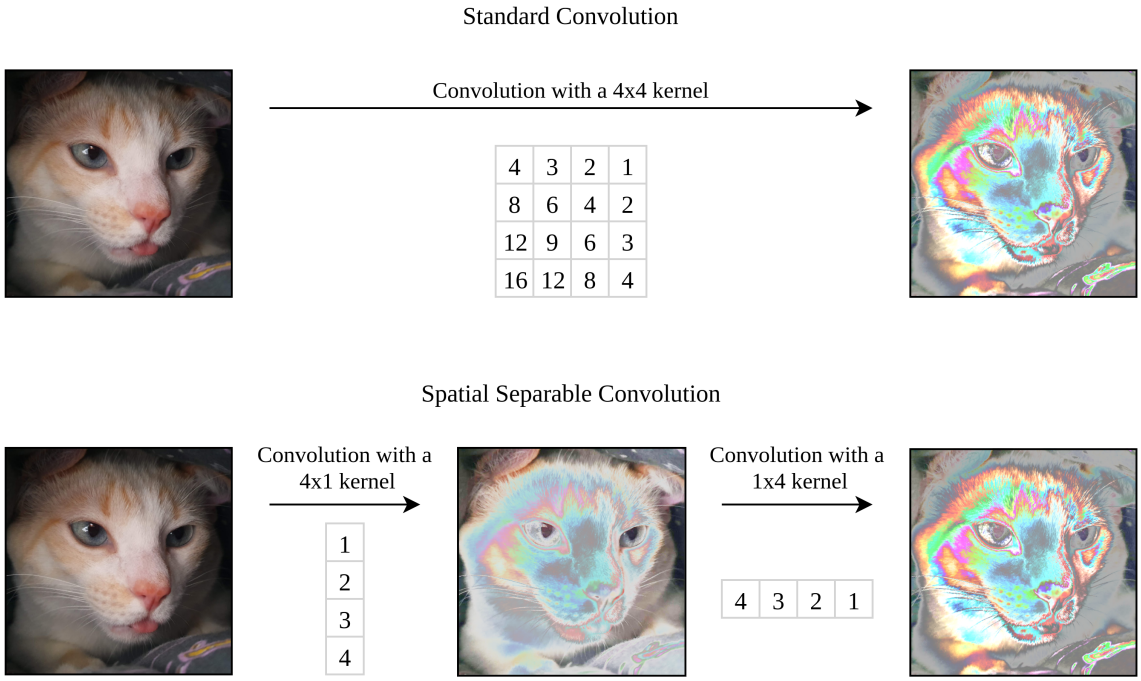

Figure 11 – Standard and Spatial Separable convolutions.

Although its simple formula, the spatial separable convolution has significant limitations. First, it does not have an equivalent version for each possible standard convolution, since the kernel of the standard convolution needs to have a size that allows its factorization. Second, since it works over spatial dimensions, it is not applicable for one-dimensional problems, for example, text classification. Due to these restrictions, its usage is not a prevailing direction in the deep learning field.

By the comparison of equations 4.2 and 4.4 its if possible to notice that the depthwise separable convolution performs significantly fewer multiplication operations than the standard convolution, saving about 85% of computational power. Moreover, the reduction of parameters accomplished by the depthwise separable convolution allows reducing the

proneness to overfitting and storage size of the network, without reducing its accuracy significantly (HOWARD et al., 2017; CHOLLET, 2017). Hereupon, there are no critical drawbacks in the replacement of the standard convolutions by its depthwise separable versions, unless the model has already fewer parameters. In this case, an additional reduction of parameters might cause a failure in the network learning during the training process.

Concerning the spatial separable convolutions, its usage is limited. Since its usage depends on a matrix decomposition, not always the standard convolutions are eligible. Also, since it works in the spatial dimension, they are not adequate to one-dimensional tasks as the text related ones.

5 GLOBAL AVERAGE POOLING

5.1 THE METHOD

The usage of densely connected layers as the final classification layers has been the traditional architecture choice for convolutional neural networks, presenting meaningful results on both images (SIMONYAN; ZISSERMAN, 2015) and text (CONNEAU et al., 2017) classification tasks. This architecture choice is associated with the usage of multiples series of convolutional layers interspersed with polling operations. The convolutional layers extract different components of the input data and represent them through feature maps, which fed up the final densely connected layers. These final layers are frequently fully connected layers and work as a standard neural network classifier that "interprets" feature map outputs and performs the category predictions.

Notwithstanding, the employment of Fully Connected (FC) layers drastically increases the number of trainable parameters of the network. For example, on the image classification model VGG16, the FC layers represent 89% of the total amount of the network parameters, while in the text classification model VDCNN, the FC layers represent 77% of the total amount of the network, in average, depending on the network depth configuration. The presence of a high number of parameters leads to two undesirable characteristics: 1. The network becomes more prone to overfitting, which prejudices its generalization ability. ; 2. The network becomes heavier, requiring more computational processing power and increasing its size when stored.

To prevent the overfitting urged by the usage of FC layers, Hinton et al. proposed the Dropout. This regularizer randomly sets half of the activations that feeds FC layers to zero during the training step. This approach leads to improve the generalization ability and largely prevents overfitting (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). However, Lin et al. came up with a technique that allows getting rid of FC layers and its deficiencies: the Global Average Pooling (LIN; CHEN; YAN, 2013).

The global average pooling technique consists of replacing the fully connected layers by Global Average Pooling (GAP) layers to reduce the total number of parameters in the model and consequentially making it less prone to overfitting. GAP layers work by reducing the spatial dimension of a tensor. Let h, w, and d be the height, width, and depth dimensions of a 3D tensor. The GAP transforms a $h \times w \times d$ tensor in a $1 \times 1 \times d$ tensor, by averaging all the $h \times w$ values. Figure 12 illustrates this process.

The first model, designed with GAP layers, namely NiN, was proposed by Lin et al. (LIN; CHEN; YAN, 2013). In the NiN model, each image category in the dataset is represented by one activation map given by the last max-pooling layer. Hereupon, a GAP layer receives these activation maps and averages them, producing a vector with a single

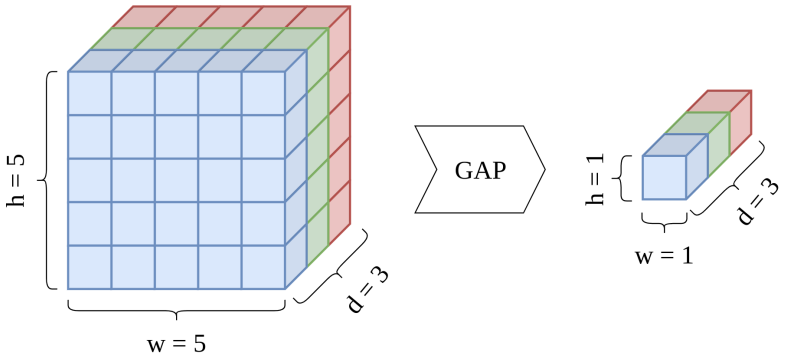

Figure 12 – Global Average Pooling transformation.

entry for each possible target class. Lastly, a softmax activation function outputs the predicted class probabilities.

Other authors, as Hoa et al. (DenseNet), Iandola et al. (SqueezeNet), and Santos et al. (SquezeNet-DSC), followed the same direction of eliminating all FC layers (LE; CERISARA; DENIS, 2017; IANDOLA et al., 2016; SANTOS et al., 2018). Kaiming et al., on the other hand, performed a more conservative modification using a global average pooling layer followed by one FC layer and softmax activation function to perform the class prediction (HE et al., 2016a).

Besides the two main GAP benefits, which are the reduction of the trainable parameter and reduction of over-fitting tendency, Lin et al. emphasize two more important benefits. By the removal of the FC layers, the feature maps become more closely related to the classification categories, acting as a class pertinence map. Also, the network becomes more robust to spatial translations in the data. If the discriminative features do exist in the feature map, no matter where they are, it will still be captured due to the averaging operation (LIN; CHEN; YAN, 2013).

6 SVDCNN MODEL FOR TEXT CLASSIFICATION

The proposed model is designed to be a light-weight version of the Very Deep Convolutional Neural Networks (VDCNN) model. Therefore, the objective is reducing the number of parameters so that the network can fit in constrained platforms as embedded systems and mobile devices while presenting minimal accuracy performance degradation. To achieve this purpose, we performed two architectural modifications in the base model. First, the replacement of the standard convolutions used in the convolutional blocks by Temporal Depthwise Separable Convolution (TDSC). Next, the application of the Global Average Pooling (GAP) technique to avoid the fully connected layers. The resulting proposed architecture is called Squeezed Very Deep Convolutional Neural Networks (SVDCNN).

6.1 TEMPORAL DEPTHWISE SEPARABLE CONVOLUTIONS ON SVDCNN MODEL

The use of Depthwise Separable Convolution (DSC) over standard convolutions allowed to reduce the number of parameters without relevant accuracy loss (HOWARD et al., 2017). As explained in section 4.2, DSC work decompounding the standard convolution into two parts: Depthwise and Pointwise. The first one is responsible for applying a convolutional filter to each channel of the input. When working with images, the network input channels are the RGB components, whereas, in a text input, we use the dimensions of the embedding for representation. The result is one feature map by channel for both cases above. The second convolution unifies the generated feature maps successively applying 1x1 convolutions to achieve the target amount of feature maps.

TDSC are DSC that work with one-dimensional convolutions. Although DSC holds verified results in image classification networks, the use of its temporal version for text-related tasks is less explored. Fig. 2a presents the architecture of a standard temporal convolution present in the VDCNN model (composed by the convolution, a Batch Normalization, and a ReLU layers), while Fig. 2b presents the equivalent TDSC architecture.

For a more formal definition, let P_{tsc} be the number of parameters of a Temporal Standard Convolution (TSC), where In and Out are the numbers of Input and Output channels respectively, and D_k is the kernel size:

$$P_{tsc} = In * Out * D_k \tag{6.1}$$

Alternatively, a TDSC achieves fewer parameters (P_{tdsc}):

$$P_{tdsc} = In * Dk + In * Out \tag{6.2}$$

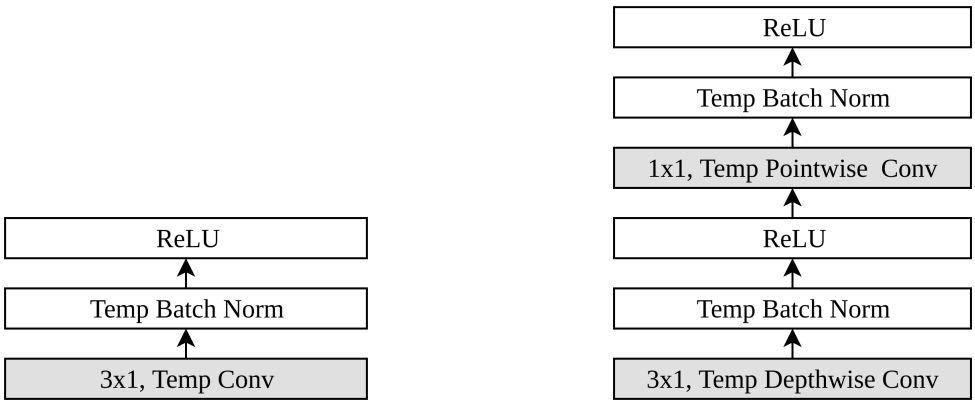

Figure 13 – a) Temporal Standard Convolution; b) Temporal Depthwise Separable Convolution.

In the VDCNN model, one convolutional block is composed of two temporal standard convolutional layers. The first one doubles the number of feature maps, while the second keeps the same value received as input. In our model, we proposed changing the temporal standard convolutions by TDSC .

Fig. 14 presents the standard convolutional block on the left and the proposed convolutional block using TDSC on the right. The pattern used in the figure for the convolutional layers is the following: "Kernel Size, Conv type, Output Feature Maps"; as a brief example consider "3x1, Temporal Conv, 256", which means a Temporal Convolution with kernel size 3 and 256 feature maps as output. From Equation 6.1, we have the number of parameters of the original convolutional block ($P_{convblock}$) as follows:

$$P_{convblock} = In * Out * 3 + Out * Out * 3 \tag{6.3}$$

Moreover, from equation 6.2, the number of parameters of the proposed convolutional block ($P_{convblock-tdsc}$) that uses TDSC being:

$$P_{convblock-tdsc} = In * 3 + In * Out + Out * 3 + Out * Out \tag{6.4}$$

For illustration, following the same characteristics of Fig. 14, consider that the number of input channels In is equal to 128, and the number of output channels Out is equal to 256. Our proposed approach accumulates a total of 99,456 parameters. In contrast, there are 294,912 parameters in the original convolutional block. The use of TDSC yields a reduction of 66.28% in the network size.

Lastly, since each standard temporal convolution turns into two (Depthwise and Pointwise), the number of convolutions per convolutional block has doubled. Nevertheless, these two convolutions work as one because it is not possible to use them separately, keeping the same proposal. In this way, we count them as one layer in the network depth. This de-

cision holds the provided depth architectures the same as the VDCNN model summarized in Table 2, contributing to a proper comparison between the models.

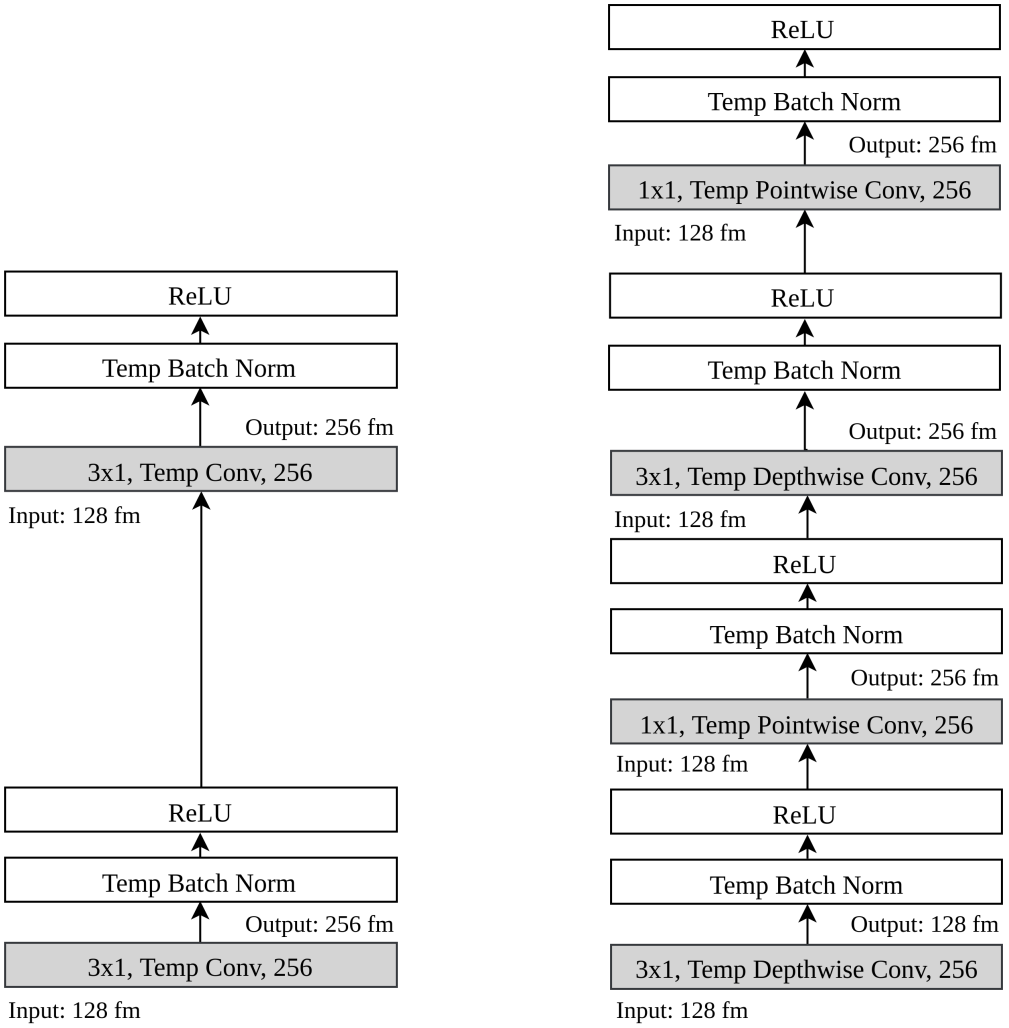

Figure 14 – a) Standard convolutional block of the VDCNN; b) Modified convolutional block of the SVDCNN.

6.2 GLOBAL AVERAGE POOLING TECHNIQUE ON SVDCNN MODEL

The VDCNN model uses a k-max-pooling layer ($k = 8$) followed by three Fully Connected (FC) layers to perform the classification task (Fig. 15). Although this approach is the traditional architecture choice for text classification Convolutional Neural Network (CNN), it introduces a significant number of parameters in the network. Also, considering the classification layers, the FC are the only layers that contribute to the number of parameters since the k-max-pooling layer does not have trainable parameters.

The number of parameters present in the classification layers of VDCNN model (Pc_{vdcnn})

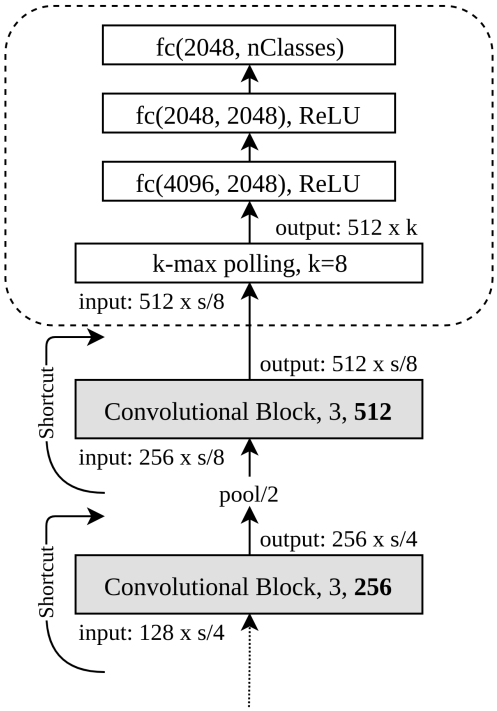

Figure 15 – VDCNN classification layers.

is calculated as follows, considering a problem with four target classes:

$$Pc_{vdcnn} = fc1 + fc2 + fc3 + pooling \tag{6.5}$$

While, the number of parameters of an FC layer holds is obtained as follows:

$$fc = fc_{in} * fc_{out} \tag{6.6}$$

Expanding equation 6.5 by using equation 6.6 it is obtained the equation 6.7:

$$Pc_{vdcnn} = (fc1_{in} * fc1_{out}) + (fc2_{in} * fc2_{out}) + (fc3_{in} * fc3_{out}) + 0 \tag{6.7}$$

Finally, feeding the proper values (Fig. 15) in equation 6.7, the number of parameters present in the classification layers of VDCNN model is obtained:

$$\begin{aligned} Pc_{vdcnn} &= (4,096 * 2,048) + (2,048 * 2,048) + (2,048 * 4) \\ Pc_{vdcnn} &= 12,591,104 \end{aligned} \tag{6.8}$$

As seen in chapter 5, a more recent architecture tendency relies on the reduction of fully connected layers by the usage of the GAP technique. The application of GAP guarantees some essential benefits such as the reduction of parameters without significantly degrade of the network accuracy (LIN; CHEN; YAN, 2013), and making the network less prone to overfitting.

Following the SqueezeNet philosophy, the idea is to modify the base model so that the last convolutional layer outputs one activation map for each target class. To accomplish that, we used a temporal convolutional layer that takes the place of the k-max-pooling layer. Next, the activation maps are fed into a GAP layer, which produces a tensor with a unique entry for each possible class. Lastly, a softmax activation provides the probability associated with each class, enabling the prediction.

Fig. 16 presents this proposed modification. Now, considering the classification layers, the only that has parameters is the temporal convolutional layer added since the GAP layer does not have any trainable parameter.

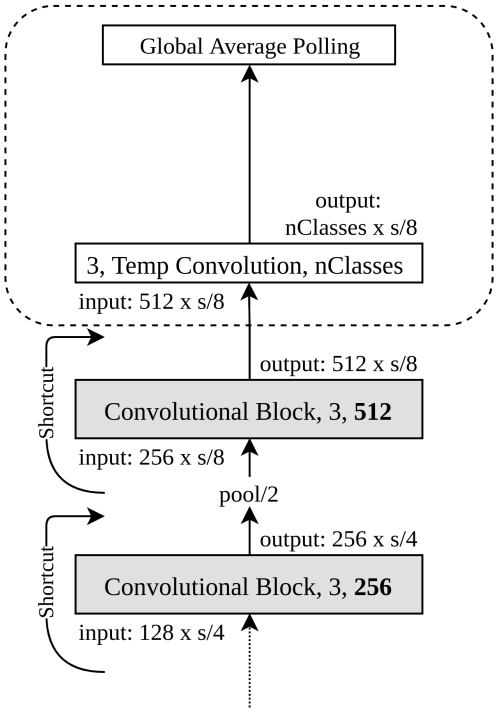

Figure 16 – SVDCNN classification layers.

Furthermore, the number of parameters present in the classification layers of SVDCNN model (Pc_{svdcnn}) is calculated as follows, also considering a problem with four target classes:

$$Pc_{svdcnn} = P_{temp_{conv}} + polling \tag{6.9}$$

While the number of parameters of a Temporal Convolution layer holds is obtained as follows:

$$P_{temp-conv} = K_w * In_{channels} * Out_{channels} \tag{6.10}$$

Expanding equation 6.9 by using equation 6.10 it is obtained the equation 6.11:

$$P_{temp-conv} = K_w * In_{channels} * Out_{channels} + 0 \tag{6.11}$$

Finally, feeding the proper values (Fig. 16) in equation 6.11, the number of parameters present in the classification layers of SVDCNN model is obtained:

$$\begin{aligned} P_{gap} &= 512 * 4 \\ P_{gap} &= 2,048 \end{aligned} \tag{6.12}$$

The proposed approach accumulates a total of 2,048 parameters in the classification layers. In contrast, there are 12,591,104 parameters in the original classification method. This architecture choice yields a reduction of 99.98% in the number of parameters.

7 EXPERIMENTAL METHODOLOGY

The experiment's goal is to investigate the impact of modifying the convolutional block of Very Deep Convolutional Neural Networks (VDCNN) model to Temporal Depthwise Separable Convolution (TDSC) and using the Global Average Pooling (GAP) technique over the original classification approach with three Fully Connected (FC) layers. We investigate if the techniques proposed to optimize Convolutional Neural Network (CNN) for text classification concerning constrained platforms and without degrading accuracy. Therefore, we compare our model with state-of-the-art CNN. We evaluate Char-CNN, VDCNN, and SVDCNN according to the number of parameters, storage size, inference time, and accuracy. The source code of the proposed model is available in the GitHub repository SVDCNN[1].

The original VDCNN paper reported the number of parameters of the convolutional layers, in which we reproduce in this dissertation. For SVDCNN and Char-CNN, we calculated the abovementioned number of parameters from the network architecture implemented in PyTorch. As for the FC layer's parameters, the number is obtained as the summation of the product of the input and output size of each FC layer for each CNN.

Considering the network parameters P and assuming that one float number on Cuda environment takes 4 bytes, we can calculate the network storage in megabytes, for all the models, as follows:

$$S = P * 4 \div 1,024^2 \tag{7.1}$$

Regarding the inference time, its average and standard deviation were calculated as the time to predict one instance of the AG's News dataset throughout 1,000 repetitions.

The SVDCNN experimental settings are similar to the original VDCNN paper, using the same dictionary and the same embedding size of 16 (CONNEAU et al., 2017). The training is also performed with Stochastic Gradient Descent (SGD), utilizing a size batch of 64, with a maximum of 100 epochs. We use an initial learning rate of 0.01, a momentum of 0.9, and a weight decay of 0.001. All the experiments were performed on an NVIDIA RTX 2080Ti GPU + Intel Core i7 4770s CPU.

7.1 DATASETS

When a model works with low-level raw features, for example, characters, it usually obtains better accuracy and generalization results when trained over largescale datasets. Still, most of the open-source datasets used for text classification related tasks are considerably short, and large-scale datasets are designed with a training set significantly more

[1] Link: <https://github.com/lazarotm/SVDCNN-v2>

concise than the testing set (LEWIS et al., 2004). To solve this problem, Zhang et al. generated a set of largescale datasets for text classification tasks, which was used for the first time in the Character-level Convolutional Networks for Text Classification paper.

The model's performance is evaluated on six of the large-scale public datasets introduced by Zhang et al. (ZHANG; ZHAO; LECUN, 2015): AG's News, Yelp Review Polarity, Yelp Review Full, Amazon Review Polarity and Amazon Review Full. Table 4 presents the statistics of the datasets mentioned above.

Table 4 – Datasets used in experiments

Dataset	#Train	#Test	#Classes	Classification Task
AG's News	120k	7.6k	4	News categorization
Yelp Polarity	560k	38k	2	Sentiment analysis
Yelp Full	650k	50k	5	Sentiment analysis
Amazon Review Full	3,000k	650k	5	Sentiment analysis
Amazon Review Polarity	3,600k	400k	2	Sentiment analysis

7.1.1 AG's News Dataset

The AG's news dataset uses the AG's news corpus. ComeToMyHead organization constructed this corpus and employed almost one million news articles from about two hundred news sources. The AG news dataset has 30000 training and 1900 test samples, randomly chosen, from each of the four largest classes present in the AG's news corpora. Also, only the title and description fields were utilized. Table 5 shows one example of each class of the AG News dataset.

Table 5 – AG's news' instance examples

Instance – "Tittle", "Description"	Class
"Famous scofflaws hit Japan", "US chess legend Bobby Fischer is the latest in a string of problem migrants for Japan."	World
"Defeat for GB canoeists", "British canoeists Nick Smith and Stuart Bowman are out of the men's C2 doubles."	Sports
"Hip Hop's Online Shop","Celebrity fashion is booming. These webpreneurs are bringing it to main street"	Business
"Catwoman far from perfect", "The Catwoman game is a major disappointment that feels like a pointless tie-in with the film."	Sci/Tech

7.1.2 Yelp Reviews Datasets

The Yelp review datasets arose from the Yelp Dataset Challenge in 2015. This dataset includes 1,569,264 text samples and their reviews. The Yelp Full dataset has 130,000 training and 10,000 testing samples for each class and was build using the number of stars given by the users as the prediction classes. Thus, since the number of stars varies between one and five, there are five prediction classes. Tables 6 shows one example of each class of the Yelp Full dataset.

Table 6 – Yelp Reviews Full's instance examples

Instance – "Description"	Class
"Great location! Close to shops and theatre. Nice staff."	5
"Good pizza and hoagies. It is always busy. Cash only."	4
"Dined in twice, food ok, atmosphere good."	3
"Slow service. Below average food. I'll pass"	2
"The only thing worse than the food is the service."	1

The Yelp Polarity dataset has 280,000 training and 19,000 test samples in each polarity. The samples with 1 and 2 stars have negative polarity, while the samples with 3, 4, and 5 have positive polarity. Tables 7 shows one example of each class of the Yelp Polarity dataset.

Table 7 – Yelp Reviews Polarity's instance examples

Instance – "Description"	Class
"Good pizza and hoagies. It is always busy. Cash only."	Positive
"Slow service. Below average food. I'll pass"	Negative

7.1.3 Amazon Reviews Datasets

The Amazon reviews datasets were built from the Stanford Network Analysis Project (SNAP). The SNAP fords 18 years with 34,686,770 reviews from 6,643,669 users on 2,441,053 products (MCAULEY; LESKOVEC, 2013). The build process was the same applies to Yelp reviews datasets: one for precise score prediction and the polarity version for soft prediction. The Full dataset includes 600,000 training and 130,000 testing samples per class, while the Polarity dataset includes 1,800,000 training and 200,000 testing samples per class. Also, the fields employed are the title and the content review. Tables 8 and 9, show one example of each class of the Amazon Full dataset and Amazon Polarity, respectively.

Table 8 – Amazon Reviews Full's instance examples

Instance – "Tittle", "Description"	Class
"Wonderful Pattern", "This is a great pattern for a special occasion or every day use... well worth it!!!"	5
"ok", "i liked good book from the past....great that we can look back on these great books and enjoy them ."	4
"OK Movie","This was a typical cop show and if you are looking for some pretty mindless entertainment this movie hits the spot."	3
"needs $$ upgrade", "Only has limited access to OBDII data stream unless you pay more money to upgrade the software."	2
"My brain froze over", "This supposed thriller is mind-numbingly dull and inept - I had to give up half way through."	1

Table 9 – Amazon Reviews Polaity's instance examples

Instance – "Tittle", "Description"	Class
"Wonderful Pattern", "This is a great pattern for a special occasion or every day use... well worth it!!!"	Positive
"My brain froze over", "This supposed thriller is mind-numbingly dull and inept - I had to give up half way through."	Negative

8 EXPERIMENTS AND RESULTS

8.1 EXPERIMENTS CONSIDERING THE NUMBER OF PARAMETERS

The use of Temporal Depthwise Separable Convolution (TDSC) promoted a significant reduction in the number of parameters held by the convolutions compared to the Very Deep Convolutional Neural Networks (VDCNN) model. For the first depth configuration, depth-9, the number of parameters went from 1.76 to 0.71 million, presenting a reduction of 59.66%. At the same time, for the more in-depth configuration depth-49, the number of parameters passed from 7.36 to 2.64 million, conferring a reduction of 64.13%. Considering all depth configurations, the overall parameter reduction achieved by the usage of TDSC was 62.42%. The Char-CNN model, which has six convolutional layers (depth-6), hols 1.37 million of parameters. Table 10 shows the number of parameters holds by the convolutional layers for each compared model.

Table 10 – Number of parameters hold by convolutional layers in millions. Best results by depth denoted in bold. The smaller the better.

	Network Depth Configuration				
	6	9	17	29	49
SVDCNN	–	**0.71**	**1.43**	1.56	2.64
VDCNN	–	1.76	3.85	4.22	7.38
Char-CNN	**1.37**	–	–	–	–

The network size reduction obtained by the use of the Global Average Pooling (GAP) technique is even more representative since both compared models use three Fully Connected (FC) layers for their classification tasks. Considering a problem with four target classes, the VDCNN model holds 12.59 million parameters on its classification layers, while the Char-CNN model holds 11.34 million parameters. The Squeezed Very Deep Convolutional Neural Networks (SVDCNN) model, on the other hand, does not have any FC layers, since they were entirely replaced by one convolutional layer and a global average pooling layer, as explained in section 6.2. Thus, the classification layers of the proposed model contain only 2,048 parameters, representing a reduction of 99.98% compared to the base model.

Comparing SVDCNN with VDCNN, the total number of parameters of the network has passed from 14.35 to 0.71 million for the depth-9, while for the depth-49, it moved from 19.97 to 2.64 million. Thus the proposed model is between 86.8% and 95.1% lighter than its base model. Following with the same comparison, the proposed model is between

79.23% and 94.41% smaller than the Char-CNN model, which counts with a total number of parameters of 12.71 million. Figure 17 shows the total number of parameters between the compared models.

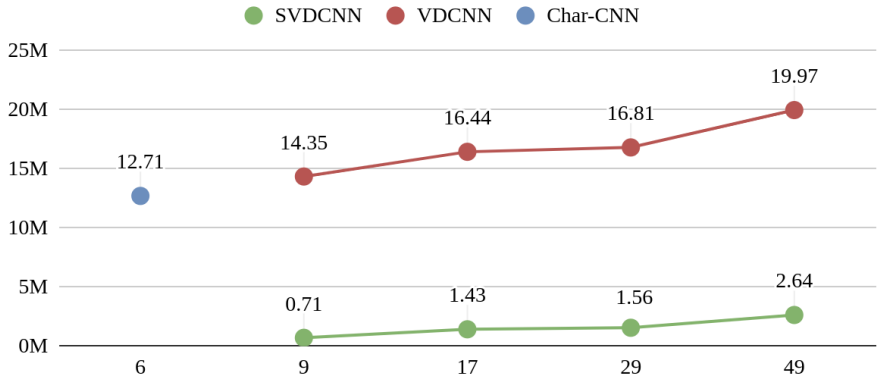

Figure 17 – Total number of parameters in Million.

8.2 EXPERIMENTS CONSIDERING THE STORAGE SIZE

The reduction of the total parameters impacts directly on the network storage size. While the most in-depth model of SVDCNN, with depth-49, has only 10MB, the shallowest VD-CNN model, with depth-9, uses 76MB of storage. Likewise, Char-CNN (which has depth 6) occupies 43.25MB. Figure 18 shows the storage size required for each compared model. This reduction is a very significant result for constrained platforms such as embedded systems and mobile devices. For example, Field Programmable Gate Array (FPGA) often have less than 10MB of on-chip memory and no off-chip memory or storage (HOWARD et al., 2017). Concerting to mobile devices, the usage of a reduced local model offers numerous benefits, such as no real-time network dependence, lower latency speed, increase scalability, privacy, and data saving.

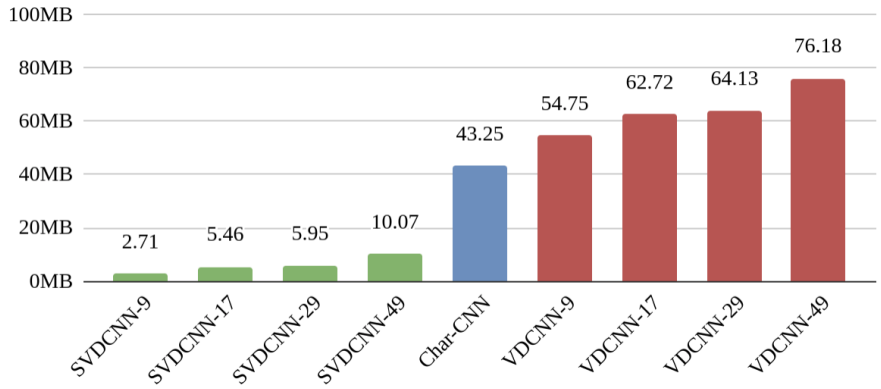

Figure 18 – Storage size required to embedding the compared models in MB.

8.3 EXPERIMENTS CONSIDERING THE SHORTCUT CONNECTIONS

As described in section 3.3, the base model presents worse results, for the depths-9, depth-17, and depth-29 architecture configurations, when shortcut connections are enabled. While for the depth-49 architecture configuration, the accuracy result is better when using the shortcut connections but not better than the results obtained by the depth-29 without shortcut. The accuracy results mentioned above were verified on the Yelp Full dataset, and the author concludes that the usage of shortcut connections does not bring any benefit to the VDCNN model.

On the other hand, the SVDCNN model has shown positive results while using shortcut connections. Over the same Yelp Full dataset, the usage of shortcut connections increases the accuracy of the depth-17, depth-29, and depth-49 architecture configurations. For the depth-49 configuration, the model is not even able to train without shortcut connections, presenting a pour accuracy performance of 28,81. Still, similar to the base model, the depth-49 configuration of the SVDCNN does not overcome the accuracy of its depth-29 configuration. Table 11 shows the results obtained by the VDCNN and SVDCNN models over the Yell Full dataset with and without shortcut connections, as well as the variation.

Table 11 – Accuracy results of VDCNN and SVDCNN over the Yelp Full dataset, without and with Shortcut Connections. Best values by depth by model in bold.

	Accuracy		
	Without Shortcut	With Shortcut	Variarion
VDCNN			
9	**62,37**	59,73	-2,64
17	**63,90**	60,82	-3,08
29	**64,72**	63,99	-0,73
49	62,59	**63,85**	1,26
SVDCNN			
9	**62,30**	62,04	-0,26
17	63,26	**63,32**	0.06
29	61,05	**63,39**	2,34
49	28,81	**63,20**	34,39

The main point about the good results presented by the SVDCNN model in contrast with the VDCNN model, concerning the use of shortcut connections, is that in practical terms, the proposed model is more profound than its base model due to the usage of depthwise separable convolutions. As described in section 6.1, each standard temporal convolution of the convolutional blocks was substituted by two convolutions: one depth-

wise and one pointwise. Since it is not possible to use them separately, keeping the same propose, we adopted to count them as one to make easier the comparison between the models. Still, the real depths of the proposed model are more extensive. Table 12, presents the practical depths of the SVDCNN model, while table 2 presents the depths of the VDCNN model.

While the VDCNN model has the depth-9, depth-17, depth-29, and depth-49 architecture configurations, its analogous architecture configurations in the SVDCNN model are in fact, depth-18, depth-34, depth-58, and depth-98, respectively. Again, since each depthwise and pointwise convolution are functional only when working together, we choose to count them as one to compare the models. Still, concerning the usage of shortcut connections, it is essential to clarify the reason why it implies better accuracy results on the proposed model: its architecture is, in practice, way more profound than the base model architecture.

Table 12 – Adopted and Practical depths of the SVDCNN architecture and its number of convolutional layers

Adopted Depth	9	17	29	49
Last Convolutional Layer	1	1	1	1
Convolutional Block 512	4	8	8	12
Convolutional Block 256	4	8	8	20
Convolutional Block 128	4	8	20	32
Convolutional Block 64	4	8	20	32
First Convolutional Layer	1	1	1	1
Pratical Depth	18	34	58	98

8.4 EXPERIMENTS CONSIDERING THE INFERENCE TIME

Deep learning processing architectures have the property of offering high parallelization power. Therefore, it is expected a fewer inference time while performing predictions on dedicated deep learning hardware in comparison to non-dedicated ones. Nevertheless, this is not the only aspect that determines the inference time of a model.

The parallelization potential of a model can vary according to its architecture. As we have more parameters per layers, more parallelizable a model tends to be, while the increase of the depth gets the opposite result. Another natural comprehension fact is that if a model has few parameters, there exists less content to be processed, and then we have a faster inference time. Concerning constrained platforms, like embedded systems and mobile devices, the presence of dedicated deep learning hardware such as a Graphics

Processing Unit (GPU) is not entirely feasible. This kind of hardware usually requires more energy and dissipates more heat, two undesirable features for such constrained platforms.

A metric that can be deduced from the inference time is the inference time ratio. This metric is the quotient between Central Processing Unit (CPU) and GPU inference times, and it is a general indicator of how independent of dedicated hardware is the inference time performance of a model. The range of values obtained by this metric varies between 0 and 1. The closer to 1, the smaller is the difference of the inference time due to the use of dedicated hardware — the closer to 0 the opposite effect. Consequently, the closer to 1 the inference time ratio of a model, the more it tends to be useful for constrained platforms.

Table 13 – Time results for AG's News dataset. For GPU, CPU, and Ratio, best results by depth denoted in bold.

	Inference Time		
	GPU	CPU	Ratio
SVDCNN			
9	3.95ms ± 0.25	**15.82ms±0.77**	**0.25**
17	6.62ms ± 0.27	**28.50ms±1.60**	**0.23**
29	10.90ms ± 0.36	46.78ms ± 2.70	**0.23**
VDCNN			
9	**3.12ms±0.23**	21.06ms ± 0.62	0.15
17	**5.08ms±0.22**	31.50ms ± 1.42	0.16
29	**7.55ms±0.29**	**44.30ms±2.94**	0.17
Char-CNN			
6	7.72ms ± 0.42	216.71ms ± 4.89	0.04

Concerning to SVDCNN model, each convolutional layer, of the convolutional blocks, was substituted by two convolutions, as shown in Fig. 13. As explained in Sections 6.1 and 8.3, we decided to count the two convolutions as only one layer in the network depth calculus. Nevertheless, compared to the baseline model, the number of convolutions per convolutional block has essentially doubled, for the same depth configuration. Due to this particularity, the SVDCNN tends to show higher inference times compared to the base model on high parallelizable platforms, like GPU. On the other hand, due to the notable parameter reduction, SVDCNN is expected to obtain better results than the baseline model over constrained platforms, which deal with memory and processing constraints.

The inference times obtained for the three compared models and their respective ratios are available in Table 13. When performing predictions over GPU, the proposed model presented results with a small performance loss compared to the inference time of the

baseline. As aforementioned, this result happens due to the architectural change proposed, where two convolutions substituted each convolutional layer of the convolutional blocks. On the other hand, the SVDCNN model presented two desirable properties for constrained systems deployment. First, the SVDCNN obtained lower inference times than VDCNN model when executing on non-dedicated hardware (CPU). Applying the Wilcoxon Paired test, the SVDCNN results for the depth-9 and depth-17 architecture configurations were significantly lower in comparison with the baseline model, with a confidence level of 95%. For depth-9, it was obtained 15.82ms against 21.06ms and for depth 17, 30.65ms against 28.50ms of the baseline model. Second, the inference time ratio was higher in all depth levels compared, 0.24 against 0.15 on average. It indicates that the SVDCNN model is more undependable of dedicated hardware then the baseline model. Looking to Char-CNN, this model got notably inferior results compared to the proposed method, with 216.71ms of CPU inference time and a Ratio of 0.04.

8.5 EXPERIMENTS CONSIDERING THE ACCURACY PERFORMANCE

Regarding accuracy performance, usually, a model with such parameter reduction should present some loss of accuracy in comparison to the original model. Nevertheless, the difference in accuracy between VDCNN and SVDCNN models is at most 1%, which is pretty modest, considering the parameters and storage size reduction aforementioned. In Table 14, it is possible to see the accuracy results obtained by the compared models. Still, the proposed model keeps overcoming the Char-CNN accuracy on the three largest datasets analyzed. Another significant result obtained is that the base property of VDCNN model was preserved on its squeezed version: the performance still increasing up with the depth.

Table 14 – Accuracy results. Best global results in bold.

	SVDCNN			VDCNN			Char-CNN
	9	17	29	9	17	29	6
Accuracy							
Ag News	90.23	90.54	91.13	90.83	91.12	91.27	**92.36**
Yelp Polarity	95.11	95.22	95.26	95.12	95.50	**95.72**	95.64
Yelp Full	62.30	63.32	63.39	63.27	63.93	**64.26**	62.05
Amazon Full	61.34	62.18	62.35	62.05	62.61	**63.00**	59.57
Amazon Polarity	95.08	95.23	95.39	95.30	95.59	**95.69**	95.07

Now, having as target a platform with internal storage constraints, the proposed model becomes a notably profitable option. For all evaluated datasets, the SVDCNN with depth-29 configuration overcomes the accuracy of the VDCNN model with depth-9 configuration (see Table 14). However, the first one has 5.95MB of storage size, while the last one is more the 9x bigger: 54.75MB. To become this benefit more clear see Figures 19 and 20.

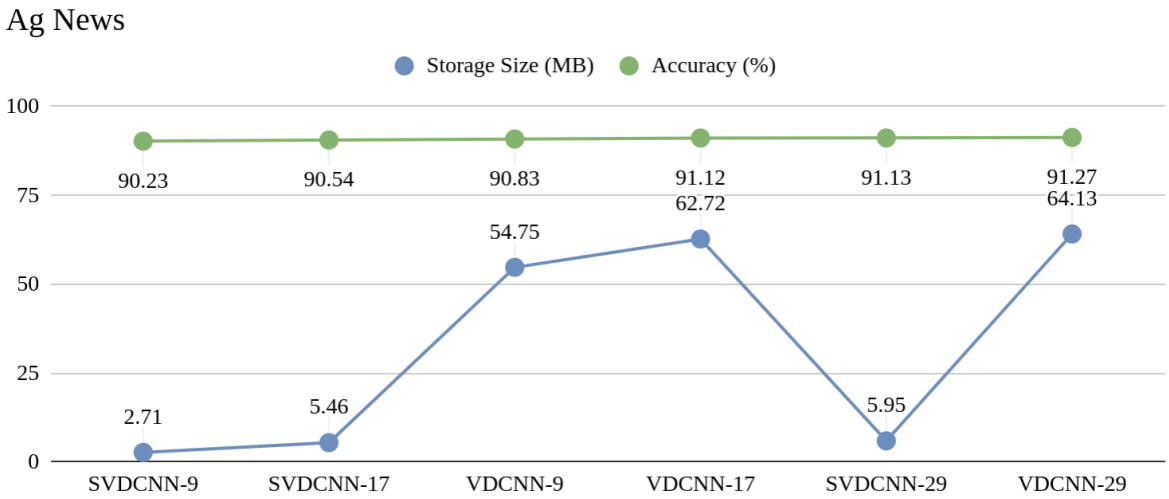

Figure 19 – Accuracy vs Storage Size comparison over Ag News dataset.

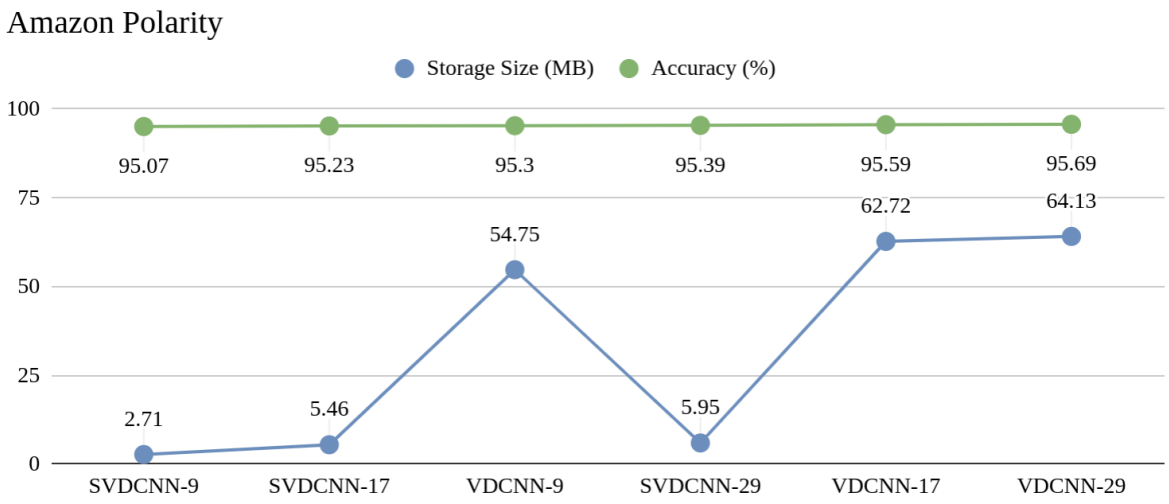

Figure 20 – Accuracy vs Storage Size comparison over Amazon Polarity dataset.

These figures show an accuracy comparison considering the storage size of the model for the Ag News and Amazon Polarity datasets, which are respectively the smallest and the biggest evaluated. Through the SVDCNN model, it is possible to obtain state-of-the-art accuracy results with significantly less storage size requirements.

9 CONCLUSION

9.1 FINAL CONSIDERATIONS

The increasing of network depth on CNN models has been a safe direction to obtain better accuracy results for image-related tasks. Also, although the depth increase results in more parameters to the network, structural network modifications were successfully proposed to make these models light-weight without significant accuracy loss.

For text-related tasks, though, there was a belief that the accuracy of CNN models could be improved by increasing its network depth. This thought changed when, in 2018, Conneaut et al. proposed the VDCNN model, a very deep CNN for text classification, that holds up to 29 layers presenting increasingly better accuracy results. This research was a mark for the CNN text-classification field, were the most profound model proposed had up to 6 layers. Still, being a light-weight model was not one of the objectives of the VDCNN. Its architecture relies on thousand of parameters, which make it unfeasible of being locally embedded on constrained platforms. Although constrained platforms do not have enough processing capabilities to train a sophisticated deep learning model, they are already capable of performing inference locally by using a previously trained embedded model. This approach enables numerous advantages such as more privacy, smaller response latency, and no real-time network dependence.

The growing availability of text data, as well as the ascension of embedded systems and mobile devices, experimented in recent years, motivates the research for structural network modifications over deep neural networks for text-related tasks aiming at its weight reduction. In this context, we presented the SVDCNN, a Squeezed version of the VDCNN model. The primary goal of this work was the reduction of parameters compared to state of the art Convolutional Neural Network (CNN) for text classification. This reduction made the SVDCNN model feasible of being deployed on constrained platforms while presenting minimal accuracy loss compared to the baseline model. To achieve this goal, we analyzed the impact of including TDSC and GAP besides the usage of skip connections.

The proposed model reduces about 92.45% the number of parameters and storage size. While the size of VDCNN model varies between 54.75 and 64.16MB, the size of SVDCNN model varies only between 2.80 and 6.03MB. Concerning accuracy performance, the network experiences an acceptable loss, below 1%. Still, differently from the base model, where the use of skip-connection did not improve the network accuracy, its use on the SVDCNN model promoted significant accuracy gains. While the use of skip-connection decreased the accuracy performance of VDCNN in 2.15% on average, to the SVDCNN model, its use increased the accuracy performance in 0.72%. This behavior occurs because, in practical terms, the SVDCNN model is deeper than the base model since each temporal

standard convolutional layer was substituted two convolutions, one depth-wise and one point-wise (review section 8.3). Therefore, skip-connections became more profitable in the proposed model.

Secondary goals of this work covered the analysis of inference time and deep learning dedicated hardware dependence. Although the difference in inference times obtained by the baseline and the proposed model is small, the difference is still statistically significant. Compared to VDCNN model, SVDCNN model presents lower inference time over CPU processing for depth 9 and 17, while offers a higher inference time ratio, 0.24 against 0.15 on average. Therefore the proposed model is less dependent on dedicated deep learning hardware, which is often a desirable behavior for constrained platforms.

As mentioned before, smaller CNN models enable its deployment on constrained platforms, such as embedded systems and mobile device. FPGA, for example, usually offer less than 10MB of on-chip memory and no off-chip memory or storage (HOWARD et al., 2017). Concerning entry-level mobile devices, there are Operational Systems (OS) with several restrictions for maximum application size. For example, on Android GO OS, applications should be less than 40MB on the device, while games should be less than 65MB on the device (GOOGLE, 2020). Also, the usage of a reduced local model offers numerous benefits, such as no real-time network dependence, lower latency speed, increase scalability, privacy, and data saving. In the image classification field, several shrunk CNN models were proposed (HOWARD et al., 2017; IANDOLA et al., 2016; SANTOS et al., 2018). However, to the best of our knowledge, works on this direction for the text classification field are still scarce, despite the several critical real-world applications which depend on text classification tasks such as sentiment analysis, recommendation, and opinion mining.

Lastly, part of this research is presented in the paper Squeezed Very Deep Neural Networks for Text Classification, published in the ICANN 2019 conference (DUQUE et al., 2019) and, the source code of the proposed model is available in the GitHub repository SVDCNN[2]

9.2 FUTURE WORKS

This work opens several new research directions. Concerning architectural changes, other pooling techniques can be applied and analyzed, aiming for further accuracy improvements. Also, other techniques able to reduce storage size after the model be trained can be applied, such as model compression (GONG et al., 2014).

Finally, another direction is to modify other sophisticated network architectures for text classification, besides CNN models, to become them more light-weight, and feasible of being locally embedded in constrained platforms.

[2] Link: <https://github.com/lazarotm/SVDCNN-v2>

REFERENCES

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2017. p. 1251–1258.

CIRESAN, D. C.; MEIER, U.; GAMBARDELLA, L. M.; SCHMIDHUBER, J. Deep big simple neural nets excel on handwritten digit recognition. *CoRR*, abs/1003.0358, 2010. Disponível em: <http://arxiv.org/abs/1003.0358>.

CONNEAU, A.; SCHWENK, H.; BARRAULT, L.; LECUN, Y. Very deep convolutional networks for text classification. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. [S.l.]: Association for Computational Linguistics, 2017.

DUQUE, A. B.; SANTOS, L. L. J.; MACÊDO, D.; ZANCHETTIN, C. Squeezed very deep convolutional neural networks for text classification. In: TETKO, I. V.; KŮRKOVÁ, V.; KARPOV, P.; THEIS, F. (Ed.). *Artificial Neural Networks and Machine Learning – ICANN 2019: Theoretical Neural Computation*. Cham: Springer International Publishing, 2019. p. 193–207. ISBN 978-3-030-30487-4.

GAO, Y.; SINGH, R.; RAJ, B. Voice impersonation using generative adversarial networks. *CoRR*, abs/1802.06840, 2018. Disponível em: <http://arxiv.org/abs/1802.06840>.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: GORDON, G.; DUNSON, D.; DUDíK, M. (Ed.). *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Fort Lauderdale, FL, USA: PMLR, 2011. (Proceedings of Machine Learning Research, v. 15), p. 315–323. Disponível em: <http://proceedings.mlr.press/v15/glorot11a.html>.

GONG, Y.; LIU, L.; YANG, M.; BOURDEV, L. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial nets. In: GHAHRAMANI, Z.; WELLING, M.; CORTES, C.; LAWRENCE, N. D.; WEINBERGER, K. Q. (Ed.). *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014. p. 2672–2680. Disponível em: <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>.

GOOGLE, A. D. *Android (Go edition)*. 2020.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016a. p. 770–778.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016b. Disponível em: <http://arxiv.org/abs/1603.05027>.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997.

HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M.; ADAM, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

HUANG, G.; LIU, Z.; MAATEN, L. van der; WEINBERGER, K. Q. Densely connected convolutional networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.]: IEEE, 2017.

IANDOLA, F. N.; HAN, S.; MOSKEWICZ, M. W.; ASHRAF, K.; DALLY, W. J.; KEUTZER, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

ISOLA, P.; ZHU, J.; ZHOU, T.; EFROS, A. A. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. Disponível em: <http://arxiv.org/abs/1611.07004>.

KAISER, L.; GOMEZ, A. N.; CHOLLET, F. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*, 2017.

KIM, Y. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F.; BURGES, C. J. C.; BOTTOU, L.; WEINBERGER, K. Q. (Ed.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.

LE, H. T.; CERISARA, C.; DENIS, A. Do convolutional networks need to be deep for text classification? In: *The Workshops of the Thirty-Second AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2017.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 1, n. 4, p. 541–551, dez. 1989. ISSN 0899-7667. Disponível em: <http://dx.doi.org/10.1162/neco.1989.1.4.541>.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*. [S.l.: s.n.], 1998. p. 2278–2324.

LEWIS, D. D.; YANG, Y.; ROSE, T. G.; LI, F. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, v. 5, n. Apr, p. 361–397, 2004. Disponível em: <http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf>.

LIN, M.; CHEN, Q.; YAN, S. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

LOTTER, W.; KREIMAN, G.; COX, D. D. Unsupervised learning of visual structure using predictive generative networks. *CoRR*, abs/1511.06380, 2015. Disponível em: <http://arxiv.org/abs/1511.06380>.

MCAULEY, J.; LESKOVEC, J. Hidden factors and hidden topics: Understanding rating dimensions with review text. In: *Proceedings of the 7th ACM Conference on Recommender Systems*. New York, NY, USA: ACM, 2013. (RecSys '13), p. 165–172. ISBN 978-1-4503-2409-0. Disponível em: <http://doi.acm.org/10.1145/2507157.2507163>.

MCCULLOCH, W. S.; PITTS, W. Neurocomputing: Foundations of research. In: ANDERSON, J. A.; ROSENFELD, E. (Ed.). Cambridge, MA, USA: MIT Press, 1988. cap. A Logical Calculus of the Ideas Immanent in Nervous Activity, p. 15–27. ISBN 0-262-01097-6. Disponível em: <http://dl.acm.org/citation.cfm?id=65669.104377>.

REED, S.; AKATA, Z.; YAN, X.; LOGESWARAN, L.; SCHIELE, B.; LEE, H. Generative adversarial text to image synthesis. In: BALCAN, M. F.; WEINBERGER, K. Q. (Ed.). *Proceedings of The 33rd International Conference on Machine Learning*. New York, New York, USA: PMLR, 2016. (Proceedings of Machine Learning Research, v. 48), p. 1060–1069. Disponível em: <http://proceedings.mlr.press/v48/reed16.html>.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, p. 65–386, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Representations by Back-propagating Errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986. Disponível em: <http://www.nature.com/articles/323533a0>.

SANTOS, A. G.; SOUZA, C. O. de; ZANCHETTIN, C.; MACEDO, D.; OLIVEIRA, A. L. I.; LUDERMIR, T. Reducing SqueezeNet storage size with depthwise separable convolutions. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. [S.l.]: IEEE, 2018.

SIFRE, L.; MALLAT, S. *Rigid-motion scattering for image classification*. Tese (Doutorado) — Citeseer, 2014.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2015.

SOCHER, R.; PENNINGTON, J.; HUANG, E. H.; NG, A. Y.; MANNING, C. D. Semi-supervised recursive autoencoders for predicting sentiment distributions. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, 2011. p. 151–161. Disponível em: <https://www.aclweb.org/anthology/D11-1014>.

SUNDERMEYER, M.; NEY, H.; SCHLUTER, R. From feedforward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Institute of Electrical and Electronics Engineers (IEEE), v. 23, n. 3, p. 517–529, mar 2015.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 1–9.

TAI, K. S.; SOCHER, R.; MANNING, C. D. Improved semantic representations from tree-structured long short-term memory networks. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. [S.l.]: Association for Computational Linguistics, 2015.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 5998–6008.

WIDROW, B.; HOFF, M. E. Adaptive switching circuits. In: *1960 IRE WESCON Convention Record, Part 4*. New York: IRE, 1960. p. 96–104.

ZEILER, M.; FERGUS, R. Visualizing and understanding convolutional networks. In: *Computer Vision, ECCV 2014 - 13th European Conference, Proceedings*. [S.l.]: Springer Verlag, 2014. (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), PART 1), p. 818–833. ISBN 9783319105895.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. In: *Computer Vision – ECCV 2014*. [S.l.]: Springer International Publishing, 2014. p. 818–833.

ZHANG, X.; ZHAO, J.; LECUN, Y. Character-level convolutional networks for text classification. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 649–657.