Pós-Graduação em Ciência da Computação

João Antônio Chagas Nunes

Additive Margin Softmax e funções Sinc para Reconhecimento de Locutor

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

Recife
2020

João Antônio Chagas Nunes

Additive Margin Softmax e funções Sinc para Reconhecimento de Locutor

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Inteligência Computacional
Orientador: Prof. Cleber Zanchettin

Recife
2020

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

N972a Nunes, João Antônio Chagas
Additive margin softmax e funções *sinc* para reconhecimento de locutor / João Antônio Chagas Nunes. – 2020.
92 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. Reconhecimento de locutor. I. Zanchettin, Cleber (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2020 - 122

João Antônio Chagas Nunes

“Additive Margin Softmax e funções Sinc para Reconhecimento de Locutor”

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: 02 de março de 2020.

BANCA EXAMINADORA

_____

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

_____

Prof. Dr. Everton Barbosa Lacerda
Callere Document Solutions/Pesquisa e Desenvolvimento

_____

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

Dedico esta dissertação a toda minha família, aos amigos e professores que me deram o apoio necessário para chegar até aqui.

AGRADECIMENTOS

Agradeço a Deus por me ter dado oportunidade de estudar e sempre prover os meios necessários para que eu seguisse na caminhada. Agradeço aos meus pais, meu irmão e Nena por sempre acreditarem em mim, sem eles eu tenho certeza que não teria chegado até aqui. Agradeço a minha namorada, Isabela, pela paciência que vem tendo desde a época do TCC.

Agradeço aos amigos desde a época da UAG, Marrone, Jefferson e Marcionilo, e aos do trabalho, em especial a Fernando, David, Ellen e Walber, que contribuíram com discussões, correções e ideias durante toda a fase de desenvolvimento do projeto.

Ao meu orientador, Cleber Zanchettin, e a David Macedo que indicaram a linha de pesquisa e que sempre me deram uma luz quando eu estava perdido. Agradeço também pelo incentivo e pelas correções que ambos fizeram em relação aos artigos.

RESUMO

Reconhecimento de locutor é uma tarefa desafiante com aplicações em diversas áreas, como autenticação, automação e segurança. O *SincNet* é um novo modelo baseado em aprendizado profundo (*deep learning*) com resultados promissores para tarefa de reconhecimento de locutor. Um fator crucial no treinamento de modelos de *deep learning* é a função de ativação utilizada, que possui impacto direto no desempenho do modelo treinado. A função de ativação *Softmax* é amplamente utilizada neste contexto, principalmente em problemas de classificação. Entretando, em alguns tipos de problemas, como por exemplo o reconhecimento facial, a amsoftmax tem apresentado resultados significativos quando comparados à versão tradicional do *Softmax*. A amsoftmax é uma nova função de ativação baseada na *Softmax* que introduz uma margem de separação aditiva entre as classes mapeadas. A margem de separação aditiva força as amostras da mesma classe a ficarem mais próximas umas das outras enquanto maximiza a distância de amostras de classes distintas. Neste trabalho foram propostas variações de modelos tradicionais considerando componentes como amsoftmax e as camadas *sinc* do modelo *SincNet* para o problema de reconhecimento de locutor. Dentre os modelos propostos se destacam o amsincnet e o *AM-MobileNet1D*. O amsincnet é um modelo baseado no *SincNet* que usa a função de ativação amsoftmax, e com isso foi possível obter um erro de classificação 55% menor que o obtido pelo *SincNet* tradicional nas bases de dados *TIMIT* e *MIT*, sem aumento significativo na complexidade do modelo. O *AM-MobileNet1D* é uma versão da rede *MobileNet V2* adaptada para trabalhar com sinais de áudio, que apresentou resultados até sete vezes mais rápidos que o modelo base *SincNet*, sem prejuízo no desempenho do modelo.

Palavras-chaves: Reconhecimento de Locutor. Deep Learning. Softmax. AM-Softmax. MobileNet.

ABSTRACT

Speaker Recognition is a challenging task with essential applications such as authentication, automation, and security. SincNet is a new deep learning based model which has produced promising results to tackle the mentioned task. To train deep learning systems, the activation function on the final layer is essential to the network performance. The Softmax activation function is a widely used function in deep learning methods, but it is not the best choice for all kind of problems. For distance-based problems, one new Softmax based activation function called Additive Margin Softmax (AM-Softmax) is proving to be a better choice than the traditional Softmax. The AM-Softmax introduces a margin of separation between the classes that forces the samples from the same class to be closer to each other and also maximizes the distance between classes. In this paper, we proposed several deep learning models to tackle the speaker recognition problem. In addition, it was made several experiments to analyse the influence of the AM-Softmax function and the Sinc layer on the speaker recognition problem. Among the proposed models, the AM-SincNet and the AM-MobileNet1D had promissing results. The proposed AM-SincNet model is based on the SincNet but uses an improved AM-Softmax layer, it had shown a classification error about 55% smaller than the tradicional SincNet model on the datasets TIMIT and MIT. On the other hand, the AM-MobileNet1D is an adapted version of MobileNet V2 built to deal with audio signals, it had shown results up to 7 times faster than the SincNet, while keeping low error rates.

Keywords: Speaker Recognition. Deep Learning. Softmax. AM-Softmax. MobileNet.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM-MobileNet1D	*Additive Margin MobileNet1D*
AM-SincNet	*Additive Margin SincNet*
AM-Softmax	*Additive Margin Softmax*
ANN	*Artificial Neural Networks*
BN	*Batch Normalization*
CER	*Classification Error Rate*
CNN	*Convolutional Neural Networks*
DNN	*Deep Neural Networks*
FBANK	*Filter Bank*
FER	*Frame Error Rate*
GA	*Gated Activation*
Gaussian PLDA	*Gaussian Probabilistic Linear Discriminant Analysis*
GPU	*Graphics Processing Unit*
Heavy-tailed PLDA	*Heavy-tailed Probabilistic Linear Discriminant Analysis*
IJCNN	*International Joint Conference on Neural Networks*
Leaky-ReLU	*Leaky Rectified Linear Unit*
LN	*Layer Normalization*
MFCC	*Mel-Frequency Cepstrum Coefficients*
PLDA	*Probabilistic Linear Discriminant Analysis*
ReLU	*Rectified Linear Unit*
RNN	*Recurrent Neural Networks*
TanH	Tangente Hiperbólica

SUMÁRIO

1 INTRODUÇÃO

Reconhecimento de locutor (identificação de quem está falando) é uma tarefa essencial com aplicações em autenticação biométrica, identificação, segurança, entre outros (BEIGI, 2011). A área é dividida em duas subtarefas principais: Identificação e Verificação de locutor. Na identificação de locutor (Figura 1), dada uma amostra de áudio, o modelo deve identificar a qual dos locutores, dentro de uma pré-determinada lista de locutores (base de dados), a amostra de áudio pertence. Por outro lado, na Verificação de locutor (Figura 2), o modelo classifica duas amostras de áudio como sendo da mesma classe ou não, ou seja, o classificador deve identificar se essas duas amostras pertencem ao mesmo locutor. A maioria dos métodos para reconhecimento de locutor presentes na literatura utilizam *i-vectors* (DEHAK et al., 2010) como extratores de características e para tarefa de classificação utilizam métodos como *Probabilistic Linear Discriminant Analysis* (PLDA) (PRINCE; ELDER, 2007), *Heavy-tailed Probabilistic Linear Discriminant Analysis* (Heavy-tailed PLDA) (MATEJKA et al., 2011) e *Gaussian Probabilistic Linear Discriminant Analysis* (Gaussian PLDA) (CUMANI; PLCHOT; LAFACE, 2013a).

A dificuldade no reconhecimento de locutor está associada aos sinais de áudio, os quais são complexos de serem modelados em características de alto e baixo nível capazes de distinguir bem diferentes locutores. Métodos que extraem características procurando por padrões arbitrados por seres humanos têm um apelo maior por causa da interpretabilidade das características extraídas. Desta forma, os seus usuários conseguem entender melhor como o método está funcionando e quais padrões estão sendo utilizados para fazer a inferência. Contudo, apesar do apelo maior pela interpretabilidade, esses métodos podem não ser os melhores de fato, pois apesar de se saber quais padrões estão sendo procurados nos sinais de áudio, não se tem nenhuma garantia de que são os melhores para a tarefa em questão. Por outro lado, métodos baseados em *Deep Learning* têm o poder de abstrair padrões que os humanos podem não ser capazes de entender (devido a alta dimensionalidade do problema). Entretanto, eles geralmente conseguem obter melhores resultados do que os métodos tradicionais, mesmo necessitando de maior poder computacional para serem utilizados.

Apesar dos avanços nos anos recentes (CAMPBELL et al., 2006; KENNY et al., 2007; CUMANI; PLCHOT; LAFACE, 2013b; HEIGOLD et al., 2016; SNYDER et al., 2016; MCLAREN; LEI; FERRER, 2015; RICHARDSON; REYNOLDS; DEHAK, 2015), o reconhecimento de locutor ainda é uma tarefa desafiante. No últimos anos, as *Deep Neural Networks*

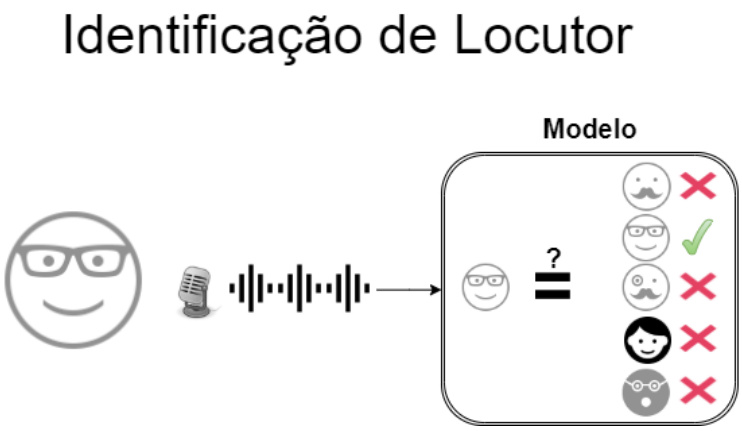


Figura 1 – Exemplo da tarefa de Identificação de Locutor.

Figura 2 – Exemplo da tarefa de Verificação de Locutor.

(DNN) (BENGIO, 2009) estão progressivamente substituindo os métodos tradicionais em tarefas de reconhecimento de padrões e processamento de sinais, como é o caso das *Convolutional Neural Networks* (CNN) (LECUN; BENGIO, 1998; KRIZHEVSKY; SUTSKEVER; HINTON, 2012a) que já se mostraram ser o estado-da-arte para trabalhar com classificação de imagens, detecção de objetos e outras tarefas relacionadas a imagens (WANG et al., 2017; He et al., 2016; REN et al., 2015). Da mesma forma, modelos de *Deep Neural Networks* (DNN) também estão sendo usados em combinação com métodos tradicionais ou em abordagens de solução ponta-a-ponta para tarefas de reconhecimento de locutor (SNYDER et al., 2016; TRIGEORGIS et al., 2016; JUNG et al., 2018). Nas abordagens híbridas é comum a utilização de *Deep Neural Networks* (DNN) para extrair as características do sinal de áudio e então codificar estas informações em um espaço vetorial de baixa dimensionalidade, de uma forma que as amostras pertencentes a mesma classe fiquem próximas umas das outras. Desta maneira, os vetores codificados em baixa dimensionalidade podem ser classificados utilizando classificadores tradicionais.

Uma interessante abordagem para a tarefa de reconhecimento de locutor base-

ada em *Deep Learning* é o modelo *SincNet* (Ravanelli; Bengio, 2018), pois ele consegue unificar o poder de processamento das abordagens baseadas em *Deep Learning* com a interpretabilidade das características extraídas através de padrões definidos por humanos. O *SincNet* usa um modelo de *Deep Learning* em conjunto com filtros *sinc* (Subseção 2.2.1) para conseguir extrair características relevantes para a classificação. O *SincNet* se mostrou melhor em comparação aos modelos tradicionais de *Convolutional Neural Networks* (CNN) que recebem como entrada tanto o sinal de áudio sem nenhum processamento prévio quanto os que recebem *Filter Bank* (FBANK). Os autores também realizaram experimentos que sugerem a superioridade do *SincNet* em relação a modelos de *Deep Neural Networks* (DNN) que recebem como entrada sinais de áudio processados por *Mel-Frequency Cepstrum Coefficients* (MFCC) (Subseção 2.2.2).

Apesar dos bons resultados obtidos pelo *SincNet*, no topo de sua arquitetura, o *SincNet* usa uma camada *Softmax* para mapear as características extraídas pela rede em um vetor de probabilidades para uma lista de possíveis resultados a fim de auxiliar no processo de separação das classes do problema tratado. A separação é feita otimizando uma superfície linear no espaço multidimensional responsável por isolar cada conjunto de amostras que representam uma classe na base de dados. Embora a função *Softmax* funcione bem para ajudar a otimizar essa superfície de separação, ela não é apropriada para diminuir a distância das amostras pertencentes a mesma classe, nem para aumentar a distância entre amostras pertencentes a classes distintas, pois no processo de otimização da superfície de separação as amostras não são forçadas a ficarem próximas do centro de suas classes. Essa característica pode prejudicar a eficiência do modelo em tarefas de reconhecimento de locutor, pois amostras localizadas muito próximas da superfície de separação do *Softmax* podem facilmente ser confundidas como pertencentes a outra classe. Para lidar com esse problema, novos métodos como o *Additive Margin Softmax* (AM-Softmax) (WANG et al., 2018) estão sendo propostos.

O AM-Softmax introduz uma margem de separação aditiva à superfície de separação das classes, o que força as amostras de classes distintas a ficarem mais afastadas umas das outras enquanto minimiza a distância das amostras pertencentes a mesma classe. A Figura 3 mostra como se comporta a margem de separação aditiva da AM-Softmax em comparação a superfície de separação da *Softmax* tradicional. Nela, os vetores W_1 e W_2 representam os centros da classe 1 e 2, respectivamente. P_0 é o limiar de separação da *Softmax*, enquanto P_1 e P_2 são a margem de separação da AM-Softmax. As regiões azul em forma de pizza são as regiões de interesse onde as amostras da classe 1 e 2 ficam localizadas quando o AM-Softmax é utilizado. Na

figura é possível ver que entre as regiões de interesse da classe 1 e 2 existe uma margem de separação definida pelos vetores P_1 e P_2 da AM-Softmax. Esse vetores são controlados pelo tamanho da margem utilizada e servem para criar uma região segura entre uma classe e outra, aumentando a distância entre amostras de classes distintas e diminuindo a distância entre amostras da mesma classe.

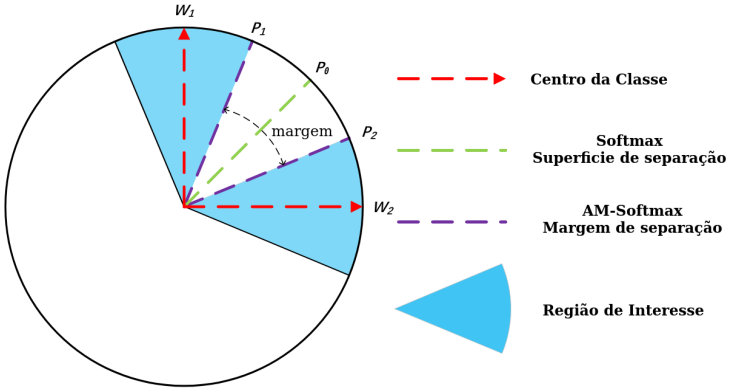


Figura 3 – Representação da *Softmax* e *Additive Margin Softmax* (AM-Softmax). **Adaptado de (WANG et al., 2018)**.

Outro ponto a ser considerado é que os modelos de *Deep Learning* são cada vez mais utilizados para agregar valor, desempenho e usabilidade em aplicações para usuários finais. Entretanto, este tipo de aplicação está aos poucos migrando para ambientes móveis. Dispositivos móveis (*mobile*) estão revolucionando muitos aspectos em nossas vidas, desde a forma em que nos comunicamos com outras pessoas até a forma em que viajamos utilizando *apps* de geolocalização, o que torna *smartphones* e outros dispositivos portáteis incrivelmente úteis. Com o massivo uso destes dispositivos, muitas aplicações que antes funcionavam consumindo os serviços de servidores, agora estão migrando para os dispositivos dos clientes, melhorando a usabilidade, performance e qualidade.

O problema com os modelos de *Deep Learning* é que eles não são naturalmente pensados e otimizados para dispositivos móveis. Eles geralmente são modelos grandes que ocupam muito espaço em disco e necessitam de um alto poder computacional para funcionar adequadamente, além de consumirem muita energia para serem executados. A maioria deles precisa de *hardware* dedicado, como *Graphics Processing Unit* (GPU) para terem um tempo de resposta razoável.

Ainda assim, alguns modelos de *Deep Learning*, como o *MobileNet*, são projetados para funcionar em dispositivos *mobile*. A arquitetura da *MobileNet* introduz o conceito de *depthwise separable convolution* (Seção 3.4), que é uma nova forma de aplicar convoluções capaz de tornar o modelo mais leve e rápido do que os modelos tradicionais de *Deep Learning*. O problema com a *MobileNet* é que ela foi projetada para trabalhar com imagens e não pode ser aplicada diretamente em sinais

de áudio. Desta forma, é necessário fazer modificações em sua arquitetura para adaptá-la ao problema da Identificação de Locutor.

1.1 OBJETIVOS

O objetivo deste trabalho é propor modelos de *Deep Learning* para o problema da Identificação de Locutor. São propostas melhorias em dois modelos da literatura no contexto do problema tratado. Os modelos investigados são o *SincNet* e o *MobileNet*. As melhorias são baseadas na mudança da *loss* utilizada nos modelos, assim como na adaptação das convoluções da *MobileNet* para melhor lidar com sinais de áudio. Nas análises são consideradas métricas como *Frame Error Rate* (FER) e *Classification Error Rate* (CER) para verificar sua acurácia, além de tempo de inferência, tamanho do modelo e quantidade de parâmetros. Para atender esse objetivo, o estudo apresenta os seguintes objetivos específicos:

- Levantamento de bases de dados para realizar os experimentos de identificação de locutor;
- Projetar novos modelos para identificação de locutor com base nos trabalhos já existentes na literatura;
- Avaliar o impacto da substituição da *Softmax* pela função melhorada AM-Softmax nos modelos propostos;
- Avaliar o uso de camadas de convolução *sinc* nos modelos propostos;
- Avaliar a influência das alterações propostas em relação aos modelos base considerando diferentes critérios de análise.

1.2 CONTRIBUIÇÕES

Foram propostos diferentes modelos de *Deep Learning* para a tarefa de identificação de locutor, dentre os modelos propostos o *Additive Margin SincNet* (AM-SincNet) (Chagas Nunes; Macêdo; Zanchettin, 2019) e o *Additive Margin MobileNet1D* (AM-MobileNet1D) se destacam pela baixa taxa de erro e pelo baixo tempo de inferência, respectivamente. Além dos dois modelos, também foi proposto o modelo *Gated Activation Net* que é baseado em funções de ativação do tipo *Gated Activation* (GA). O modelo proposto *Additive Margin SincNet* (AM-SincNet) foi publicado no *International Joint Conference on Neural Networks* (IJCNN) 2019, sob o nome de "*Additive Margin SincNet for Speaker Recognition*"e está disponível no IEEE [1].

[1] <https://ieeexplore.ieee.org/abstract/document/8852112>

O código para execução de ambos os modelos propostos está disponível no GitHub [2] [3].

1.3 ESTRUTURA DA DISSERTAÇÃO

Este trabalho está estruturado da seguinte forma: O Capítulo 2 apresenta um pouco do conceito base relacionado às Redes Neurais e seus principais componentes, assim como também aborda funções *sinc* que serão utilizadas nos capítulos subsequentes. O Capítulo 3 aborda trabalhos na literatura que estão relacionados ao desenvolvimento dos modelos propostos. No Capítulo 4 é descrito o funcionamento de cada um dos modelos propostos, assim como a inspiração em sua proposta. O Capítulo 5 apresenta os experimentos, descrevendo as bases de dados utilizadas, o processamento feito em cada amostra de áudio, como foi dividido os conjuntos de treinamento e teste em cada base de dados, as medidas de avaliações adotadas e o ambiente no qual foram realizados os experimentos. O Capítulo 6 expõe e compara os resultados obtidos em cada um dos experimentos realizados com os métodos propostos e com o método base *SincNet*. Por fim, no Capítulo 7 é feita a conclusão deste trabalho.

[2] <https://github.com/joaoantoniocn/AM-SincNet>
[3] <https://github.com/joaoantoniocn/AM-MobileNet1D>

2 REFERENCIAL TEÓRICO

Neste capítulo será abordado as principais operações utilizadas em Redes Neurais Convolucionais, como a convolução, *pooling*, função de ativação, normalização, camada totalmente conectada e *dropout*. Além da introdução das *Convolutional Neural Networks* (CNN) (Seção 2.1), na Seção 2.2 será abordado alguns métodos de processamento de sinais. Na Subseção 2.2.1 será descrito o funcionamento das funções *sinc* que serão utilizadas nos métodos propostos no Capítulo 4, assim como é parte crucial do *SincNet* (Seção 3.2). Por fim, a Subseção 2.2.2 terá uma breve descrição do *Mel-Frequency Cepstrum Coefficients* (MFCC).

2.1 REDES NEURAIS CONVOLUCIONAIS

Redes Neurais Convolucionais (CNN) (LECUN; BENGIO, 1998) têm sido uma das propostas mais utilizadas na última década para uma variedade de problemas relacionados a reconhecimento de padrões, desde de processamento de imagens até processamento de sinais, como os de áudio. As CNN são consideradas o estado da arte em problemas de reconhecimento de imagens, como reconhecimento facial, classificação de imagens, detecção e segmentação de objetos (ALOM et al., 2019) e (YANDONG HAO ZONGBO, 2016). Agora, elas também estão sendo utilizadas no campo do processamento de áudio em tarefas para reconhecimento de locutor (Chagas Nunes; Macêdo; Zanchettin, 2019; Ravanelli; Bengio, 2018; MCLAREN; LEI; FERRER, 2015; RICHARDSON; REYNOLDS; DEHAK, 2015), reconhecimento de discurso (GOODFELLOW; BENGIO; COURVILLE, 2016; RAVANELLI et al., 2017) e segmentação de áudio (HUSSAIN; HAQUE, 2018). Um dos aspectos mais importantes das CNN é a redução do número de parâmetros, nas camadas de convolução em relação a camadas totalmente conectadas, nas ANN. Essa redução possibilita aos pesquisadores modelarem grandes arquiteturas a fim de resolver problemas complexos (Albawi; Mohammed; Al-Zawi, 2017).

Apesar das redes neurais já existirem há algumas décadas, elas se tornaram mais populares nos anos recentes com o avanço no poder de processamento adquirido por meio do uso das GPUs para o processamento paralelo dos modelos. E tudo isso ocorreu alinhado aos novos modelos de CNN que se destacaram na última década em competições na área de processamento de imagens: *AlexNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012b), *Inception* (SZEGEDY et al., 2014), *VGG* (SIMONYAN; ZISSERMAN, 2015), *ResNet* (He et al., 2016) e *MobileNet* (HOWARD et al., 2017; Sandler et al., 2018).

Cada um dos modelos citados acima tem sua particularidade e funciona com uma arquitetura diferente. Por exemplo, enquanto a *VGG* aposta em uma arquitetura simétrica e é considerada como uma arquitetura padrão, a *Inception* faz uma série de convoluções com filtros de tamanho diferentes em paralelo. A *ResNet*, por sua vez, utiliza-se de seu módulo residual que permite treinar uma arquitetura com camadas mais profundas, enquanto a *MobileNet* aposta em um modelo mais leve com menos parâmetros e com uma inferência mais rápida. Embora cada modelo possua uma arquitetura diferente, todos eles pertencem a classe das CNN e compartilham algumas características que podem ser vistas na Figura 4, como a camada de convolução, *subsampling* ou *pooling* e a camada totalmente conectada que são descritas em mais detalhes nas subseções abaixo.

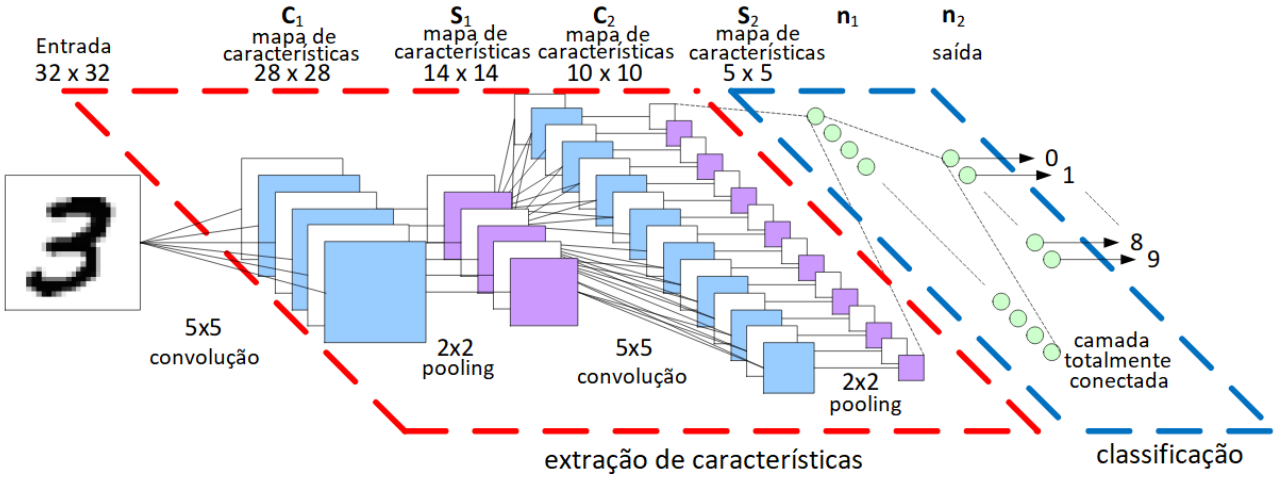


Figura 4 – Exemplo de arquitetura CNN para o reconhecimento de dígitos. **Adaptado de (PEEMEN; MESMAN; CORPORAAL, 2011)**.

2.1.1 Camada de Convolução

Na CNN, a camada de convolução é a responsável pela extração das características. Ela é composta por filtros que contêm os padrões a serem extraídos dos dados. Esses padrões são aprendidos junto com o processo de treinamento da rede e cada filtro é responsável por um padrão diferente. As características aprendidas pelos filtros variam de baixo a alto nível dependendo da localização do filtro na arquitetura do modelo.

Na Figura 5 é possível ver um exemplo do funcionamento de uma convolução em uma matriz que representa um canal de cor de uma imagem. A convolução é feita movendo o filtro através da matriz de entrada e multiplicando o filtro pela região da imagem na qual ele se encontra. A multiplicação é feita "*pixel*"a "*pixel*"de uma forma em que o *pixel* na primeira coluna da primeira linha do filtro é multiplicado

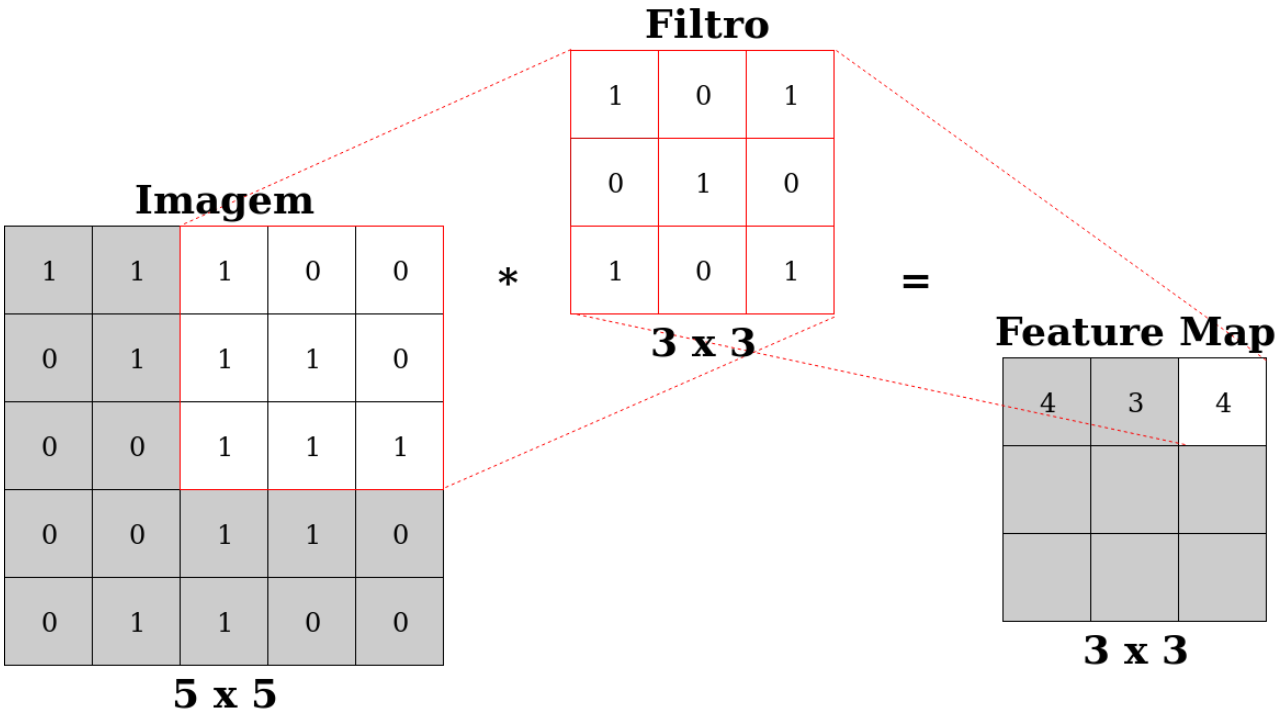


Figura 5 – Exemplo de uma convolução aplicada a uma imagem com filtro de tamanho 3×3.

apenas pelo *pixel* da primeira coluna na primeira linha da submatriz na qual o filtro está. Depois da multiplicação, é feita a soma dos valores adquiridos em cada *pixel* multiplicado. Essa soma junto com a multiplicação do filtro pela subimagem se torna uma característica no *feature map*. O processo de convolução é feito até que o filtro tenha percorrido toda a matriz de entrada (que representa a imagem de entrada). A Figura 6 ilustra como é gerada a submatriz ou subimagem a partir de uma janela de convolução, enquanto a Figura 7 mostra como é calculado o resultado de um filtro de convolução em uma subimagem ou submatriz.

O processo de convolução também conta com alguns parâmetros que podem ser ajustados e influenciarão no tamanho do *feature map* de saída. Um desses parâmetros é a quantidade de filtros que vai ser utilizada em cada convolução, desta forma o *feature map* ganha uma dimensão a mais, a qual indicará a quantidade de filtros utilizados. No exemplo da Figura 5 o *feature map* gerado tem dimensão $1 \times 3 \times 3$, pois foi feita uma convolução com apenas 1 filtro. Caso tivessem sido utilizados n filtros para essa convolução, o *feature map* resultante possuiria dimensão $n \times 3 \times 3$.

Um outro parâmetro a ser configurado no processo de convolução é o *stride*, ele indica o tamanho do passo que o filtro de convolução vai utilizar ao se mover na matriz de entrada. Quanto maior o tamanho do passo escolhido, menor será o *feature map* de saída. Um valor padrão para o *stride* é 1, o que significa que a janela de convolução vai se mover 1 *pixel* de cada vez durante o processo de convolução

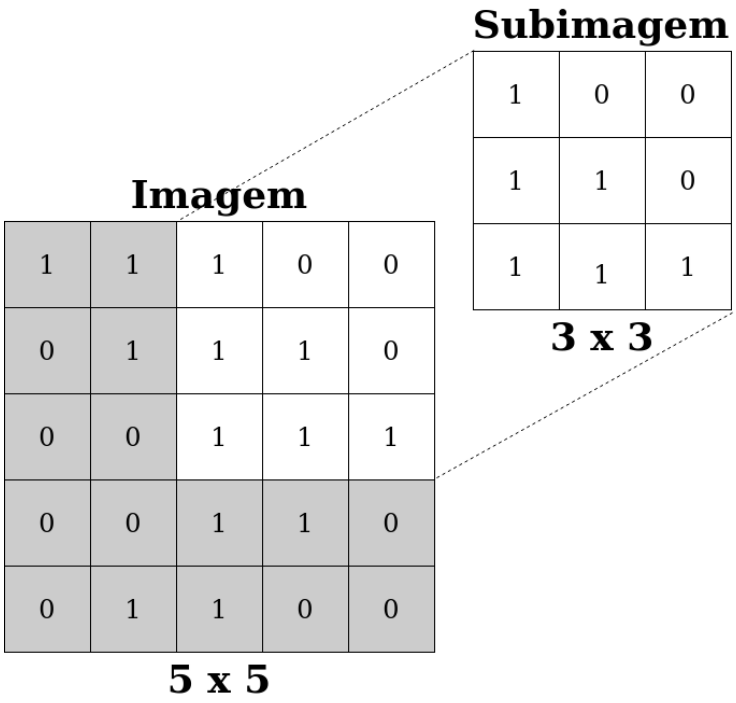


Figura 6 – Geração de uma subimagem a partir de uma janela de convolução.

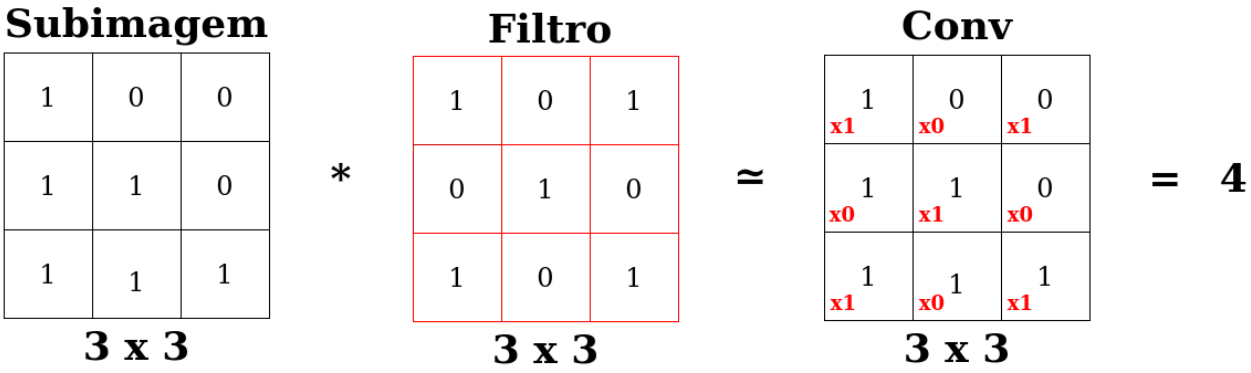


Figura 7 – Exemplo do funcionamento da aplicação do filtro de convolução aplicado a uma subimagem.

assim como feito no exemplo da Figura 5.

Por fim, um terceiro parâmetro importante a ser configurado em uma operação de convolução é o *padding*. Com o *padding* é possível configurar se é desejado adicionar uma margem de *pixels* ao redor da imagem, assim como quais valores são preferíveis colocar nessa margem. Um valor comum para preencher o *padding*, quando se opta por utilizá-lo, é o valor zero, que muitas vezes é chamado de *zero-padding*.

2.1.2 Camada de *Pooling*

A camada de *pooling*, também denominada de *downsampling* ou *subsampling*, é responsável por diminuir a dimensionalidade do *feature map* gerado pela camada de convolução. Existem algumas operações diferentes de *pooling* que têm o mesmo objetivo de reduzir o número de características extraídas na camada de convolução. Um dos tipos de operação de *pooling* mais utilizados é o *max pooling*, no qual uma janela desliza no *feature map* selecionando apenas as características que tiveram maior valor na camada de convolução. Um exemplo do *max pooling* pode ser visto na Figura 8, na qual foi utilizado um filtro de tamanho 2×2 com um *stride* de 2. Um dos motivos do *max pooling* ser uma escolha comum entre os tipos de *pooling* é que os filtros de convolução retornam um valor alto quando encontram o padrão que procuram em uma determinada região da imagem, desta forma, uma camada de *max pooling* preservaria apenas as respostas mais altas dos filtros de convolução mantendo assim as características mais relevantes extraídas pela camada de convolução enquanto remove os valores menos significantes.

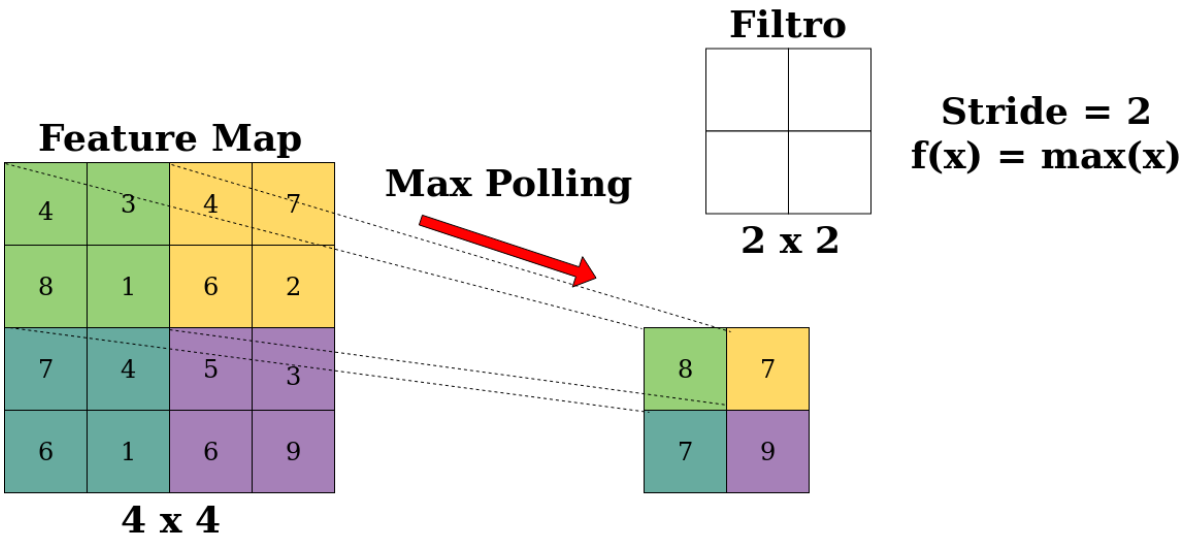


Figura 8 – Operação de *max pooling* em um *feature map* utilizando filtro de tamanho 2×2 e *stride* = 2.

Outra operação comum de *pooling* é o *average pooling* capaz de reduzir a dimensionalidade dos *feature maps* ao utilizar as médias das características em uma determinada janela, como pode ser visto na Figura 9. O *average pooling* constrói novas características levando em consideração todas as ativações do *feature map*, o que pode ser relevante em alguns tipos de problema.

É importante destacar a importância da camada de *pooling* nas Redes Neurais Convolucionais, pois elas atuam de forma a reduzir a dimensionalidade dos *feature maps* diminuindo o consumo de memória da rede ao descartar características não

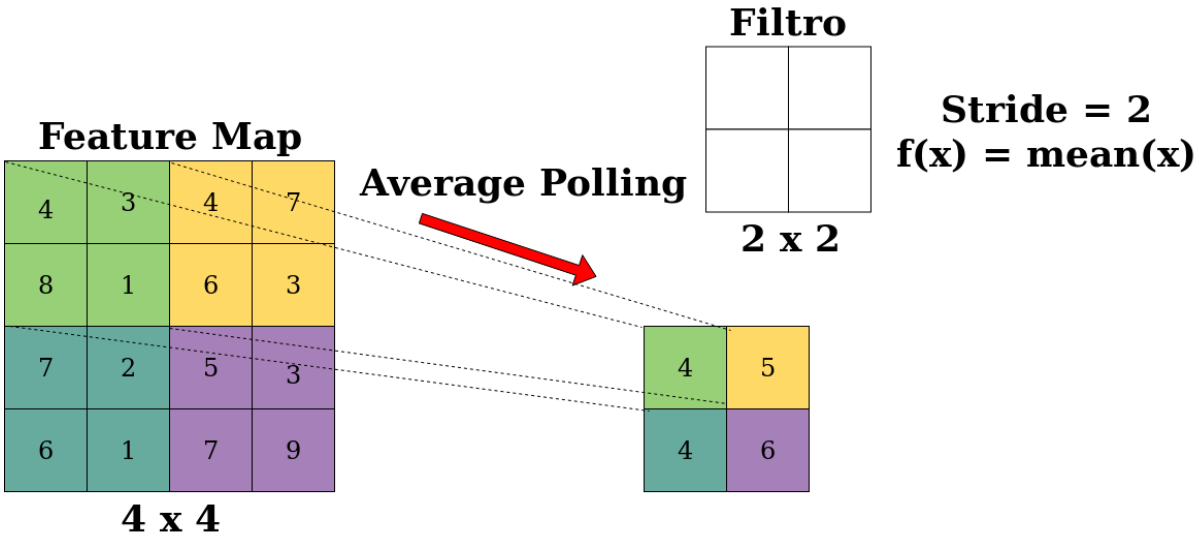


Figura 9 – Operação de *average pooling* em um *feature map* utilizando filtro de tamanho 2×2 e *stride* = 2.

relevantes, enquanto permite a construção de camadas mais profundas. A remoção de características não relevantes para o modelo também torna a execução da rede mais rápida pois evita uma série de operações mais complexas. Outra vantagem das camadas de *pooling* é que elas não adicionam complexidade ao modelo, uma vez que elas não possuem parâmetros que precisam ser aprendidos durante o treinamento da rede e, além disso, a aplicação de *pooling* em uma CNN a torna menos variante a translações dos padrões procurados pela rede.

2.1.3 Função de Ativação

Funções de ativação são responsáveis por controlar quando e como os neurônios são ativados na rede, além do intervalo de seus valores. Essa ativação limita quando e como uma informação vai ser propagada de uma camada para outra mais profunda. A função de ativação também é responsável por introduzir não linearidade aos dados à medida que eles vão propagando pela rede, o que sem ela uma rede neural seria apenas uma transformação linear do seus dados de entrada até o final da arquitetura (FAN, 2016).

Existe uma variedade de funções de ativação como a *Sigmoid*, TanH, *Softmax*, ReLU (NAIR; HINTON, 2010), Leaky-ReLU (MAAS; HANNUN; NG, 2013), entre diversas outras. A função de ativação *Sigmoid* é descrita pela Equação 2.1, ela faz o mapeamento dos dados de entrada para um intervalo de 0 a 1 e é centrada em $y = 0,5$ como pode ser visto na Figura 10 (a). Com um comportamento bem similar ao da *Sigmoid*, a TanH também tem seu gráfico em forma sigmoidal, porém, funciona em intervalos diferentes. Ela transforma os dados de entrada para um intervalo de -1

a 1 e é centrada em $y = 0$, como pode ser visto na Figura 10 (b). Sua fórmula é descrita na Equação 2.2 e como pode ser observado, ela é bem próxima da fórmula da *Sigmoid* descrita na Equação 2.1. A TanH também pode ser descrita em função da *Sigmoid* como pode ser visto na Equação 2.3.

Outra função bastante utilizada nas camadas intermediarias das CNN é a ReLU, ela introduz não linearidade aos dados a medida que zera valores negativos. A fórmula da ReLU é descrita na Equação 2.5 e a Figura 11 (a) apresenta seu comportamento no mapeamento dos dados. Uma função de ativação derivada da função ReLU é a Leaky-ReLU, ela se comporta de maneira parecida com a ReLU tradicional, porém, não zera os valores negativos como pode ser visto na Equação 2.6 e na Figura 11 (b). A função *Softmax* é geralmente utilizada na última camada do modelo, pois ela consegue mapear os dados para um intervalo de 0 a 1 de uma forma em que seu *output* é também a probabilidade atribuída a cada classe com soma 1. A *Softmax* é definida pela Equação 2.4.

$$f(x) = sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.2}$$

$$tanh(x) = 2sigmoid(2x) - 1 \tag{2.3}$$

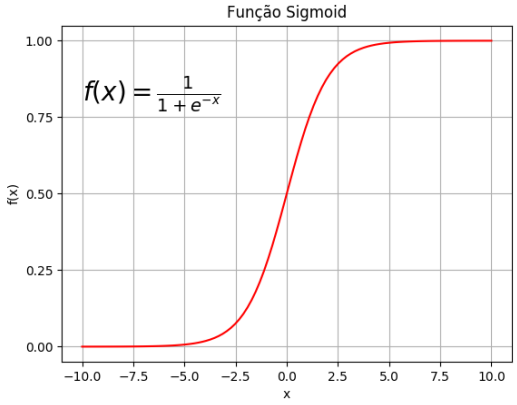
$$f(\mathbf{x})_i = softmax(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \tag{2.4}$$

$$f(x) = relu(x) = max(0,\ x) \tag{2.5}$$

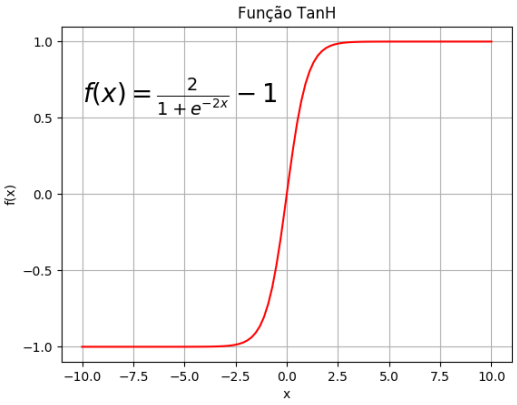
$$f(x) = leaky\_relu(x) = max(0.01x,\ x) \tag{2.6}$$

2.1.4 Batch Normalization

Nas redes neurais, a camada de *Batch Normalization* (BN) (IOFFE; SZEGEDY, 2015) é responsável pela normalização dos dados entre as camadas escondidas do modelo. Amostras que possuem características em diferentes escalas podem sofrer uma maior influência das *features* com maior escala no processo de treinamento. Desta forma, a normalização dos dados é importante para eliminar a influência negativa que características com uma escala grande podem exercer sob as que variam em uma escala pequena. Ela já era aplicada no *input* de modelos de aprendizagem de



(a) Sigmoid



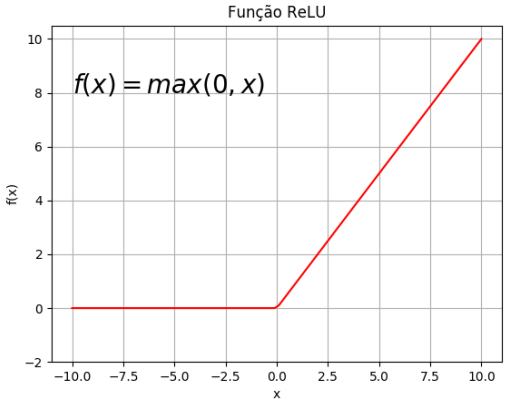
(b) TanH

Figura 10 – Plot das funções de ativação *Sigmoid* (a) e TanH (b).

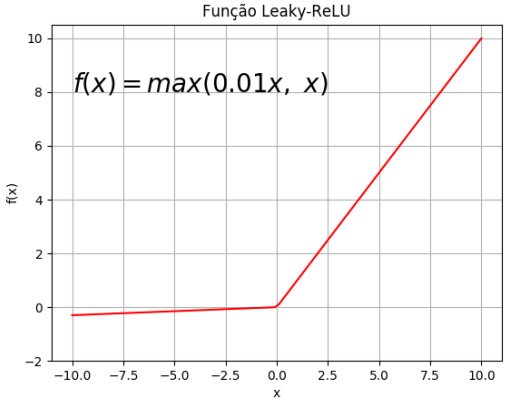
máquina há algum tempo, melhorando seus resultados e diminuindo o tempo de convergência dos modelos. Porém, com o surgimento do BN ela passou a ser aplicada nas camadas escondidas das redes neurais, e, de acordo com (IOFFE; SZEGEDY, 2015), o uso de BN conseguiu diminuir em até 14 vezes as épocas de treinamento dos modelos de *deep learning*, mantendo, assim, a mesma acurácia. Um *batch* é um subconjunto da base de dados.

A camada de BN geralmente é utilizada antes das camadas de ativação. Ela funciona normalizando as características de cada amostra pela média μ e o desvio padrão σ calculados entre as amostras do *batch*. As Equações 2.7 e 2.8 mostram como é calculado a média μ e a variância σ^2 das amostras no *batch*, respectivamente. Nela a variável m representa o número de amostras no *batch*, enquanto x_i é a i-ésima amostra do batch.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.7}$$



(a) ReLU



(b) Leaky-ReLU

Figura 11 – Plot das funções de ativação ReLU (a) e Leaky-ReLU (b).

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \tag{2.8}$$

Desta forma a amostra $\hat{x}_i$ normalizada é descrita pela Equação 2.9, na qual ϵ é uma constante adicionada a variância σ^2 para evitar divisões por zero.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.9}$$

Ainda, o BN adiciona os parâmetros β e γ que são aprendidos no processo de treinamento do modelo e ajudam a definir a influência que a normalização deve ter nos dados, podendo até ser utilizados para reverter o processo de normalização efetuada nos dados quando o modelo decidir, durante o processo de treinamento, que isso é o melhor para o problema. Sendo assim, considerando que a camada de BN faz um processo de transformação nos dados de $x_i \rightarrow y_i$, onde x_i é a amostra de entrada e y_i é a amostra de saída normalizada já com o uso dos parâmetros β e γ, a

Equação 2.10 mostra como os parâmetros β e γ se relacionam com $\hat{x}_i$.

$$y_i = \gamma \hat{x}_i + \beta \tag{2.10}$$

Sobre a normalização das amostras, os parâmetros μ e σ que representam respectivamente a média e o desvio padrão de cada *feature* dos *batches*, são calculados individualmente para cada um dos *batches* durante o processo de treinamento. Entretanto, na fase de teste do modelo nem sempre é utilizado um *batch* inteiro para classificação, frequentemente as amostras são classificadas uma por uma, o que torna impraticável o calculo de μ e σ. Para resolver esse problema, durante a fase de testes o BN estima os valores para μ e σ de acordo com a média dos valores obtidos durante o processo de treinamento. Além disso, o BN é suscetível ao tamanho do *batch* utilizado, uma vez que o tamanho do *batch* vai influenciar diretamente no μ e σ calculados, fazendo com que *batches* muito pequenos sejam mais suscetíveis a ruído.

2.1.5 *Layer Normalization*

O *Layer Normalization* (LN), proposto por (BA; KIROS; HINTON, 2016), é outro método de normalização que foi desenvolvido para lidar com os problemas do *Batch Normalization* (BN). Como explicado na seção anterior, o bom funcionamento do BN depende do tamanho do *batch* utilizado, uma vez que *batches* muito pequenos vão adicionar ruído nas estatísticas calculadas sobre eles. Como o BN utiliza a média e o desvio padrão calculado nos *batches* para normalizar as amostras, *batches* de tamanho 1 simplesmente não fazem sentido para ele, o que o torna impraticável para uso em problemas onde uma amostras é processada de cada vez. Desta forma, para resolver a limitação imposta no tamanho dos *batches*, o LN passa a calcular as métricas de média μ e desvio padrão σ sobre as características de cada amostra. Assim, ao contrário do BN que mantinha μ_k e σ_k para cada *feature* do *batch*, onde k é uma referência para a k-ésima característica, o LN calcula μ_i e σ_i para cada amostra do *batch*, sendo i uma referência para a i-ésima amostra do *batch*. Desta forma, como μ_i e σ_i são calculados de forma independente para cada amostra, o tamanho do *batch* não exerce influência sobre o *Layer Normalization* (LN).

$$\mu_i = \frac{1}{K} \sum_{k=1}^{K} x_{i,k} \tag{2.11}$$

A Equação 2.11 mostra como é calculada a média μ_i da i-ésima amostra x_i na camada de LN. Nela, K representa o número de características da amostra x_i, enquanto k é uma referência para a k-ésima caraterística da amostra x_i. Da mesma

forma, a Equação 2.12 mostra como a variância σ_i^2 é calculada numa camada de LN.

$$\sigma_i^2 = \frac{1}{K}\sum_{k=1}^{K}(x_{i,k} - \mu_i)^2 \tag{2.12}$$

A normalização das amostras x_i é dada pela Equação 2.13, na qual cada *feature* $x_{i,k}$ é subtraída da média da amostra μ_i e dividida pelo desvio padrão da amostra σ_i. No calculo do desvio padrão é adicionado uma constante ϵ para evitar divisões por zero. Cada característica da amostra normalizada é denotada por $\hat{x}_{i,k}$.

$$\hat{x}_{i,k} = \frac{x_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \tag{2.13}$$

Assim como na camada de BN, o processo de re-escala e *shift* das amostras, utilizando os parâmetros β e γ que são aprendidos durante o treinamento do modelo, também é feito na camada de LN. Esse processo é descrito pela Equação 2.10, onde y_i é a transformação de x_i através da camada de LN.

Pelas Equações 2.11 e 2.12 é possível ver que a média e o desvio padrão utilizados na camada de LN são calculadas individualmente para cada amostra, o que possibilita o uso de *batches* de qualquer tamanho sem prejudicar o desempenho do método. De fato, ao contrário da camada de *Batch Normalization* (BN), *batches* de tamanho 1 já não são um problema para a camada de *Layer Normalization* (LN), uma vez que as métricas calculadas para uma determinada amostra x não são influenciadas pelas demais amostras do *batch*.

2.1.6 Camada Totalmente Conectada

A camada totalmente conectada é a responsável por mapear as características extraídas nas camadas anteriores em diferente classes, o que é feito comumente com a ajuda da função de ativação *Softmax*. Ela normalmente é localizada no final da arquitetura da rede, na qual vai receber como *input* os dados gerados pelas camadas anteriores que são responsáveis pela extração de características e vai, com ajuda de alguma função de ativação, calcular a probabilidade dessas características pertencerem a cada uma das classes definidas no problema.

Essa camada trabalha fazendo uma combinação não linear dos dados com ajuda das Funções de Ativação (Subseção 2.1.3) e os vetores de pesos que vão atribuir uma importância adequada a cada característica do vetor de entrada. Tal importância atribuída a cada característica é medida pelo vetor de pesos que cada neurônio possui, e o vetor de pesos, por sua vez, aprende o valor adequado a atribuir a cada característica nova através do processo de treinamento da rede.

A Figura 12 apresenta uma representação de uma camada totalmente conectada. Os dados no vetor de *input* são oriundos das camadas anteriores do modelo, as quais são responsáveis pela extração das características. O fluxo dos dados na imagem ocorre da esquerda para direita, desta forma, o vetor de entrada propaga pelas camadas totalmente conectadas do modelo que vão aprender uma associação dessas características com suas devidas classes. Ao final do modelo, a rede gera como *output* as probabilidades atribuídas a cada classe.

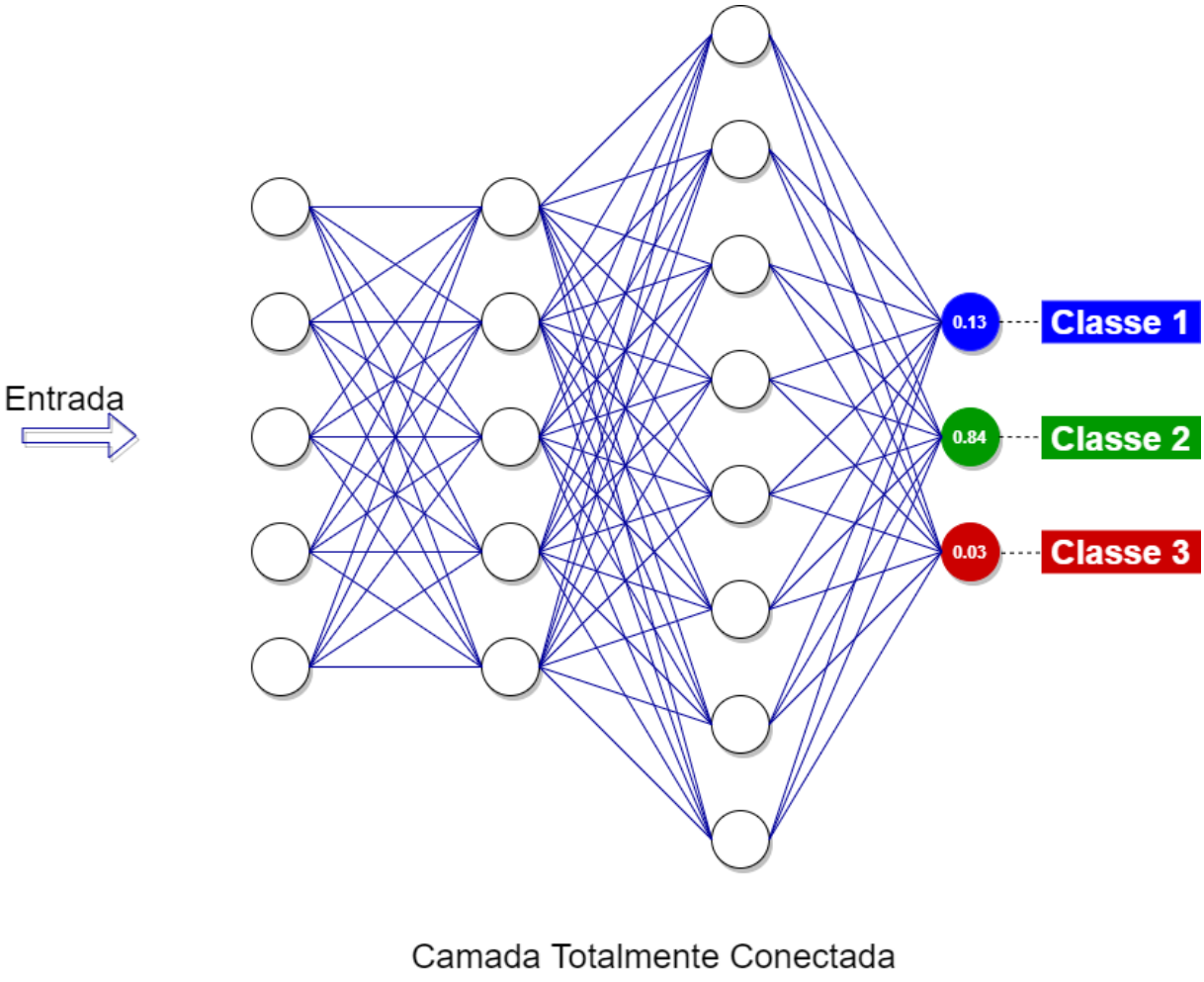


Figura 12 – Exemplo de camada totalmente conectada.

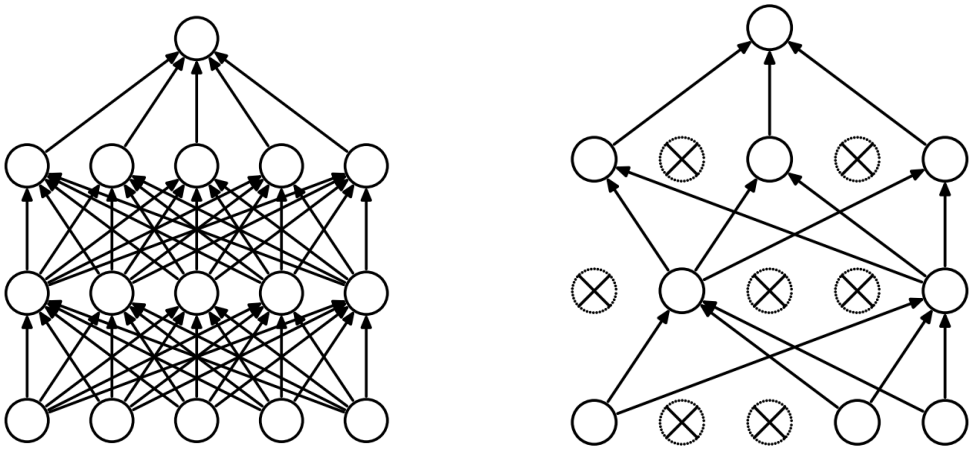
2.1.7 *Dropout*

A camada totalmente conectada das redes neurais tem uma alta capacidade de aprender padrões, o que pode se tornar algo negativo quando a base de dados de treinamento não é muito grande ou quando o treinamento é prolongado. Nessas ocasiões, o modelo se sobreajusta às amostras vistas no processo de treinamento, causando o que é chamado de *overfitting*. O problema do *overfitting* é que o modelo

fica enviesado na base de treinamento e acaba não generalizando bem os resultados para a base de testes. Uma técnica utilizada para reduzir os efeitos desse tipo de problema é o *dropout* (HINTON et al., 2012; TOMPSON et al., 2015). Ele é utilizado durante o processo de treinamento do modelo para introduzir ruído nas camadas totalmente conectadas e forçar a rede a aprender mais de uma rota para realizar a mesma tarefa.

O *dropout* funciona desativando alguns neurônios da camada totalmente conectada durante o processo de treinamento da rede. Os neurônios que serão desativados são escolhidos aleatoriamente e mudam de época para época. Desta forma, para desativar os neurônios é configurada uma probabilidade de ativação em cada neurônio, essa probabilidade é um parâmetro do *dropout* e pode variar de camada para camada. A Figura 13 mostra um exemplo da aplicação de *dropout* nas camadas totalmente conectadas de uma rede. Na Figura 13 (a), é ilustrada uma camada totalmente conectada padrão, na Figura 13 (b) é mostrado como essa camada totalmente conectada funciona ao utilizar o *dropout*. Após a aplicação do *dropout*, alguns neurônios são desligados aleatoriamente, forçando a rede a performar sua tarefa sem contar com a ajuda desses neurônios.

Durante o processo de treinamento, vários conjuntos diferentes de neurônios são desativados aleatoriamente, o que faz com que a rede como um todo aprenda vários caminhos diferentes para levar uma amostra x para uma resposta y, $x \rightarrow y$. De fato, essas várias soluções que o modelo aprende para resolver o mesmo problema, aumentam sua capacidade de generalização e diminuem as chances do modelo se sobreajuste a base de dados de treinamento, evitando o *overfitting*. O *dropout* só é utilizado durante o processo de treinamento das redes, para realizar o processamento das amostras de testes são utilizados todos os neurônios da camada totalmente conectada.



(a) Camada Totalmente Conectada. (b) Aplicando *Dropout*.

Figura 13 – Ilustração da aplicação do *dropout* em uma camada totalmente conectada. **Adaptado de (SRIVASTAVA et al., 2014)**.

2.2 PROCESSAMENTO DE SINAIS

Esta seção apresenta alguns métodos de processamento de sinais como as funções *sinc* e *Mel-Frequency Cepstrum Coefficients* (MFCC). Ambas podem ser utilizadas para o processamento de sinais de áudio, como pode ser visto em (HUSSAIN; HAQUE, 2018) e (Ravanelli; Bengio, 2018).

2.2.1 Funções *Sinc*

As funções *sinc* são bastante utilizadas na área de processamento de sinais. Em sua forma normalizada elas representam a transformada de Fourier (RABINER; SCHAFER, 2010) da função retangular sem escala. A função normalizada *sinc* é descrita pela Equação 2.14 para $x \neq 0$ e $sinc(x) = 1$ para $x = 0$. Além da função normalizada, existe a fórmula para função *sinc* não normalizada que pode ser vista na Equação 2.15 definida para $x \neq 0$, enquanto continua sendo definida de maneira similar a função normalizada para $x = 0$.

As funções *sinc* possuem algumas propriedades que podem ser visualizadas na imagem 14 e serão descritas a seguir. A função $sinc(x)$ é uma função par, o que significa que ela é simétrica no eixo y. Outra propriedade das funções *sinc* é que $sinc(x) = 0$ para todos os valores de x pertencentes ao conjunto dos números inteiros, $x \in Z$. Por último, outra propriedade importante das funções *sinc* é que elas são definidas como 1 para $x = 0$, ou seja, $sinc(0) = 1$.

$$sinc(x) = \frac{sin(\pi x)}{\pi x} \tag{2.14}$$

$$sinc(x) = \frac{sin(x)}{x} \tag{2.15}$$

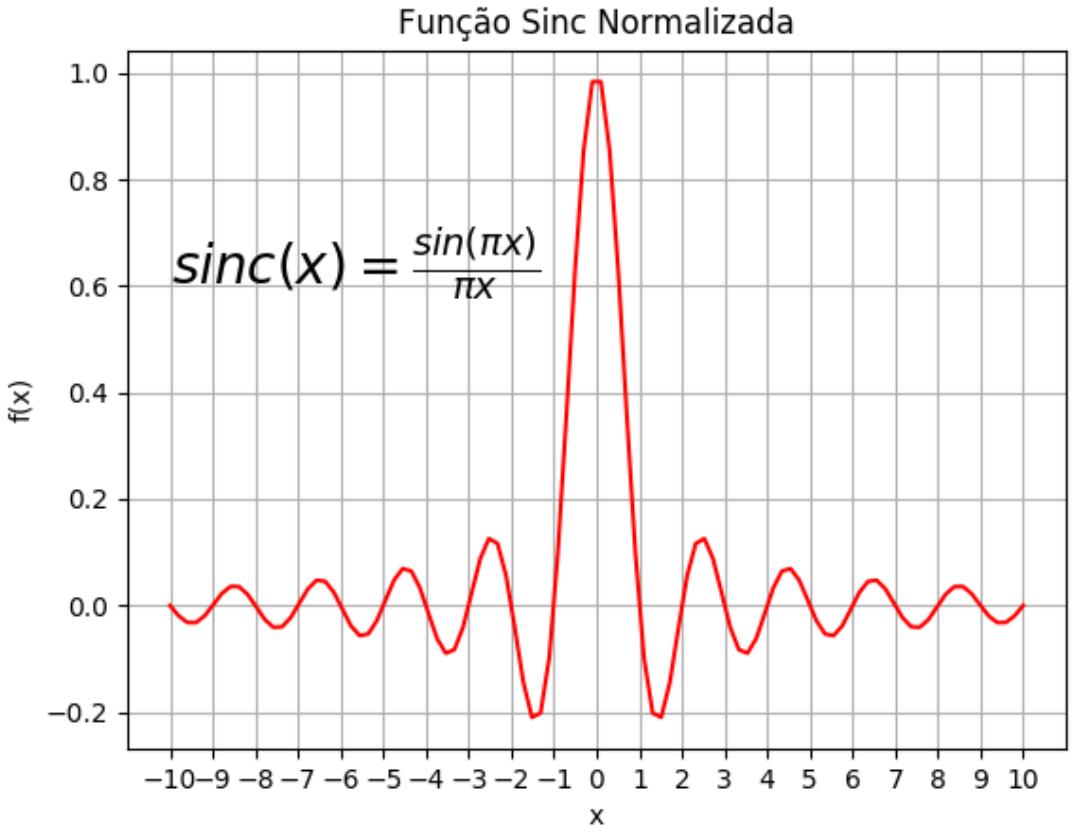


Figura 14 – Ilustração da função sinc normalizada.

2.2.2 *Mel-frequency cepstral coefficients (MFCC)*

O MFCC é um método de extração de características que utiliza coeficientes para representar as características dos sinais de áudio. Esses coeficientes são calculados a partir da resposta de filtros *mel* de forma triangular que são aplicados no sinal de áudio para capturar a quantidade de energia em determinada frequência na escala *mel*. Esse processo visa imitar a percepção das características identificadas pelo ouvido humano. De acordo com (Sadewa; Wirayuda; Sa'adah, 2015) o MFCC tenta imitar a forma em que humanos percebem as ondas de áudio.

As características extraídas através do MFCC são comumente utilizadas em sistemas modernos de reconhecimento a partir de sinais de áudio. O MFCC já foi utilizado como extrator de características em diversos trabalhos envolvendo problemas como: reconhecimento de locutor (Sadewa; Wirayuda; Sa'adah, 2015; Bansal; Imam; Bharti, 2015; DASH et al., 2012; Dehak et al., 2011), reconhecimento de discurso (Winur-

sito; Hidayat; Bejo, 2018; Vijayan et al., 2017) e segmentação de áudio (HUSSAIN; HAQUE, 2018).

2.3 CONSIDERAÇÕES FINAIS

Neste capítulo foram vistos os conceitos básicos sobre CNNs, abordando o funcionamento de suas camadas de convolução, *pooling* e funções de ativação, além das técnicas utilizadas para normalização das *features* como, *Batch Normalization* (BN) e *Layer Normalization* (LN), o funcionamento da camada totalmente conectada, responsável pela classificação das características extraídas e o uso de *dropout* para diminuir o efeito de *overfitting*. Ainda foram abordados alguns conceitos sobre processamento de sinais como funções *sinc* e o extrator de características *Mel-Frequency Cepstrum Coefficients* (MFCC).

3 TRABALHOS RELACIONADOS

Neste capítulo serão introduzidos os trabalhos que influenciaram os métodos propostos nessa dissertação, assim como uma revisão da literatura sobre o problema de Reconhecimento de Locutor. Primeiro, na Seção 3.1 serão apresentados alguns trabalhos sobre o tema de identificação e verificação de locutor. Na Seção 3.2 será descrito o *SincNet*, que servirá como base de comparação para os modelos propostos. Na Seção 3.3 será apresentado o *Additive Margin Softmax* (AM-Softmax), um importante componente no desenvolvimento do método proposto *Additive Margin SincNet* (AM-SincNet). A Seção 3.4 traz informações sobre a *MobileNet* que fez parte de alguns dos experimentos realizados neste trabalho. Por fim, a Seção 3.5 trará conceitos sobre a *SwishNet* e os *Gated Activation* (GA) que serviram de inspiração para criação do modelo proposto *Gated Activation Net*.

3.1 REVISÃO DA LITERATURA

Em (Muckenhirn; Magimai.-Doss; Marcell, 2018), foi desenvolvido um modelo para verificação de locutor que assim como o *SincNet* aprende as características diretamente do sinal de áudio, sem o uso de nenhuma técnica de processamento de sinais como *Mel-Frequency Cepstrum Coefficients* (MFCC) ou *Filter Bank* (FBANK). Neste trabalho eles primeiro treinaram uma *Convolutional Neural Networks* (CNN) para executar o trabalho de identificação de locutor, onde ela recebia como entrada a amostra de áudio e respondia com a probabilidade dessa amostra pertencer a cada um dos locutores. Por fim, em cima da arquitetura do sistema de identificação de locutor foi construído o sistema de verificação de locutor, onde eles substituíram a última camada da CNN por apenas duas saídas: locutor autêntico e impostor. O sistema também foi adaptado para receber como entrada duas amostras de áudio para que a comparação pudesse ser feita. Os experimentos foram realizados na base de dados *Voxforge* [1] e mostraram resultados competitivos com o estado da arte.

(Tang et al., 2017) propôs um novo *framework* multitarefa para os problemas de identificação de locutor e reconhecimento de discurso. Seu modelo é baseado em *Recurrent Neural Networks* (RNN) e, de acordo com o autor, se apoia no principio de que a mente humana aprender a interpretar a fala e identificar quem está falando de maneira colaborativa. Desta forma, (Tang et al., 2017) treina seu modelo de forma colaborativa, onde a saída de uma tarefa é propagada para a outra. De acordo com o autor, os experimentos realizados com seu novo *framework* indicaram

[1] <http://www.voxforge.org/>

que o processo de treinamento colaborativo conseguiu melhorar o desempenho em ambas as tarefas, identificação de locutor e reconhecimento de discurso, quando comparado a sistemas que trabalham com apenas uma tarefa por vez.

O uso de *data augmentation* foi aplicado em (Snyder et al., 2018) para melhorar o desempenho dos sistemas na tarefa de identificação de locutor. Como a geração de amostras com seus respectivos *labels* é relativamente custosa, o autor resolveu contornar esse problema utilizando *data augmentation* para aumentar o tamanho dos *datasets*. Neste trabalho, *Deep Neural Networks* (DNN) foram utilizadas para geração de *embeddings* a partir das amostras de áudio. Esses *embeddings* eram então classificados com técnicas como *Probabilistic Linear Discriminant Analysis* (PLDA). Nesse contexto, *data augmentation* foi aplicado nos *embeddings* gerados pela DNN de forma a aumentar a quantidade de amostras de cada classe na base de dados. O autor utilizou adição de ruído como operação para o processo de *data augmentation* e concluiu que o uso de técnicas de *data augmentation* é uma forma barata e rápida de aumentar as amostras da base de dados e que conseguiu diminuir as taxas de erros dos classificadores utilizados.

3.2 *SINCNET*

Proposto por (Ravanelli; Bengio, 2018), o *SincNet* é um modelo de rede neural artificial que unifica o poder das abordagens de *Deep Learning* para extração de características com a interpretabilidade das características extraídas através de padrões definidos por humanos. O *SincNet* é um modelo *end-to-end* para reconhecimento de locutor que usa *Deep Learning* e filtros *sinc* em sua composição. Nele, a primeira camada da rede neural que é responsável por uma convolução tradicional é substituída por uma convolução utilizando filtros *sinc* parametrizados.

As funções *sinc* são modeladas para o processamento de sinais digitais como sinais de áudio, desta forma, o uso delas como primeira camada convolucional ajuda a capturar as características mais significativas para a rede. As características extraídas pelas funções *sinc* são de mais fácil compreensão para humanos do que as características normalmente capturadas por uma camada de convolução tradicional, pois este tipo de função já é amplamente utilizado na área de processamento de sinais.

A nova camada de convolução utilizando filtros *sinc* parametrizados é responsável por varrer todo o sinal de áudio extraindo características básicas de baixo nível que serão mais tarde processadas pelas camadas mais profundas da rede neural. O uso das funções *sinc* ajuda a rede neural a aprender características mais relevantes do sinal de áudio. Ainda, o uso das funções *sinc* melhora o tempo de convergência

do modelo, uma vez que a camada de convolução com as funções *sinc* tem significantemente menos parâmetros do que uma camada de convolução normal. Cada função *sinc* de qualquer tamanho tem apenas dois parâmetros a serem aprendidos contra L parâmetros de uma convolução tradicional, na qual L é o tamanho do filtro utilizado na convolução. Outra vantagem das funções *sinc* é o fato delas serem simétricas, o que significa que o custo computacional para processar um filtro *sinc* pode ser reduzido pela metade apenas calculando metade do filtro e espelhando o resultado na outra parte do filtro.

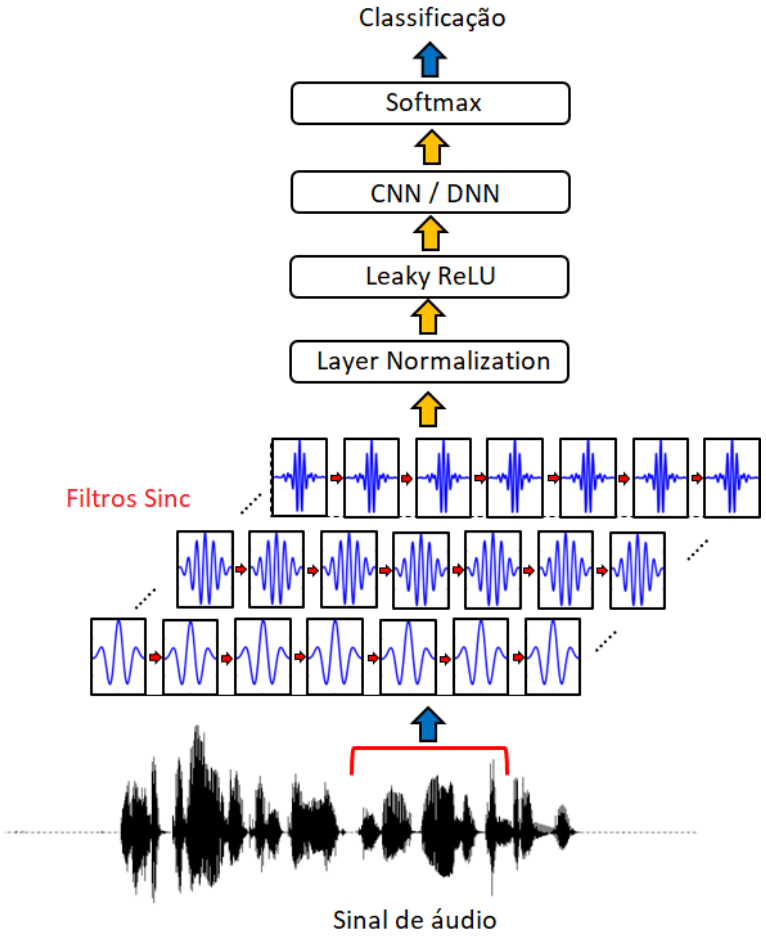


Figura 15 – Arquitetura da SincNet. **Adaptado de (Ravanelli; Bengio, 2018)**.

A primeira camada da rede executa convoluções baseadas em funções *sinc* utilizando 80 filtros de tamanho $L = 251$. O modelo então aplica mais duas camadas de convolução tradicional, cada uma utilizando 60 filtros com $L = 5$. *Layer Normalization* (LN) (BA; KIROS; HINTON, 2016) é aplicado tanto no *input* da rede quanto em todas as camadas de convolução. Depois das camadas de convolução, mais três camadas totalmente conectadas são aplicadas, cada uma com 2048 neurônios, a normalização dessas camadas é feita através de *Batch Normalization* (BN) (IOFFE; SZEGEDY, 2015). Todas as camadas escondidas da rede são não lineares e usam a

função de ativação *leaky-ReLU* (MAAS; HANNUN; NG, 2013). No topo do modelo, o *SincNet* usa uma camada *Softmax* que é responsável por mapear as características finais processadas pela rede em um espaço multidimensional correspondente às diferentes classes ou aos diferentes locutores.

A Figura 15 é uma ilustração da arquitetura da *SincNet*, na qual os sinais de áudio são processados primeiramente pelas camadas de convolução utilizando filtros *sinc*. Após o processamento inicial, as características extraídas são normalizados através de uma camada de *Layer Normalization* (LN) e passam por uma camada de ativação utilizando a função Leaky-ReLU. O resultado dessas operações é utilizado como *input* para camadas de convolução tradicional que vão alimentar uma *Deep Neural Networks* (DNN) com camadas totalmente conectadas. Por fim, as *features* processadas pelas camadas totalmente conectadas vão ser mapeadas nas diferentes classes da base de dados utilizando uma camada *Softmax*.

3.3 *ADDITIVE MARGIN SOFTMAX*

Em (WANG et al., 2018) foi proposto uma nova função de ativação para a área de reconhecimento facial, ela é baseada na tradicional *Softmax* e chamada de AM-Softmax. Diferente da *Softmax* tradicional que faz uma separação das classes por uma superfície linear, a AM-Softmax utiliza uma margem de separação aditiva entre as classes, que pode ser ajustada por um parâmetro configurável pelo projetista. Essa margem de separação se mostrou melhor do que a tradicional superfície linear da *Softmax* para o problema de reconhecimento facial, pois ela permite uma melhor separação entre amostras de classes distintas enquanto força as amostras da mesma classe a ficarem mais próximas uma das outras, melhorando assim tarefas relacionadas a classificação e verificação. Esses benefícios foram obtidos sem aumento considerável na complexidade de processamento, quando comparado com a *Softmax* tradicional. A equação da AM-Softmax é apresentada abaixo.

$$Loss = -\frac{1}{n}\sum_{i=1}^{n} log\frac{\phi_i}{\phi_i + \sum_{j=1,j\neq y_i}^{c} exp(s(W_j^T f_i))} \tag{3.1}$$

$$\phi_i = exp(s(W_{y_i}^T f_i - m)) \tag{3.2}$$

Nas Equações 3.1 e 3.2, W representa a matriz de pesos enquanto f_i é a entrada da i-ésima amostra na última camada totalmente conectada. A expressão $W_{y_i}^T f_i$ também é conhecida como *target logit* da i-ésima amostra. Os parâmetros s e m são responsáveis pela escala e pela margem aditiva, respectivamente. Embora o parâmetro responsável pela escala possa ser aprendido durante o processo de otimização da rede, isso pode deixar a convergência muito lenta. Uma alternativa para

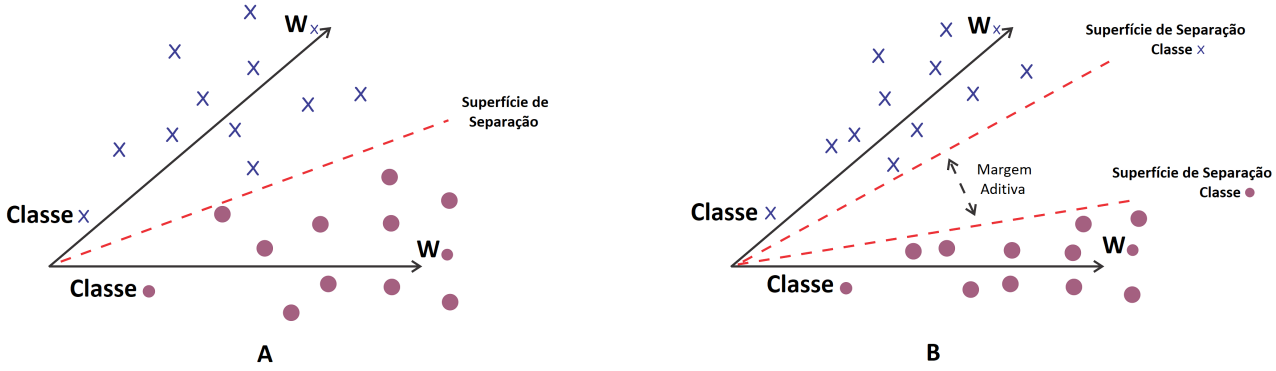


Figura 16 – Comparação entre Softmax (A) e AM-Softmax (B). **Adaptado de (Chagas Nunes; Macêdo; Zanchettin, 2019)**.

contornar esse problema é seguir (WANG et al., 2018) e setar o parâmetro s para um valor constante. Por outro lado, o parâmetro m responsável pela margem aditiva merece um estudo mais cauteloso para saber sua real influência sobre o problema em questão. A Figura 16 ilustra a diferença entre a *Softmax* tradicional (Figura 16 A) e a AM-Softmax (Figura 16 B). Nela é possível verificar como a AM-Softmax consegue separar melhor as amostras de diferentes classes enquanto minimiza a distância das amostras pertencentes a mesma classe. Isso é feito através do uso da margem de separação aditiva, que cria um espaço seguro entre as duas classes no qual nenhuma amostra deve permanecer. A *Softmax* tradicional, Figura 16 A, dispõe apenas de uma fronteira simples para dividir as duas classes, o que faz com que amostras muito próximas da fronteira possam ser facilmente confundidas como pertencentes a outra classe. Enquanto isso, na AM-Softmax, Figura 16 B, as amostras que se localizam próximo da fronteira de suas classes ainda permanecem a uma distância segura das outras classes, o que dificulta os possíveis erros de classificação nessas amostras.

3.4 *MOBILENET*

A *MobileNet* (HOWARD et al., 2017; Sandler et al., 2018) é um modelo de rede neural desenvolvido para se adaptar melhor a dispositivos móveis e embarcados nos quais o poder computacional é reduzido e a memória disponível limitada. Nos últimos anos houve um notável interesse em construir modelos de redes neurais para dispositivos móveis e embarcados, como pode ser visto em (JIN; DUNDAR; CULURCIELLO, 2014; Wang; Liu; Foroosh, 2017; IANDOLA et al., 2016; WU et al., 2016; RASTEGARI et al., 2016). Entretanto, na maioria desses trabalhos o foco é apenas diminuir o tamanho da rede para que se possa utilizar os modelos em dispositivos com recursos limi-

tados de memória, sem tentar diminuir a latência na execução destes (em alguns casos isso pode ocorrer como consequência da redução do tamanho do modelo). Com isso em vista, o foco principal da *MobileNet* é diminuir o tempo de execução da rede utilizando estratégias que acabam, por consequência, reduzindo o tamanho do modelo.

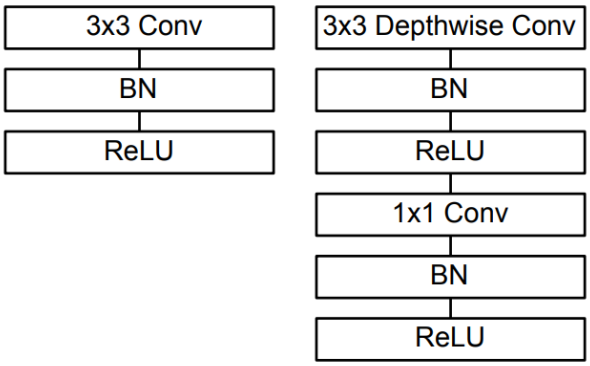
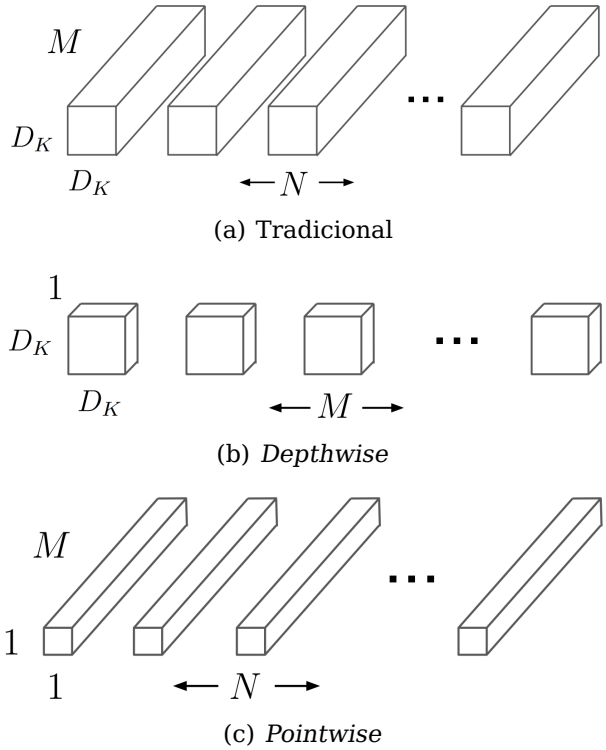


Figura 17 – Comparação entre uma camada de convolução normal e uma camada de convolução *depthwise*. **Fonte: (HOWARD et al., 2017)**.

A base da arquitetura da *MobileNet* é construída a partir de *depthwise separable convolutions* que foram desenvolvidos para substituir as camadas tradicionais



(a) Tradicional

(b) *Depthwise*

(c) *Pointwise*

Figura 18 – Comparação entre uma convolução tradicional e uma convolução *depthwise*. Tradicional (a), *Depthwise* (b) e *Pointwise* (c). **Adaptado de (HOWARD et al., 2017)**.

de convolução. A *depthwise separable convolution* é construída a partir de duas outras camadas: a *depthwise convolution* e a *pointwise convolution*. Essas duas camadas funcionam dividindo o trabalho de convolução em dois. A *depthwise convolution* aplica um único filtro de convolução em cada canal da amostra de entrada, enquanto a *pointwise convolution* funciona aplicando um filtro de tamanho 1×1 na saída da *depthwise convolution* para fazer uma combinação entre seus valores de saída. A *MobileNet* usa *depthwise separable convolutions* de tamanho 3×3 o que reduz o custo computacional de 8 a 9 vezes quando comparado a camadas de convolução tradicional (HOWARD et al., 2017).

Na Figura 17 pode-se observar a estrutura de uma camada de convolução tradicional e uma *depthwise*. A coluna da esquerda representa uma camada de convolução tradicional, na qual uma convolução é seguida por uma camada de BN e em seguida por uma camada ReLU (NAIR; HINTON, 2010). Já a coluna da direta representa uma camada de convolução *depthwise*, na qual a convolução *depthwise* é seguida por uma camada de *Batch Normalization* (BN), ReLU e então é passada uma convolução de tamanho 1×1 (*pointwise*) seguida também por BN e ReLU.

A Figura 18 mostra a diferença entre como é operado uma convolução tradicional e uma convolução *depthwise*. Na Figura 18 (a) é ilustrado uma representação de N filtros de convolução tradicional com *kernel* de tamanho $D_K \times D_K$ e profundidade M. Na convolução tradicional cada filtro de convolução, independente do tamanho do *kernel*, tem profundidade M, onde M é o número de canais da amostra de entrada. Desta forma, cada filtro de convolução opera em todos os canais da amostra de entrada. O número de canais resultante de uma operação de convolução tradicional é controlado através do número de filtros utilizado, sendo assim, na Figura 18 (a), a amostra de entrada tem M canais de profundidade e seu resultado terá N canais de profundidade na saída. Por outro lado, na *MobileNet* o processo de convolução é dividido em duas etapas: Convolução *Depthwise* Figura 18 (b) e *Pointwise* Figura 18 (c). Na convolução *Depthwise* os filtros de convolução tem dimensão $D_K \times D_K \times 1$, onde $D_K \times D_K$ é o tamanho do *kernel* utilizado e 1 é a profundidade do filtro de convolução. Nela, cada filtro de convolução opera em apenas 1 canal da amostra de entrada, dessa forma são necessários M filtros de convolução *depthwise* para cobrir todos os canais da amostra de entrada e cada filtro de convolução *depthwise* fica especializado em apenas um canal da amostra de entrada. Depois da convolução com filtros *depthwise*, são aplicadas convoluções com filtros *Pointwise*. Na Figura 18 (c) é possível ver como a convolução com filtros *Pointwise* funciona, nela cada filtro tem dimensão $1 \times 1 \times M$, onde 1×1 é o tamanho do *kernel* de convolução e M é a profundidade do filtro. Na convolução *Pointwise*

cada filtro de convolução tem profundidade M, onde M é o número de canais da amostra de entrada, nesse caso o filtro de convolução *Pointwise* funciona como uma combinação linear do resultado dos filtros *Depthwise*, nele o número de canais da amostra de saída é controlado pelo número de filtros *Pointwise* utilizado, na Figura 18 (c) o número de filtros *Pointwise* utilizados é N.

3.5 *SWISHNET*

Proposta por (HUSSAIN; HAQUE, 2018), *SwishNet* é uma CNN 1D para o processamento de áudio que utiliza como entrada as características extraídas por MFCC (LOGAN, 2000). O uso de convoluções 1D faz com que a *SwishNet* seja leve e consequentemente mais rápida do que CNNs 2D tradicionais. O consumo reduzido de memória também faz com que a *SwishNet* seja uma opção para dispositivos móveis e embarcados. Avaliado nas bases de dados *MUSAN* (SNYDER; CHEN; POVEY, 2015) e *GTZAN* (Tzanetakis; Cook, 2002), a *SwishNet* mostrou bons resultados nas tarefas de classificação e segmentação.

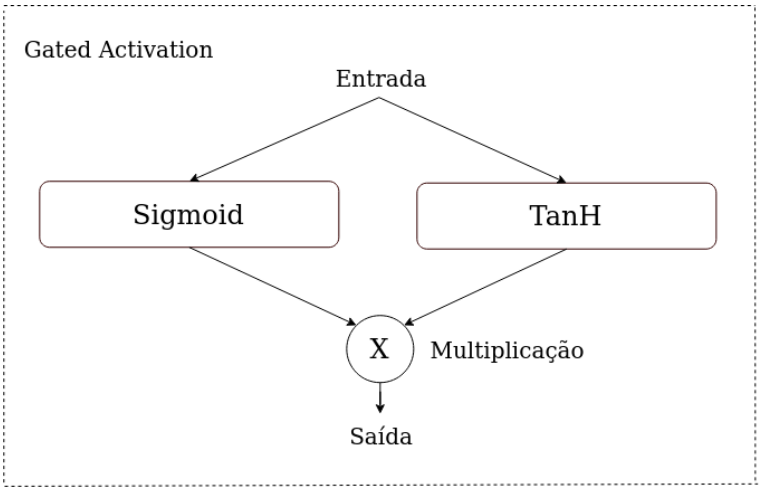


Figura 19 – Função de Gate Activation.

Inspirada na *WaveNet* (OORD et al., 2016a) e na *Inception* (SZEGEDY et al., 2014), a arquitetura da *SwishNet* usa múltiplas camadas de convolução em conjunto com funções de *Gated Activation* (GA) (OORD et al., 2016b) que substituem a tradicional ReLU pela *Sigmoid* e *TanH* e selecionam as características que vão propagar de uma camada da rede para outra. Ainda, a *SwishNet* faz uso de *residual* e *skip connections* que são aplicadas para melhorar a acurácia da rede e facilitar o treinamento. A Figura 19 mostra como o GA funciona. Nela os dados de entrada propagam para uma camada *Sigmoid* e uma camada *TanH* paralelamente. Após o processamento em ambas as camadas de ativação, os resultados são unidos novamente através da multiplicação do *output* de cada camada de ativação, para gerar

a saída da função. A função de ativação GA é utilizada para substituir a ReLU na *SwishNet*, desta forma, ela é localizada sempre após uma camada de convolução.

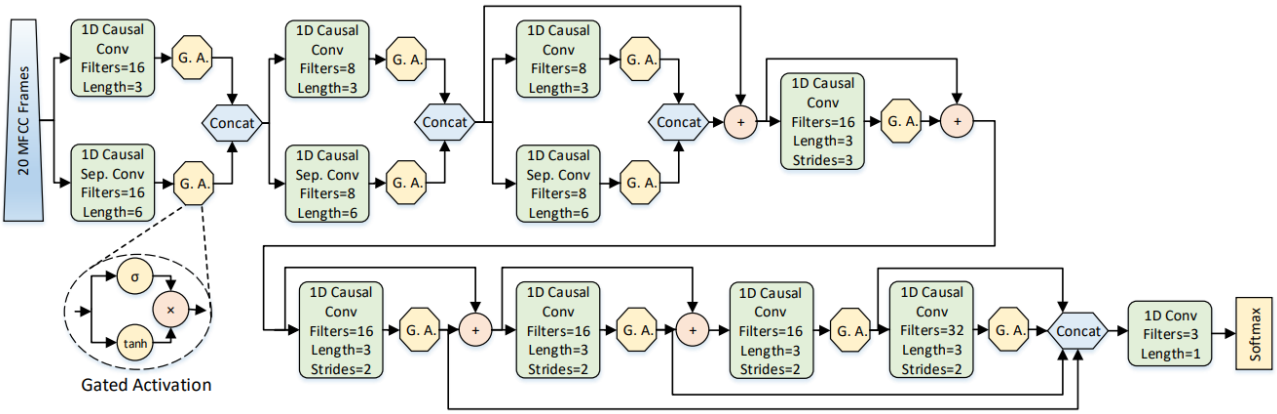


Figura 20 – Arquitetura da *SwishNet* na sua forma compacta. **Fonte: (HUSSAIN; HAQUE, 2018)**.

A Figura 20 mostra a arquitetura da *SwishNet*. Nela, as amostras de entrada são características extraídas dos sinais de áudio utilizando *Mel-Frequency Cepstrum Coefficients* (MFCC). Os dados de entrada passam por duas camadas de convolução 1D em paralelo, cada uma com 16 filtros de convolução, porém com *kernel* de tamanho diferente, tamanho 6 e 3, respectivamente. Após cada camada de convolução é utilizada uma camada de *Gated Activation* (GA), o resultado das convoluções e do GA é então concatenado para passar pra próxima camada da rede. A próxima camada tem a mesma configuração da primeira, entretanto utiliza um número de filtros de convolução reduzido, a partir dela todas as camadas de convolução subsequentes utilizam *skip connections* para ajudar no treinamento da rede. A partir da quarta camada da rede as convoluções param de ser feitas em paralelo com o tamanho do *kernel* variado e passam a ser realizadas de forma tradicional utilizando $kernel = 3$. A *SwishNet* tem ao todo 9 camadas de convolução mais uma camada *Softmax* que é responsável por mapear as características extraídas entre as classes da base de dados.

3.6 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada uma revisão da literatura sobre o problema de reconhecimento de locutor. A arquitetura do modelo base *SincNet*, como as funções *sinc* são utilizadas e o motivo delas serem interessantes em aplicações de reconhecimento de locutor. Também foi mostrado o funcionamento da função *Additive Margin Softmax* (AM-Softmax) e suas vantagens sobre a *Softmax* tradicional. Os modelos *MobileNet* e *SwishNet* também foram explorados neste capítulo, com foco

no motivo que os tornam modelos interessantes para aplicações em ambientes embarcados. No modelo *SwishNet* foi destacada a função de ativação *Gated Activation* (GA) que servirá como base do modelo proposto *Gated Activation Net* (Seção 4.4).

4 MODELOS PROPOSTOS

Neste capítulo serão apresentados os modelos propostos neste trabalho. A Seção 4.1 descreverá o *Additive Margin SincNet* (AM-SincNet), que é uma adaptação do *SincNet* com a versão melhorada da *Softmax*, a AM-Softmax. A Seção 4.2 descreverá os métodos propostos *MobileNet1D* e AM-MobileNet1D, que são a versão proposta do *MobileNet V2* para o problema de reconhecimento de locutor. Na Seção 4.3 é apresentado o *Sinc MobileNet* que é a versão do *MobileNet1D* proposta aqui em conjunto com as camadas *sinc* do *SincNet*. Por fim, na Seção 4.4 é descrita a arquitetura do modelo proposto *Gated Activation Net* que foi inspirado por operações de *Gated Activation* (GA) encontradas na *SwishNet* e como ele pode ser utilizado em conjunto com as camadas *sinc*.

4.1 *ADDITIVE MARGIN SINCNET*

O *Additive Margin SincNet* (AM-SincNet) (Chagas Nunes; Macêdo; Zanchettin, 2019) é um modelo de rede neural para reconhecimento de locutor baseado na arquitetura da *SincNet* (Ravanelli; Bengio, 2018) (Seção 3.2) e na função de ativação AM-Softmax (WANG et al., 2018) (Seção 3.3).

Como descrito na Seção 3.2, a *SincNet* é um modelo de rede neural para reconhecimento de locutor que utiliza funções *sinc* (Subseção 2.2.1) como filtros para sua primeira camada de convolução. O uso das funções *sinc* na primeira camada de convolução da *SincNet* reduz drasticamente o número de parâmetros a serem aprendidos durante o processo de treinamento da rede, permitindo ao modelo convergir mais rápido e aprender características de mais fácil compreensão por humanos, uma vez que as funções *sinc* extraem características mais interpretáveis para um especialista humano. A *SincNet* já mostrou bons resultados para o problema de reconhecimento de locutor, sua arquitetura foi comparada com modelos tradicionais de CNN e alguns outros modelos que se utilizam de características extraídas através de FBANK e MFCC e em todos os cenários a *SincNet* mostrou melhores resultados. A maior contribuição da *SincNet* foi no uso de funções *sinc* como filtros em sua primeira camada de convolução, entretanto, a *SincNet* utiliza a função *Softmax* para calcular a probabilidade a posteriori dos dados de entrada pertencerem a cada uma das classes de locutores. Apesar da função *Softmax* ser uma escolha comum para determinada tarefa, ela não é particularmente capaz de produzir uma distinção clara entre cada uma das classes na camada final da rede. Portanto, a AM-SincNet faz a alteração da última camada da *SincNet* responsável

por calcular as probabilidades a posteriori, trocando assim, a função *Softmax* pela função melhorada AM-Softmax.

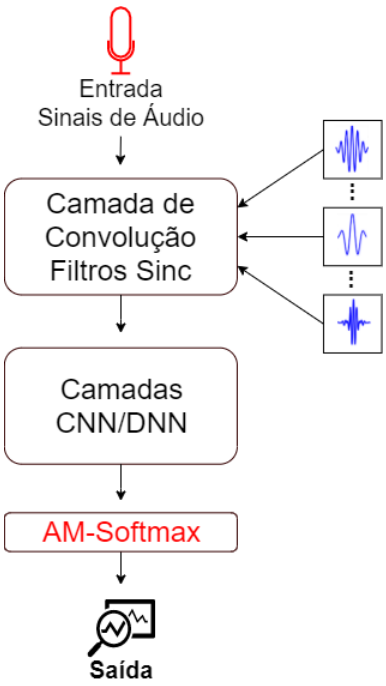


Figura 21 – Arquitetura da AM-SincNet. **Adaptado de (Chagas Nunes; Macêdo; Zanchettin, 2019)**.

A AM-Softmax é uma função de ativação derivada da tradicional *Softmax* que introduz uma margem aditiva em vez de uma superfície linear para realizar a separação das classes. A margem aditiva (Equações 3.1 e 3.2) funciona como um melhor separador das classes do que a tradicional superfície linear de separação da *Softmax*. Isso se explica porque a margem de separação aditiva acaba forçando as amostras de classes distintas a ficarem mais separadas umas das outras, ao mesmo tempo que a distância entre amostras pertencentes a mesma classes é reduzida, como pode ser visto na Figura 16. A AM-Softmax foi inicialmente proposta para o problema de reconhecimento facial, contudo, em (Chagas Nunes; Macêdo; Zanchettin, 2019) (artigo derivado desta dissertação) foi visto que sua eficácia também se estende para o problema reconhecimento de locutor. A Figura 21 apresenta uma ilustração da arquitetura do método proposto AM-SincNet, na qual os sinais de áudio são primeiro processados por uma camada de convolução com filtros *sinc*, que servem para extrair características relevantes da amostra de áudio. As características extraídas pelos filtros *sinc* passam então pra uma camada de convolução tradicional e depois para uma DNN com camadas totalmente conectadas. Por fim, as *features* resultantes são mapeadas entre as classes da base de dados através da função melhorada AM-Softmax.

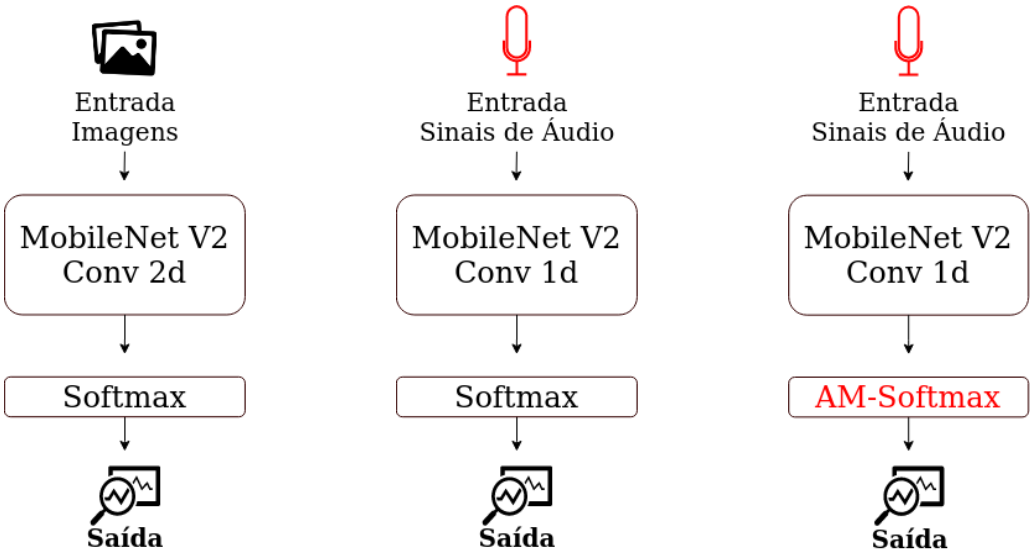
O método proposto AM-SincNet combina o poder de extração de características dos filtros *sinc* com a capacidade melhorada de distinção das classes da AM-Softmax. Isso é feito sem prejuízo na latência do modelo, uma vez que a substituição da camada *Softmax* pela camada AM-Softmax não aumenta a complexidade do modelo.

4.2 *AM-MOBILENET1D*

Como foi visto na Seção 3.4, a *MobileNet* proposta por (HOWARD et al., 2017) é uma rede neural convolucional construída para ser leve e rápida, o que a torna interessante para trabalhar com dados de alta dimensionalidade como os sinais de áudio utilizados aqui, que tem 16.000 características por segundo de gravação. Entretanto, a arquitetura tradicional da *MobileNet* foi modelada para trabalhar com imagens e as imagens geralmente são representadas por matrizes tridimensionais nas quais se tem uma dimensão para altura da imagem, outra dimensão para largura e por fim uma dimensão para os canais de cores da imagem (também conhecidos como volumes). Diferente das imagens, os sinais de áudio são geralmente representados por matrizes bidimensionais com uma dimensão para o tempo e outra para o canal do áudio. Nas operações de convolução, os canais dos dados, como os canais de cores da imagem e os canais do áudio, são considerados como canais da convolução e já são automaticamente incluídos na operação. Desta forma, convoluções para operar com dados como imagem são chamadas de convolução 2D, enquanto convoluções para trabalhar com dados iguais aos sinais de áudio são chamadas de convolução 1D. Como o *MobileNet* é formado apenas de convoluções 2D, foi necessário sua adaptação para lidar com sinais de áudio.

Neste experimento foi utilizada a *MobileNetV2* (Sandler et al., 2018), nela foi feita a adaptação de todas as operações de convolução, *Batch Normalization* (BN) e *pooling* destinadas a trabalhar com dados bidimensionais por operações equivalentes próprias para lidar com dados de apenas uma dimensão (sem considerar os canais). Essa mudança foi inspirada pela *SwishNet* (HUSSAIN; HAQUE, 2018) que utilizou convoluções 1D nos sinais de áudio processados pela MFCC para reduzir a complexidade da rede a tornando mais leve e rápida. Deste modo, a versão da *MobileNet* proposta nesse trabalho é uma versão com menos parâmetros que se torna mais leve e rápida destinada a processar sinais com uma dimensionalidade menor que as imagens o que é o caso dos sinais de áudio.

A Figura 22 apresenta uma abstração entre o modelo da *MobileNetV2* proposto por (Sandler et al., 2018) e os modelos propostos neste trabalho neste contexto para o problema de reconhecimento de locutor. Na Figura 22 (a) representa o modelo



(a) MobileNetV2 (b) MobileNet1D (c) AM-MobileNet1D

Figura 22 – Comparação entre a *MobileNet* tradicional (a) e os modelos propostos: *MobileNet1D* (b) e AM-MobileNet1D (c).

tradicional da *MobileNetV2* e na Figura 22 (b) apresenta o modelo proposto *MobileNet1D*. Como comentado anteriormente, a diferença entre os dois modelos está nas operações relacionadas aos dados bidimensionais no modelo tradicional que foram modificadas para trabalharem com dados de apenas uma dimensão nos modelos propostos. Na Figura 22 (a) pode ser visto que o modelo está nomeado com uma *Conv 2d*, isso ocorre porque a convolução nesse caso está preparada para receber imagens como entrada, sendo assim, cada convolução vai operar nas dimensões de altura e largura da imagem, tratando os canais de cores como canais de convolução. Da mesma forma, nas Figuras 22 (b) e (c) os modelos são representados com uma *Conv 1d*, isso ocorre pelo fato de que os sinais de áudio tem uma dimensão a menos que as imagens e da mesma forma a convolução trata os canais do áudio como canais de convolução. Em ambas as subimagens o fluxo dos dados é o mesmo. Primeiro os dados de entrada são processados pela arquitetura da *MobileNetV2* utilizando *Conv 2d* para *inputs* do tipo imagem e *Conv 1d* para amostras de uma dimensão, como os sinais de áudio. Por fim, as características extraídas são mapeadas nas diferentes classes através de uma camada *Softmax* ou AM-Softmax.

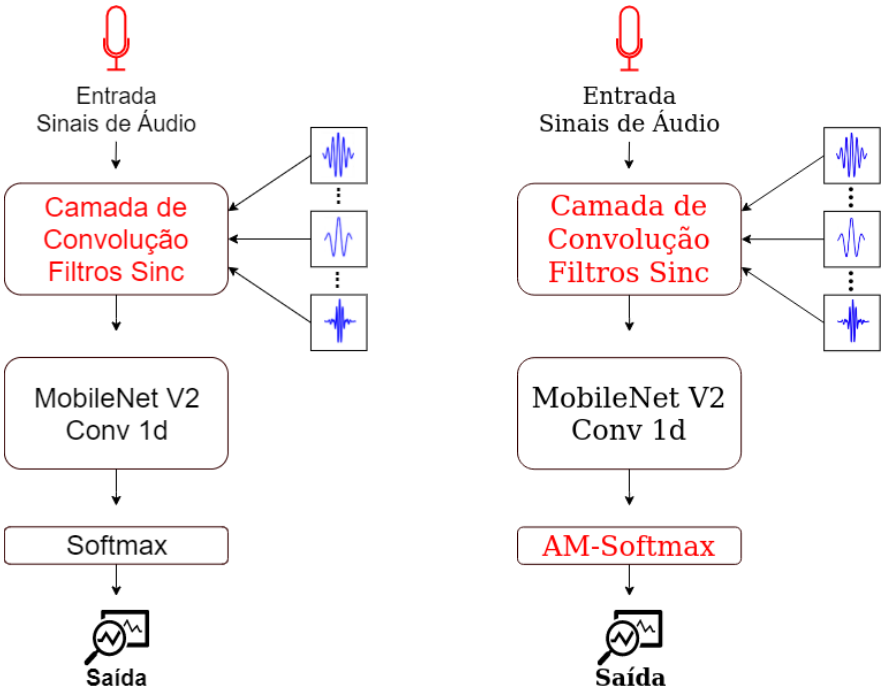
O modelo proposto, *MobileNet1D* também pode ser adaptado para utilizar a função melhorada AM-Softmax. Essa adaptação pode ser feita mudando apenas a última camada da *MobileNet1D*, substituindo a camada da *Softmax* pela camada da AM-Softmax, como pode ser visto na Figura 22 (c). O método feito da junção da *MobileNet1D* com a função melhorada AM-Softmax, recebe o nome de AM-MobileNet1D. Assim como no AM-SincNet a substituição da última camada do

MobileNet1D pela AM-Softmax não aumenta a complexidade do modelo, apenas melhora sua capacidade de classificação das amostras.

4.3 *SINC AM-MOBILENET1D*

O *Sinc MobileNet* segue o mesmo princípio dos métodos propostos na seção anterior, o *MobileNet1D* e o AM-MobileNet1D (Seção 4.2), que é a adaptação da *MobileNet* para trabalhar com sinais de áudio. Entretanto, na *Sinc MobileNet1D* é adicionado um módulo a mais para extração de características, esse módulo é responsável pela primeira transformação dos dados e fica localizado na primeira camada da arquitetura lidando diretamente com os sinais de áudio. O módulo adicionado é o módulo de convolução com funções *sinc* da *SincNet* e objetiva extrair características mais relevantes e interpretáveis dos sinais de áudio sem um aumento considerável na complexidade do modelo, pois as funções *sinc* introduzem poucos parâmetros a serem aprendidos pela rede.

Assim como o *MobileNet1D*, o *Sinc MobileNet1D* é também um modelo leve e rápido quando comparado ao modelo tradicional da *MobileNet*, isso é dado pela substituição das operações destinadas às imagens por operações mais adequadas a sinais de áudio, reduzindo assim o número de parâmetros a serem aprendidos pelo modelo. A camada de convolução com funções *sinc* também promete extrair carac-



(a) *Sinc MobileNet1D* (b) *Sinc AM-MobileNet1D*

Figura 23 – Arquitetura da *Sinc MobileNet1D* e *Sinc AM-MobileNet1D*.

terísticas mais relevantes para o problema do reconhecimento de locutor, uma vez que as funções *sinc* são funções designadas para tratamento de sinais, principalmente sinais de áudio.

A Figura 23 (a) apresenta a arquitetura do modelo proposto *Sinc MobileNet1D*, nela é possível ver que os sinais de áudio são primeiro processados pela camada de convolução *sinc* com o intuito de extrair características relevantes dos sinais de áudio. Posteriormente, as características extraídas pela camada de convolução *sinc* são passadas para o modulo de processamento da *MobileNet* com as operações adaptadas para processar sinais com a mesma dimensão dos sinais de áudio. Por fim, o modelo proposto faz o mapeamento das características processadas para cada uma das classes na base de dados, esse processamento é feito pela função *Softmax* que vai liberar como saída a probabilidade dessas características pertencerem a cada uma das classes do *dataset*.

Assim como no modelo proposto AM-MobileNet1D, o *Sinc MobileNet1D* também pode ser adaptado para uma versão utilizando uma camada AM-Softmax no lugar da camada *Softmax*. Essa versão é chamada de *Sinc AM-MobileNet1D* e pode ser vista na Figura 23 (b), o fluxo de dados na *Sinc AM-MobileNet1D* é o mesmo da *Sinc MobileNet1D* e a substituição da camada *Softmax* pela camada com a AM-Softmax não adiciona complexidade extra ao modelo.

4.4 *GATED ACTIVATION NET*

A *Gated Activation Net* é um modelo de rede neural para processamento de áudio proposto neste trabalho com o proposito de classificar sinais de áudio em diferentes classes de locutores, fazendo assim a tarefa de reconhecimento de locutor. O modelo proposto foi baseado principalmente nos módulos de *Gated Activation* (GA) da *Wavenet* (OORD et al., 2016a) que também são utilizados na *SwishNet* (HUSSAIN; HAQUE, 2018) e nas convoluções com diferentes tamanhos de *kernel* utilizados na *SwishNet*. Seguindo o mesmo princípio dos modelos propostos nas Seções 4.2 e 4.3, a *Gated Activation Net* utiliza apenas convoluções 1d, próprias para lidar com a dimensão dos sinais de áudio, o que torna a rede mais leve e rápida quando comparada a uma CNN tradicional com camadas de convolução 2d. Isso é dado pela diminuição no número de parâmetros a serem aprendidos pelo modelo nas camadas de convolução, como explicado nas seções anteriores.

O modelo proposto é construído a partir de alguns módulos base como é possível ver na Figura 24 e serão descritos em detalhes nas subseções a seguir. O modelo recebe como entrada características extraídas a partir dos sinais de áudio, essas características podem ser o resultado do processamento desses sinais por

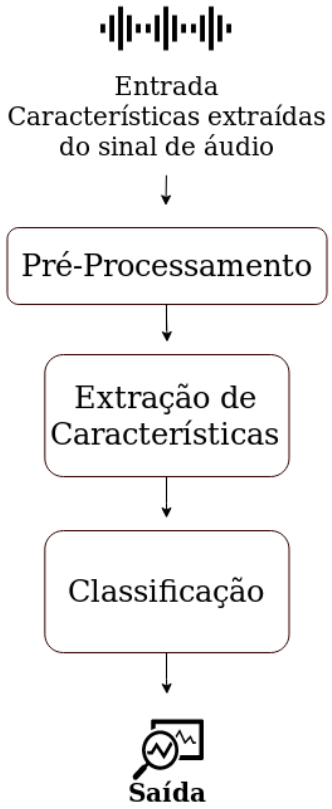


Figura 24 – Arquitetura da Gated Activation Net.

uma camada de convolução *sinc*, como é feito na *Sinc MobileNet* (Seção 4.3), ou até mesmo os sinais de áudio sem nenhum processamento prévio.

4.4.1 Pré-Processamento dos dados

O módulo de pré-processamento dos dados tem a função de preparar os dados para as próximas camadas do modelo, normalizando as características de entrada e eliminando as que não tiveram altas respostas nos filtros de convolução. A Figura 25 apresenta o fluxo de dados neste módulo, no qual os dados de entrada são normalizados através de uma camada de *Batch Normalization* (BN) e depois a dimensionalidade das características é reduzida por uma camada de *max pooling*.

4.4.2 Extração de Características

A etapa de extração de características no modelo proposto *Gated Activation Net* é uma das etapas mais importantes no fluxo de dados neste modelo, ela é composta por quatro camadas do módulo *BaseLayer* ligadas em série, na qual a saída de um módulo é a entrada de outro. Essa ligação permite que a rede aprenda características mais abstratas nas camadas mais elevadas da arquitetura. A Figura 26 ilustra como esse módulo é montado, no qual os dados de entrada passam por qua-

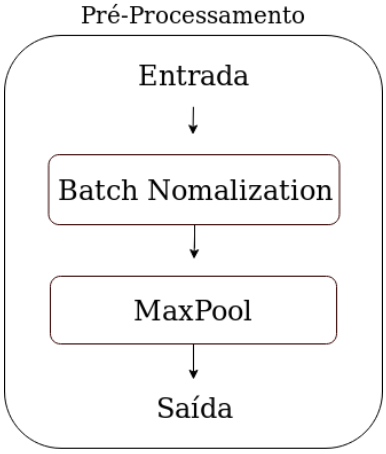


Figura 25 – Pré-processamento dos dados no modelo proposto *Gated Activation Net*.

tro camadas do módulo *BaseLayer*. A primeira camada possui um número total de 40 filtros de convolução, enquanto a segunda camada possui 80, a terceira camada tem 160 filtros e, por fim, a última camada do módulo *BaseLayer* tem apenas 80 filtros de convolução 1d.

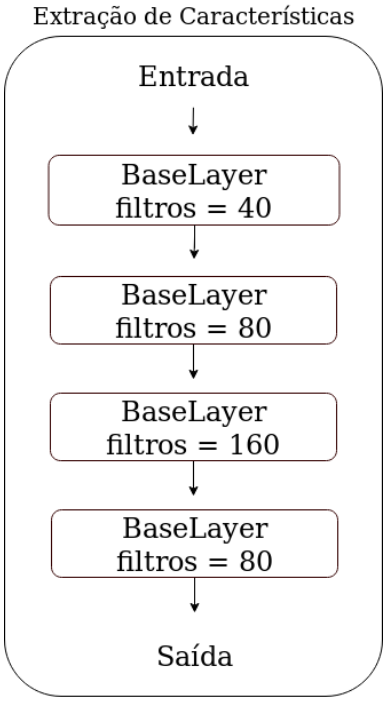


Figura 26 – Extração de Características no modelo proposto *Gated Activation Net*.

O módulo *BaseLayer* que compõe a etapa de extração de características (Figura 26) recebe como parâmetro o número de filtros de convolução a serem realizadas em cada instância do módulo. Ele por sua vez é inspirado na arquitetura da *SwishNet* que realiza convoluções com filtros de tamanho de *kernel* diferentes, isso possibilita a rede a aprender padrões em diferentes escalas de tempo. A Fi-

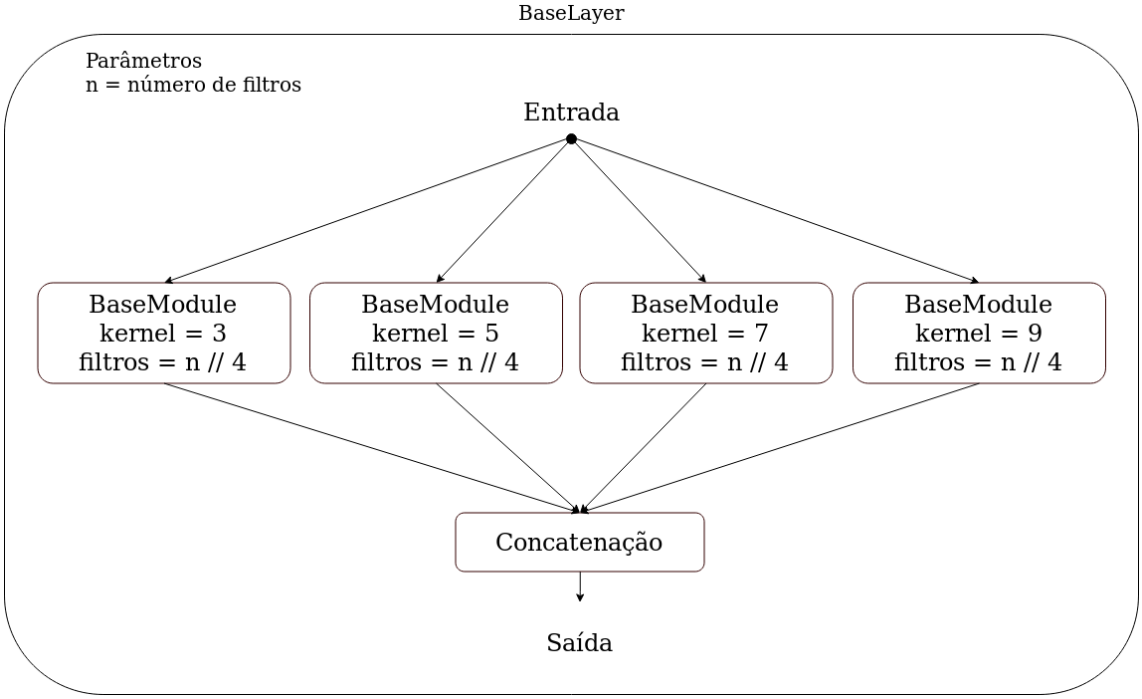


Figura 27 – Módulo *BaseLayer* no modelo proposto *Gated Activation Net*.

gura 27 mostra como é organizado o módulo *BaseLayer*, nele os dados de entrada são propagados para quatro instâncias do módulo *BaseModule* que estão ligadas em paralelo, desta forma, cada instância recebe os mesmos dados de entrada do módulo e vão aprender padrões nesses dados a partir de convoluções com janelas de *kernel* diferentes. Cada uma das instâncias do *BaseModule* ligadas em paralelo computam o mesmo número de filtros de convolução $int(\frac{n}{4})$ que é o número total de filtros recebido como parâmetro pelo módulo *BaseLayer*, n, dividido pelo número de instâncias do módulo *BaseModule* que é 4, o resultado dessa divisão é convertido para um número inteiro, caso não seja uma divisão exata. Ao contrário do número de filtros, cada instância do módulo *BaseModule* utiliza um tamanho de janela de *kernel* diferente, a primeira instância utiliza um *kernel* de tamanho 3, a segunda tem tamanho 5, a terceira tem tamanho 7 e a última instância tem tamanho 9. Após o processamento por cada uma das instâncias do *BaseModule*, os resultados de cada uma delas é então concatenado e propagados para fora do módulo.

Por fim, o módulo mais básico da etapa de extração de características do modelo proposto *Gated Activation Net* é o módulo *BaseModule*. O *BaseModule* recebe como parâmetro o número de filtros a serem utilizados na convolução junto com o tamanho do *kernel*. Além da operação básica de convolução, este módulo replica a operação de *Gated Activation* (GA) utilizada na *SwishNet* e na *Wavenet*. A Figura 28 ilustra como este módulo é organizado, nela é possível ver que os dados de entrada passam primeiro por uma convolução 1d com *kernel* de tamanho k e com n

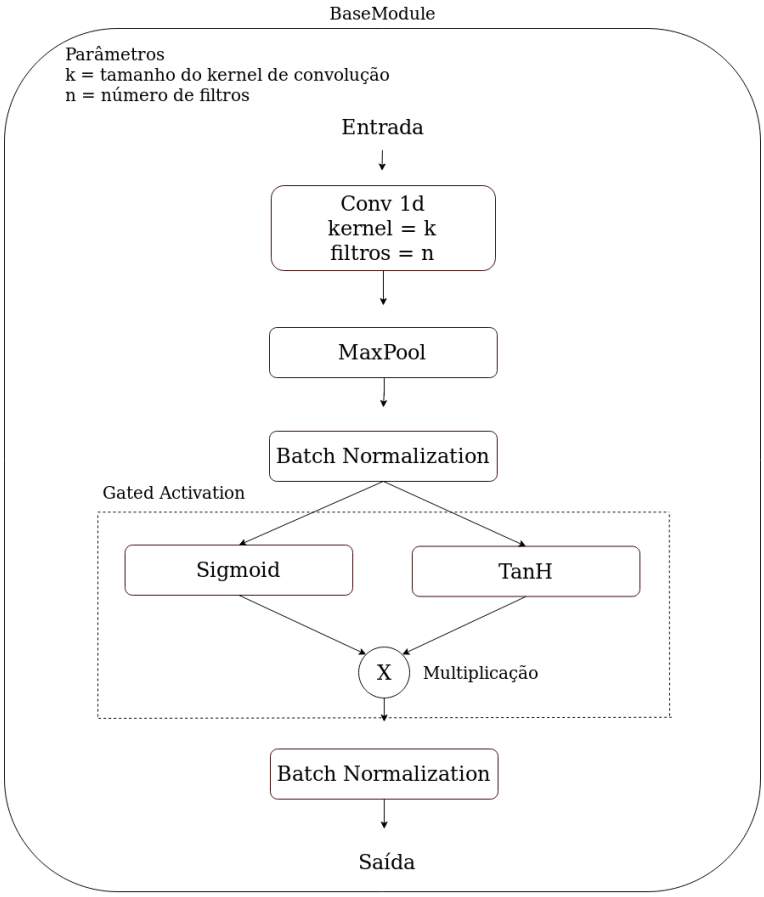


Figura 28 – Módulo *BaseModule* no modelo proposto *Gated Activation Net*.

filtros de convolução, onde k e n são parâmetros requeridos pelo módulo. Depois da camada de convolução, as características extraídas são reduzidas a partir de uma camada de *max pooling* na qual só permanecerão as características que tiveram maior grau de ativação nos filtros de convolução, ou seja, as características que representam os padrões que os filtros de convolução estão procurando. Após a redução da dimensionalidade feita pela camada de *max pooling*, os dados são normalizados por uma camada de *Batch Normalization* (BN) e depois passam por um *Gated Activation* (GA). No GA, os dados são transformados em paralelo por duas funções de ativação diferentes: a *Sigmoid* e a *TanH*. Depois de propagar os dados em paralelo pelas funções de ativação *Sigmoid* e *TanH*, os dados são unidos novamente através de uma multiplicação dos resultados adquiridos em cada função de ativação. Por fim, após a saída dos dados do GA, eles são novamente normalizados por uma camada de BN e são propagados para fora do *BaseModule*.

4.4.3 Classificação

A etapa de classificação é realizada logo após a etapa de extração de características. Ela é responsável por aprender relações entre as características extraídas

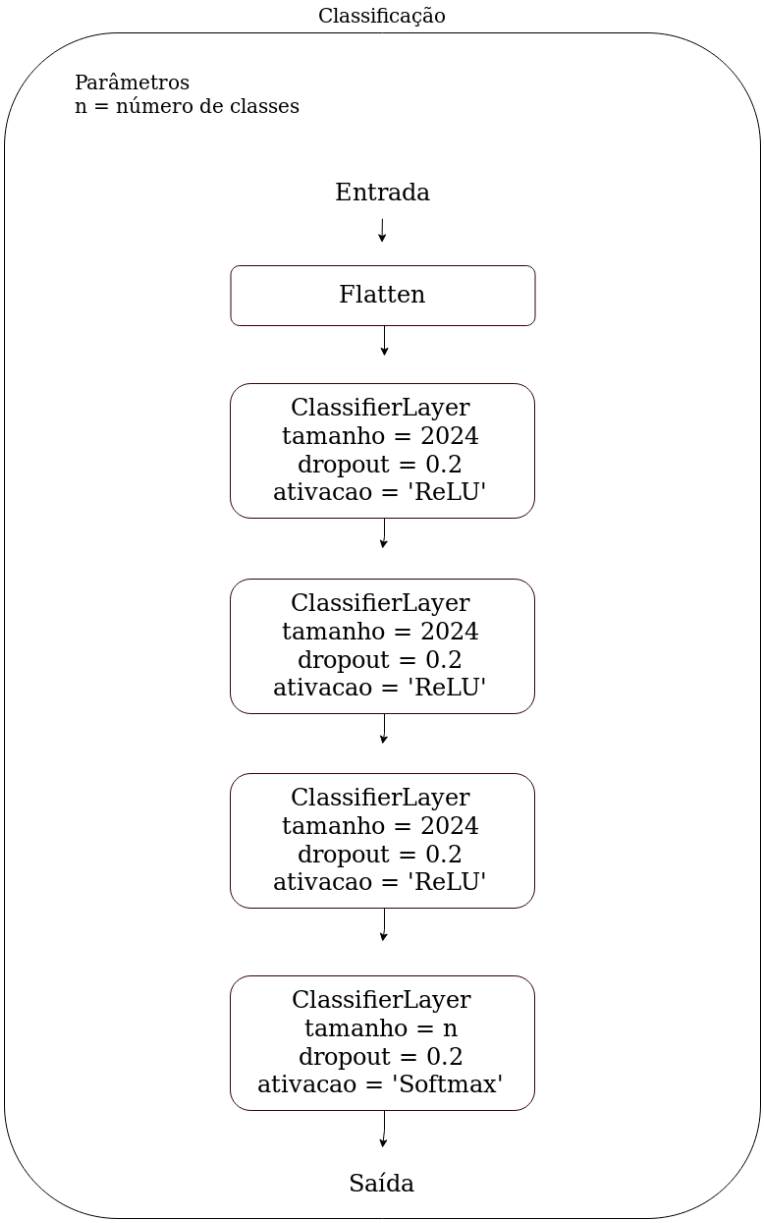


Figura 29 – Módulo de Classificação no modelo proposto Gated Activation Net.

na etapa anterior e as classes de locutores dispostas na base de dados. A Figura 29 ilustra como é organizado o módulo de classificação no modelo proposto *Gated Activation Net*. Nele, os dados de entrada vindos do módulo de extração de características (Subseção 4.4.2) são primeiro redimensionados para um *array* 1d através de uma camada de *flatten*. Esse redimensionamento é feito concatenando a saída final de cada filtro resultante do módulo de extração de características. Após isso, os dados são propagados através de três camadas totalmente conectadas organizadas pelo módulo *ClassifierLayer*. Cada instância do módulo *ClassifierLayer* é configurada com 2024 neurônios, um *dropout* de 0.2 e a função de ativação *ReLU*. Por último, mais uma instância do módulo *ClassifierLayer* é configurada com a função de mapear os dados finais nas n diferentes classes da base de dados, onde n

é o número de classes que é recebido por parâmetro pelo módulo de classificação. Essa última instância, possui *dropout* de 0.2 e função de ativação *Softmax*.

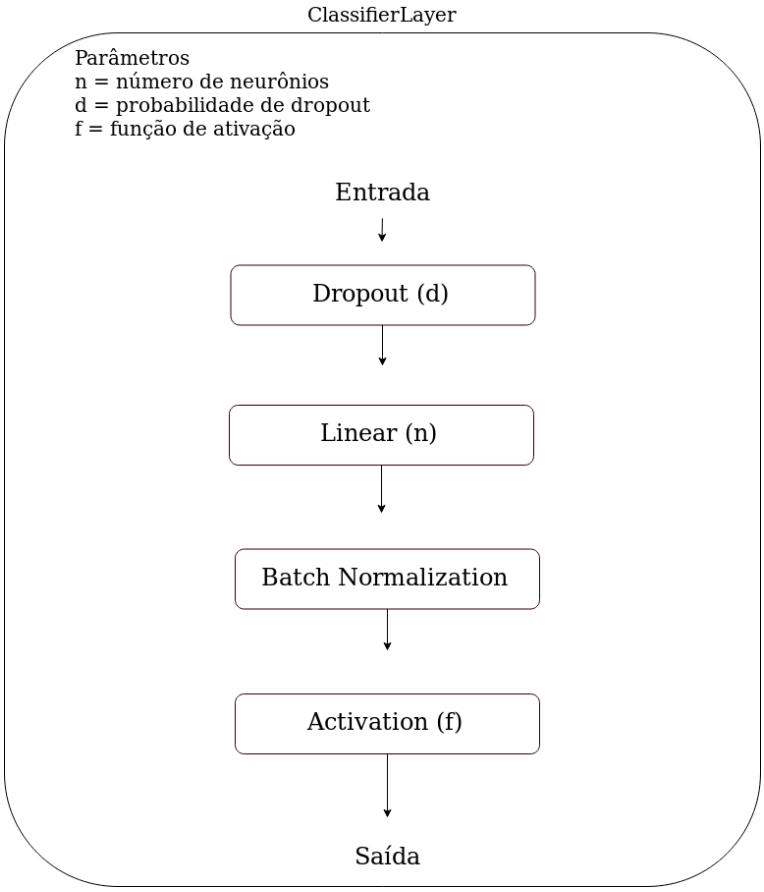


Figura 30 – Módulo *ClassifierLayer* no modelo proposto Gated Activation Net.

O módulo *ClassifierLayer* funciona como uma camada totalmente conectada, porém ele compacta a camada totalmente conectada e algumas outras operações necessárias em um único módulo. A Figura 30 ilustra como esse módulo é organizado e quais operações são aplicadas ao propagar os dados por ele. Primeiro, uma camada de *dropout* é aplicada zerando aleatoriamente uma porcentagem d de dados que passa por ela, isso é feito para forçar a rede a aprender mais de um caminho para a classificação correta dos dados. Após isso, os dados resultantes são passados por uma camada linear que vai propagá-los em n novas informações, na qual a rede aprende através do processo de ajuste de pesos a relevância correta entre cada informação de entrada para as n informações de saída. Depois da camada linear, os dados são normalizados através de uma camada de *Batch Normalization* (BN) e são transformados por meio de uma função de ativação f. Os valores de probabilidade de *dropout* d, número de neurônios na camada linear n e função de ativação f são todos recebidos como parâmetros do módulo.

4.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram abordados os modelos propostos *Additive Margin SincNet* (AM-SincNet), *MobilNet1D*, *Gated Activation Net* e suas variações utilizando a função *Additive Margin Softmax* (AM-Softmax) e a camada de convolução *sinc* da *SincNet*. Foi mostrado como o *SincNet* junto com a *Additive Margin Softmax* (AM-Softmax) resultaram no método proposto AM-SincNet e a intuição por trás dessa união. Também foi visto como a *MobileNet*, criada para trabalhar com imagens, foi adaptada para trabalhar no problema de reconhecimento de locutor, assim como, a inspiração por trás da criação do método proposto *Gated Activation Net*.

5 EXPERIMENTOS

Este capítulo objetiva apresentar como foram elaborados os experimentos para avaliar os métodos propostos no capítulo anterior em comparação ao modelo base *SincNet*. A Seção 5.1 introduzirá as bases de dados utilizadas nos experimentos. A Seção 5.2 irá detalhar quais tipos de processamento foram realizados nas amostras de áudio para prepará-las para os experimentos. Na Seção 5.3 será mostrado como as bases de dados foram divididas em conjuntos de treino e teste. A Seção 5.4 apresenta quais métricas foram utilizadas para avaliar os modelos e como elas funcionam. Por fim, a Seção 5.5 detalha o ambiente utilizado para rodar os experimentos, desde a configuração da máquina até os parâmetros utilizados no processo de treinamento.

5.1 BASE DE DADOS

Com o intuito de avaliar a eficácia dos modelos propostos e compará-los com o modelo base *SincNet*, foram realizados experimentos utilizando duas bases de dados diferentes contendo uma diversidade de locutores: *TIMIT* (GAROFOLO et al., 1993) e *MIT* (Woo; Park; Hazen, 2006).

A base de dados *TIMIT* é construída a partir de amostras de áudios capturadas de diversos locutores com diferentes sotaques. Ao todo, a base de dados é formada por áudios de 630 pessoas e dentre essas pessoas existem amostras dos oito principais sotaques americanos. Sobre as amostras de áudio, todas foram gravadas em inglês e cada pessoa foi instruída a ler 10 sentenças consideradas foneticamente ricas. A base de dados foi levantada em esforço comum entre as instituições *Massachusetts Institute of Technology (MIT)*, *SRI International (SRI)* e *Texas Instruments, Inc. (TI)*. Ela é originalmente separada em dois grupos distintos: o conjunto de treinamento e o de teste. Neste trabalho, para uma comparação justa, foi seguido o mesmo protocolo utilizado no trabalho original da *SincNet*, descrito em, (Ravanelli; Bengio, 2018), no qual foi utilizado apenas o conjunto de treinamento da base de dados *TIMIT* que é formado por 462 locutores, com 8 amostras de áudio de cada, totalizando 3.969 amostras de áudio.

A base de dados *MIT* é construída a partir de amostras de áudio coletadas através de dispositivos móveis em uma variedade de ambientes para induzir a presença de ruído natural. As amostras de áudio deste *dataset* são divididas em dois conjuntos principais: Usuários autênticos e Impostores. A coleta dos dados foi feita em duas sessões separadas para as amostras dos usuários autênticos e uma outra ses-

são para os impostores, cada sessão durou em torno de 20 minutos e foi gravada em dias diferentes. Para simular condições normais do dia a dia, as gravações foram feitas em três cenários diferentes: em um escritório silencioso, em uma hall de entrada com barulho razoável e em uma rua barulhenta. A base de dados *MIT* é formada por 88 locutores, dos quais 39 são mulheres e 49 são homens. Durante as gravações cada participante foi instruído a ler uma lista de palavras contendo alguns nomes e sabores de sorvete, com isso, foram gravadas 54 frases por participante por sessão.

Apesar da base de dados *MIT* ser dividida em conjuntos de usuários autênticos e usuários impostores, o conjunto de usuários impostores não é completo. Não existem amostras de usuários impostores para cada usuário autêntico desta base de dados. Por esse motivo, para não comprometer o balanceamento da base de dados, neste trabalho foi considerado apenas o conjunto dos usuários autênticos do *dataset MIT*, que por sua vez, é formado por duas sessões gravadas em dias diferentes, somando 48 classes de locutores contendo 108 amostras de áudio cada, o que totaliza um total de 5.184 amostras de áudio.

5.2 PROCESSAMENTO DOS DADOS

Uma etapa de pré-processamento de dados é aplicada nas bases de dados citadas na seção anterior antes que elas sejam apresentadas aos modelos propostos nesse trabalho. Uma das etapas do processo de pré-processamento dos dados é a normalização dos dados que é aplicada às amostras de áudio antes delas servirem de entrada para os modelos de reconhecimento de locutor. Também foram removidos intervalos sem fala no começo e final das amostras de áudio, como pode ser visto na Figura 31. Além disso, como as amostras de áudio são de tamanhos diferentes e os modelos precisam de uma entrada de tamanho fixo, cada amostra de áudio foi dividida em *frames* de $200ms$ com $10ms$ de sobreposição entre eles, como é ilustrado na Figura 32. Para as amostras de áudio que não forem divisíveis pelo tamanho do *frame*, o resíduo dessa divisão não será considerado na classificação. As duas bases de dados possuem taxa de amostragem de 16.000 características por segundo.

Figura 31 – Remoção dos intervalos sem fala nos sinais de áudio.

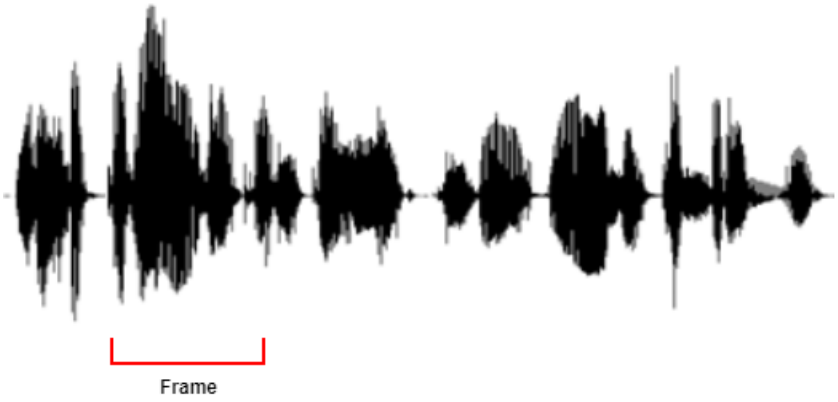


Figura 32 – Separação dos *frames* nos sinais de áudio.

5.3 DIVISÃO DOS CONJUNTOS DE DADOS

Para o processo de execução dos experimentos, foi utilizado *holdouts* como técnica de validação cruzada. Desta forma, as bases de dados tiveram que ser divididas em dois conjuntos distintos: conjunto de treino e conjunto de teste. Na base de dados *TIMIT* para cada locutor foram separadas cinco amostras aleatórias para o conjunto de treino e as três amostras remanescentes para o conjunto de teste. Da mesma forma, para a base de dados *MIT* foram separados 70% das amostras de forma aleatória para o conjunto de treinamento e os 30% restantes para o conjunto de testes. Em ambas as bases de dados os conjuntos de treino e teste são balanceados entre os locutores, além disso, a separação dos conjuntos de treinamento e teste foram feitas de forma a que uma amostra só pode pertencer a um dos dois conjuntos, ou seja, não há sobreposição de amostras entre os conjuntos de treinamento e teste.

5.4 MEDIDAS DE AVALIAÇÃO

Para o processo de avaliação e comparação dos modelos é necessária a utilização de algumas métricas que possam medir, de forma concisa, quão confiável é cada modelo. Desta forma, como medida de avaliação para os modelos deste trabalho, foram utilizadas as métricas que são comumente utilizadas em problemas desta natureza na literatura, como em (Ravanelli; Bengio, 2018), (GE et al., 2017), (Triefenbach; Martens, 2011), são elas: *Frame Error Rate* (FER) e *Classification Error Rate* (CER). Além do FER e o CER, também foi avaliado o tempo médio de execução de cada modelo.

Como explicado na Seção 5.2 e ilustrado na Figura 32, os sinais de áudio são divididos em *frames* para serem processados pelos modelos. Desta forma, o FER é o erro associado a cada *frame* da amostra de áudio processada pelo modelo. Ele é calculado a partir das probabilidades que o modelo oferece como resposta daquela amostra (*frame*) pertencer a uma determinada classe. Apesar do erro associado ao *frame* ser utilizado na decisão final para vincular um sinal de áudio ao seu devido locutor, ele não é em si a resposta final esperada pelo usuário, pois só representa a resposta parcial do modelo referente a 1 dos n *frames* que um sinal de áudio pode gerar.

A resposta do modelo a um determinado *frame* é chamada de *pout*. Ela contém a probabilidade do *frame* pertencer a cada uma das N classes da base de dados. A Equação 5.1 mostra como *pout* é classificada em uma das N classes. Nela é calculado a maior resposta de *pout* dentre todas as classes, atribuindo para *pred* a classe com maior probabilidade do *frame* pertencer. A Equação 5.2 calcula se o *frame* foi classificado corretamente ou não, comparando a classe que o *frame* foi classificado *pred* com o *label* correto para esse *frame*, *lab*. Desta forma, *diff* retorna 0 se o *frame* foi classificado corretamente e 1, caso contrário. Por fim, a Equação 5.3 mostra como o FER é calculado, nela é tirado a média do erro de todos os *frames* da base de dados. Onde F representa o número de *frames* na base de dados, $pred_f$ a classe na qual o $frame_f$ foi classificado e lab_f a classe na qual o $frame_f$ realmente pertence.

$$pred = max(pout) \tag{5.1}$$

$$diff(pred,\ lab) = \begin{cases} 1 & se \quad pred \neq lab \\ 0 & se \quad pred = lab \end{cases} \tag{5.2}$$

$$FER = \frac{1}{F} \sum_{f=1}^{F} diff(pred_f,\ lab_f) \tag{5.3}$$

Por outro lado, o CER é o erro final da classificação, o qual geralmente costuma ser menor que o FER pois considera o resultado da classificação de cada *frame* da amostra de áudio. A resposta final esperada pelo usuário é obtida verificando a classe com maior ativação depois de se ter calculado o somatório das respostas de cada *frame* para uma determinada amostra de áudio, desta forma, a métrica de avaliação CER tem relação direta com essa resposta final calculada.

A Equação 5.4 descreve como o resultado obtido na classificação de todos os *frames* de uma amostra de áudio é utilizado para determinar a classificação final dessa amostra. Nela, *best* representa a classe na qual a amostra de áudio foi classificada. Essa classificação é feita com base na classificação de cada um dos *frames* extraídos do sinal de áudio. *best* é calculado somando as probabilidades obtidas em cada um dos *frames*, e por fim, escolhendo a classe que obteve a maior soma, ou seja, a classe com maior probabilidade da amostra de áudio pertencer. Na Equação 5.4, F_n representa o número de *frames* de uma amostra de áudio n, enquanto $pout_f$ é um vetor contendo as probabilidades do *frame* f pertencer a cada uma das classe da base de dados. Por fim, a Equação 5.5 mostra como o *Classification Error Rate* (CER) é calculado. Nela, o CER é calculado através da média do resultado de classificação de cada amostra de áudio. Onde, N representa a quantidade de amostras de áudio pertencentes a base de dados, assim como, $best_n$ é a classe na qual a amostra n foi classificada e lab_n é a classe na qual a amostra n realmente pertence.

$$best = max\left(\sum_{f=1}^{F_n} pout_f\right) \tag{5.4}$$

$$CER = \frac{1}{N} \sum_{n=1}^{N} diff(best_n,\ lab_n) \tag{5.5}$$

Além das métricas relacionadas à acurácia dos modelos, neste trabalho também foi avaliado o tempo de inferência de cada modelo, o número de parâmetros e o tamanho do modelo em *megabytes*. O tempo de inferência foi medido com modelos já treinados e se refere ao tempo, em milissegundos, gasto pela máquina para rodar um *batch* de tamanho 128. A máquina utilizada para mensurar o tempo necessário para execução de cada modelo é descrita na Tabela 1. Como as amostras de áudio podem ter tamanhos diferentes, a análise do tempo gasto para processar cada amostra pode não significar muito, dado que a variância desse tempo provavelmente seria alta. Desta forma, para uma comparação mais justa, foi analisado o tempo de execução do modelo por *batch*. Os *batchs* foram fixados em um tamanho

de 128 para cada modelo, o que significa que o tempo mensurado se refere a execução de 128 *frames*. Os resultados de tempo médio de execução são calculados a partir do tempo de execução medido em 6.561 *batches* e serão apresentados junto com seu respectivo desvio-padrão.

5.5 AMBIENTE

Para realização dos experimentos utilizando o modelo base *SincNet* descrito na Seção 3.2 e as arquiteturas propostas no Capítulo 4, foi utilizada uma máquina nas configurações descritas na Tabela 1. Além disso, para o treinamento dos modelos foram utilizadas as mesma configurações do artigo original do *SincNet* (Ravanelli; Bengio, 2018), que se resume em utilizar o otimizador *RMSpropo* com taxa de aprendizagem $lr = 0,001$, $\alpha = 0,95$, $\epsilon = 10^{-7}$. Cada modelo foi treinado por 360 épocas, o que se mostrou ser suficiente para uma convergência. Para os métodos que se utilizam de algum número aleatório, como a inicialização dos pesos dos modelos, foi setado uma $seed = 1234$ para manter a consistência dos experimentos.

Tabela 1 – Configurações da máquina utilizada para executar os experimentos com os modelos.

Sistema Operacional	Ubuntu 18.04.3 LTS
CUDA	CUDA 10.1
GPU	NVIDIA RTX 2060
Processador	Core i5-9400F
Memória	24GB
Armazenamento	480GB SSD
Biblioteca	PyTorch 1.0

5.6 CONSIDERAÇÕES FINAIS

Neste capítulo foram introduzidas as bases de dados utilizadas nos experimentos deste trabalho, como elas foram desenvolvidas e as características que diferenciam uma da outra. Também foi explicado o processo de pré-processamento utilizado nas amostras de áudio, assim como a divisão das bases de dados nos conjuntos de treinamento e teste, as métricas de avaliação dos modelos e o ambiente onde os experimentos foram executados.

6 RESULTADOS E DISCUSSÃO

Este capítulo apresenta o resultado dos experimentos que foram realizados nas bases de dados *TIMIT* e *MIT* (Seção 5.1) com o modelo base *SincNet* (Seção 3.2) e os modelos propostos no Capítulo 4. Neste capítulo é feita a comparação dos modelos com base nas métricas de erro *Frame Error Rate* (FER) e *Classification Error Rate* (CER) que foram descritas na Seção 5.4, assim como o tempo de inferência, número de parâmetros e tamanho de cada modelo. A Seção 6.1 apresenta os resultados obtidos com o modelo proposto *Additive Margin SincNet* (AM-SincNet), nela é feita uma análise da evolução do FER durante as épocas de treinamento. Além disso, o AM-SincNet, por fazer uso da margem aditiva do AM-Softmax acaba recebendo um novo parâmetro m para controlar o tamanho da margem aditiva utilizada, desta forma, foi realizado um estudo utilizando 19 valores de m diferentes, variando entre $0,05$ e $0,95$ em intervalos de $0,05$, para tentar descobrir a influência do parâmetro m nos resultados de FER e CER. A Seção 6.2 apresenta os resultados obtidos com o modelo proposto baseado no *MobileNet*, da mesma forma, a Seção 6.3 apresenta os resultados do modelo baseado no *MobileNet*, porém, com a adição da camada *sinc* do *SincNet*. Nas Seções 6.4 e 6.5 é possível ver os resultados dos experimentos realizados com o modelo proposto *Gated Activation Net*, nelas estão os resultados do modelo proposto lidando com os sinais de áudio diretamente e com a adição da camada *sinc*, respectivamente. Por fim, a Seção 6.6 faz uma comparação dos resultados apresentados no capítulo, analisando a influência do uso da AM-Softmax e das camadas *sinc*. Ao decorrer das seções abaixo serão apresentados gráficos mostrando a evolução do FER durante as épocas de treinamento, mas o mesmo não foi feito para o CER, dado que os valores de CER calculados nos modelos começam muito altos, assim como acontece com o FER, porém, ao decorrer das épocas esses valores caem muito e ficam variando próximo de zero, tornando difícil uma comparação clara entre a evolução do CER de dois ou mais modelos nas últimas épocas de treinamento.

6.1 *ADDITIVE MARGIN SINCNET*

O *Additive Margin SincNet* (AM-SincNet), como descrito na Seção 4.1, é uma junção do modelo de rede neural para reconhecimento de locutor, o *SincNet* (Seção 3.2), com a função de ativação derivada da *Softmax*, a *Additive Margin Softmax* (AM-Softmax) (Seção 3.3). Como visto na Seção 3.3, a AM-Softmax introduz dois novos parâmetros em relação a *Softmax* tradicional: o parâmetro s para contro-

lar a escala das *features* e o parâmetro m para controlar o tamanho da margem de separação aditiva. Desta forma, o método proposto AM-SincNet também ganha dois parâmetros adicionais quando comparado ao *SincNet* em sua forma tradicional, proposta por (Ravanelli; Bengio, 2018), porém, nos experimentos realizados com o método proposto AM-SincNet, o parâmetro s foi fixado para um número inteiro $s = 30$, assim como em (WANG et al., 2018), restando assim, analisar a influência do parâmetro m para controlar o tamanho da margem aditiva nas métricas de avaliação *Frame Error Rate* (FER) e *Classification Error Rate* (CER). Essa avaliação foi feita a partir de experimentos realizados nas bases de dados *TIMIT* e *MIT* (Seção 5.1).

Tabela 2 – Análise da evolução do *Frame Error Rate* (%) durante as épocas de treinamento com os métodos *SincNet* e AM-SincNet na base de dados *TIMIT* .

Época	SincNet	AM-SincNet ($m =$)																		
		0,05	0,10	0,15	0,20	0,25	0,30	0,35	0,40	0,45	0,50	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95
0	**97,25**	98,34	98,61	99,10	99,09	98,51	98,83	98,77	98,76	98,71	99,06	98,08	99,13	98,14	97,65	98,21	98,78	97,79	98,79	97,50
16	55,32	60,89	59,21	63,17	61,18	57,10	56,77	56,70	57,93	57,29	58,37	54,09	56,44	54,69	57,23	60,98	55,65	54,25	56,18	**53,72**
32	50,29	51,14	48,43	45,71	43,98	50,93	43,87	44,20	46,37	44,57	43,46	44,23	45,56	49,98	44,84	44,32	48,68	42,57	46,93	**41,48**
48	46,67	41,28	45,68	40,95	40,45	40,10	40,02	41,99	39,88	45,43	40,54	40,49	39,17	41,25	38,87	37,95	42,45	**37,58**	39,18	41,60
64	45,40	37,92	39,39	42,34	36,52	46,67	39,19	41,51	38,05	42,05	38,02	38,13	37,45	36,83	38,86	37,36	37,34	40,26	**35,67**	36,91
80	43,49	41,57	35,71	35,15	34,55	36,90	34,55	36,30	36,37	36,57	34,89	36,34	36,99	34,47	34,11	34,72	34,51	35,36	35,67	**33,29**
96	44,83	34,25	34,54	35,73	33,05	35,34	33,55	34,37	34,11	33,50	33,68	36,82	33,41	33,07	33,13	34,00	34,14	33,32	33,50	**32,78**
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
328	46,39	27,79	28,19	27,47	27,00	28,64	28,77	28,76	28,21	27,82	27,37	28,82	27,40	27,54	27,90	29,39	28,32	27,74	**26,70**	27,47
344	47,93	26,80	27,08	27,71	27,08	28,69	28,72	27,92	28,73	29,00	27,42	27,50	27,18	27,54	30,00	27,60	28,68	**26,18**	27,64	26,69
360	44,64	27,06	28,96	27,62	28,97	27,47	27,90	29,22	27,57	27,07	27,86	27,81	28,28	27,92	29,76	26,95	30,85	26,95	**26,54**	26,64

A Tabela 2 mostra a evolução do *Frame Error Rate* (FER) durante as épocas de treinamento na base de dados *TIMIT* para o *SincNet* e o método proposto AM-SincNet. Nela é possível ver os valores de erro (FER) em porcentagem calculados em cada época no conjunto de teste para o *SincNet* e para o método proposto AM-SincNet [1]. Para verificar a influência do parâmetro m do AM-SincNet na métrica de erro FER, foram realizados diversos experimentos com vários valores diferentes de m no intervalo $0,05 \leq m \leq 0,95$. Nesses experimentos ambos os métodos foram treinados por exatas 360 épocas que se mostraram suficientes para uma convergência dos valores, porém a Tabela 2 mostra apenas os resultados das primeiras 96 e últimas 32 épocas (esse intervalo foi escolhido para privilegiar a visualização dos dados). Nela o melhor resultado (menor valor) de cada época foi destacado em negrito, desta forma, é possível ver que o *SincNet* só conseguiu o melhor resultado na época 0, onde nenhum dos métodos tinham tido treinamento suficiente. A partir da época 16 o método proposto já mostra melhores resultados de FER para alguns

[1] Os resultados apresentados nessa seção foram parcialmente publicados no artigo *Additive Margin SincNet for Speaker Recognition*, (Chagas Nunes; Macêdo; Zanchettin, 2019).

valores de m, como é o caso de $m = 0,55$, $m = 0,65$, $m = 0,85$ e $m = 0,95$. A diferença do FER do *SincNet* e o resultado do método proposto com $m = 0,95$, que é o melhor valor da época, é de $1,6$ pontos percentuais nessa época. Na época 32 a maioria dos valores de FER do AM-SincNet já superam o resultado do *SincNet*, tendo seu melhor valor com $m = 0,95$ que já mostra uma diferença de $8,81$ em relação ao FER do *SincNet*. A partir da época 80 o método proposto já é melhor que o *SincNet* para qualquer valor de m experimentado e a diferença entre o FER calculado no *SincNet* e o calculado no AM-SincNet com o melhor valor de m da época já é maior que 10. Nas 32 últimas épocas do experimento é possível ver que os valores de FER do AM-SincNet estão convergindo para um número entre 26 e 30, enquanto o FER do *SincNet* convergiu para um valor aproximado de 44.

Para demonstrar de forma visual, os resultados completos dos experimentos com o *SincNet* e o método proposto AM-SincNet na base de dados *TIMIT* são plotados na Figura 33. Assim como na Tabela 2, os resultados da Figura 33 mostram a evolução do FER durantes as épocas de treinamento, nela é possível ver a evolução do erro durante todas as épocas de treinamento. A Figura 33 é dividida em 4 subimagens, nas subimagens (a) e (b) é plotado o FER do *SincNet* e do AM-SincNet para 19 valores diferentes do parâmetro m. Na subimagem (a) m foi variado no intervalo $0,05 \leq m \leq 0,50$, já na subimagem (b) m variou no intervalo $0,55 \leq m \leq 0,95$. Nas subimagens (c) e (d) é plotado o FER do *SincNet* junto com a média do FER para os diversos valores de m do AM-SincNet em cada época. A média do FER do AM-SincNet na subimagem (c) é calculado a partir dos valores de m utilizados na subimagem (a). Da mesma forma, a média do FER do AM-SincNet na subimagem (d) é calculado a partir dos valores de m utilizados na subimagem (b). Apesar do método proposto apresentar algumas variações no FER para diferentes valores de m durante as épocas de treinamento, como pode ser visto na Figura 33, subimagens (a) e (b), ao decorrer das épocas o FER calculado para qualquer valor de m aparenta tender para um único valor, o que pode indicar que o parâmetro m não precisar ser otimizado e que todos os valores de m testados se mostram igualmente bons nessa base de dados. Desta forma, para simplificar os resultados obtidos, as subimagens (c) e (d) mostram uma média do FER para cada valor de m utilizado no método proposto AM-SincNet assim como o FER calculado no *SincNet* tradicional, esses resultados apontam uma significativa diferença entre os dois métodos nessa base de dados, mostrando quão menor (melhor) é o erro FER calculado no método proposto. Mesmo considerando as variações obtidas nas subimagens (a) e (b) é possível ver que a partir da época 80 nenhuma variação do método proposto apresentou FER inferior ao *SincNet* tradicional.

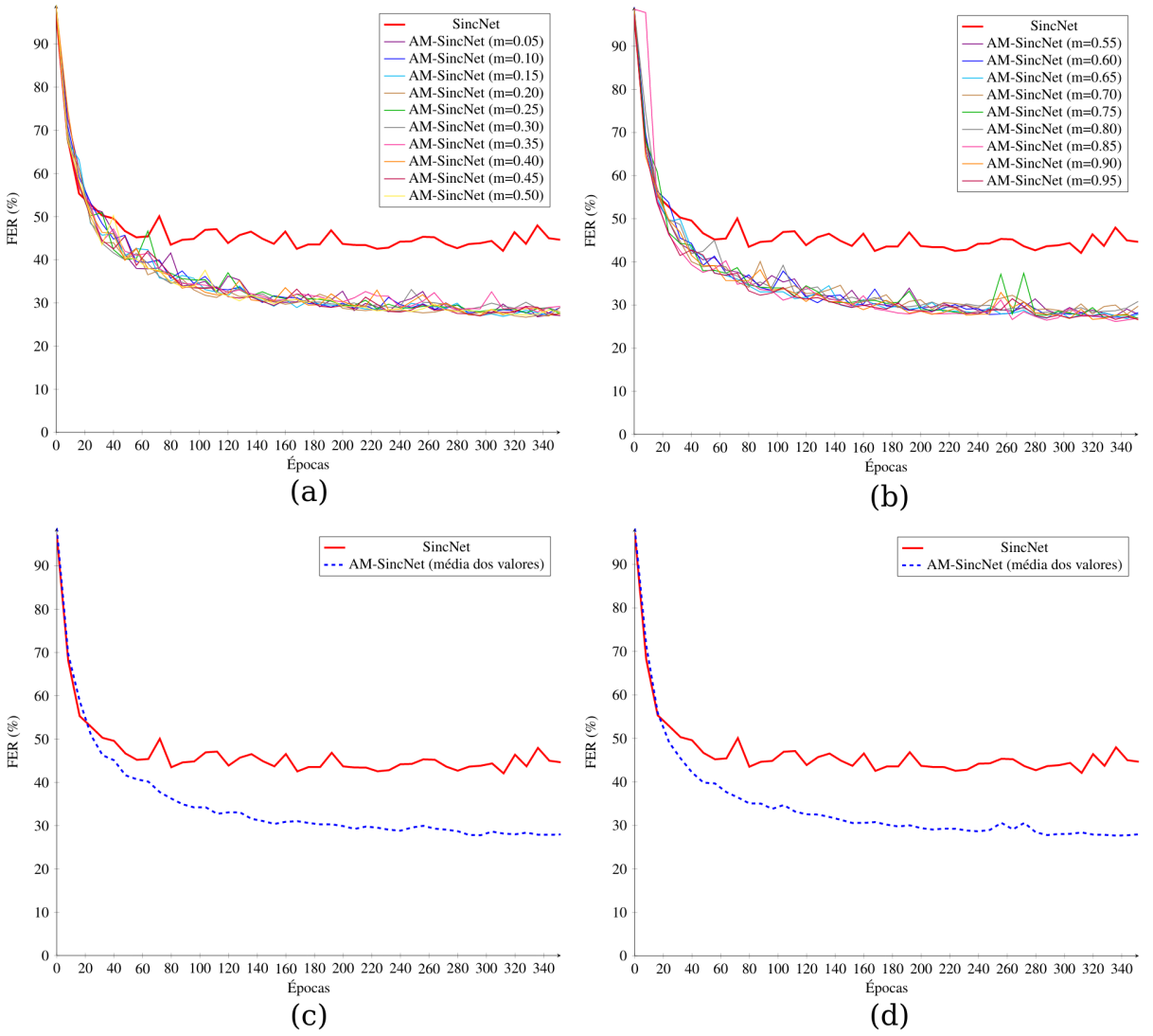


Figura 33 – Análise da evolução do *Frame Error Rate* (%) durante as épocas de treinamento com os métodos *SincNet* e AM-SincNet na base de dados *TIMIT*. (a) Resultados variando o parâmetro m no intervalo de $0,05$ e $0,50$. (b) Resultados variando o parâmetro m no intervalo de $0,55$ e $0,95$. (c) Resultados da média dos valores de m utilizados no intervalo de (a). (d) Resultados da média dos valores de m utilizados no intervalo de (b).

Assim como na Tabela 2, a Tabela 3 também mostra a evolução da medida de erro FER para o *SincNet* e o método proposto AM-SincNet, porém, na Tabela 3 os resultados são referentes ao conjunto de teste da base de dados *MIT*. Os resultados de FER são mostrados em porcentagem e o melhor resultado de cada época é destacado em negrito. Assim como na tabela anterior, para verificar a influência do parâmetro m no método proposto, foram realizados diversos experimentos variando m de $0,05$ a $0,95$ em intervalos de $0,05$. Em concordância com a tabela anterior, nos experimentos realizados na base de dados *MIT* o *SincNet* só superou o método proposto na primeira época quando nenhum dos dois métodos havia re-

Tabela 3 – Análise da evolução do *Frame Error Rate* (%) durante as épocas de treinamento com os métodos *SincNet* e AM-SincNet na base de dados *MIT*.

Época	SincNet	AM-SincNet ($m =$)																		
		0,05	0,10	0,15	0,20	0,25	0,30	0,35	0,40	0,45	0,50	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95
0	**87,96**	91,20	91,02	88,58	90,82	89,41	91,09	90,24	89,73	90,47	89,15	90,05	90,24	90,97	90,22	91,08	88,24	92,48	89,19	90,88
16	47,45	45,28	47,49	44,63	46,70	47,39	44,67	47,64	47,02	47,58	47,67	44,90	46,64	45,54	44,78	46,16	**44,04**	44,71	46,44	45,76
32	43,11	38,44	38,69	38,72	38,14	39,13	38,15	40,06	37,96	37,96	38,29	41,27	39,09	38,74	40,16	39,00	40,92	38,30	**37,71**	38,47
48	39,89	35,22	**34,77**	35,32	35,48	35,09	34,93	35,54	36,09	35,73	36,09	36,36	36,08	35,90	38,55	35,35	37,04	35,89	35,47	35,49
64	38,70	35,02	34,00	34,90	34,74	35,09	35,28	35,79	34,48	**33,85**	34,30	34,06	36,11	36,10	35,47	34,12	36,70	34,99	34,63	36,18
80	38,33	34,66	32,74	**32,39**	33,28	33,47	34,13	33,33	33,86	33,02	33,23	34,48	33,47	33,80	33,83	32,85	33,89	32,97	33,81	34,09
96	38,99	33,16	32,76	32,32	34,46	35,03	32,68	33,19	34,53	33,32	33,15	32,89	33,11	**32,25**	33,03	33,89	33,09	33,14	32,75	33,20
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
328	35,75	30,79	29,83	29,89	29,74	30,09	29,31	29,98	30,93	29,43	29,84	30,16	29,69	34,51	30,48	**28,83**	29,12	29,53	29,88	29,24
344	35,26	30,06	30,27	29,81	29,74	30,15	29,30	29,28	30,57	29,30	29,53	29,68	30,09	29,40	**29,02**	29,51	29,49	29,55	30,82	29,90
360	34,65	29,29	**29,00**	32,13	29,54	29,57	29,19	29,45	29,63	29,11	30,39	29,55	29,89	29,49	29,46	29,21	29,65	30,37	29,44	29,35

cebido treinamento suficiente. Na época 16 o método proposto já apresenta uma melhora sutil do FER para $m = 0,95$ quando comparado ao *SincNet*. A partir da época 32 o método proposto tem um erro menor que o *SincNet* para qualquer valor de m testado. Nas últimas 32 épocas o erro do *frame* calculado para o *SincNet* aparenta convergir para um valor aproximado de 34 enquanto o AM-SincNet caminha para um valor em torno de 29 independente do m utilizado, isso mostra uma diferença de 5 pontos a favor do método proposto quando uma comparação entre os dois métodos é feita.

A Figura 34 apresenta os resultados de FER dos experimentos realizados com o *SincNet* e o método proposto AM-SincNet na base de dados *MIT*. Nela é possível ver em forma de plots os resultados dispostos na Tabela 3. Assim, como na Figura 33, a Figura 34 é dividida em 4 subimagens, em (a) e (b) é mostrado o FER do *SincNet* e do método proposto para as 19 variações do parâmetro m realizadas. A (a) contém os resultados do *SincNet* junto com os resultados do AM-SincNet utilizando m no intervalo $0,05 \leq m \leq 0,50$, já na (b) o parâmetro m é variado no intervalo $0,55 \leq m \leq 0,95$. Nas subimagens (c) e (d), além do resultado do *SincNet* é plotado o resultado da média dos erros obtida com diversos valores de m para o método proposto. Na subimagem (c) o m variou no intervalo da (a) e na (d) o parâmetro m foi variado no intervalo da subimagem (b). A partir da época 40 é possível observar nas subimagens (a) e (b) que nenhuma das variações de m do método proposto obteve FER maior (pior) que o calculado no *SincNet*, e, apesar das variações de erro calculadas para cada valor de m testado, ao decorrer das épocas os resultados do AM-SincNet aparentam convergir para o mesmo valor. Nas subimagens (c) e (d) é possível ver com mais clareza que o método proposto obteve resultados de FER melhores (menores) que os observados no *SincNet* tradicional, e, apesar da distância entre os valores observados aqui serem um pouco menor que

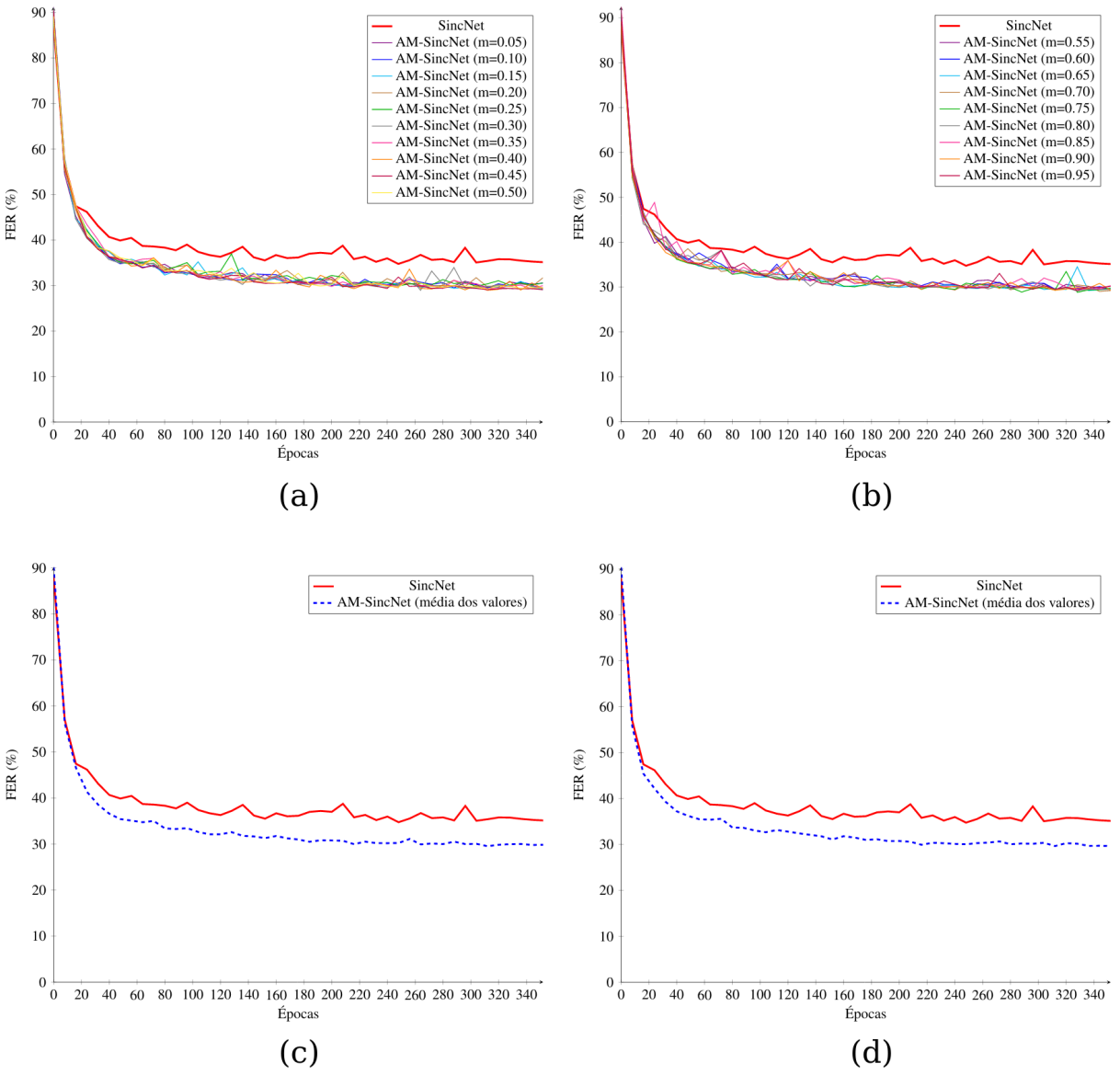


Figura 34 – Análise da evolução do *Frame Error Rate* (%) durante as épocas de treinamento com os métodos *SincNet* e AM-SincNet na base de dados *MIT*. (a) Resultados variando o parâmetro m no intervalo de $0,05$ e $0,50$. (b) Resultados variando o parâmetro m no intervalo de $0,55$ e $0,95$. (c) Resultados da média dos valores de m utilizados no intervalo de (a). (d) Resultados da média dos valores de m utilizados no intervalo de (b).

a vista na base de dados *TIMIT*, o método proposto ainda se mostra superior nessa base de dados.

Por fim, a Tabela 4 apresenta um resumo dos resultados do *SincNet* e do método proposto AM-SincNet nas bases de dados *TIMIT* e *MIT* utilizando as métricas de erro *Frame Error Rate* (FER) e *Classification Error Rate* (CER). Os resultados são um resumo do erro, FER e CER, calculado na última época de treinamento de cada modelo. Para o método proposto AM-SincNet estão inclusos os resultados para cada variação do parâmetro m que foi variado de $0,05$ até $0,95$ em intervalos de $0,05$. Na

Tabela 4 – Comparação dos resultados do *SincNet* com o modelo proposto, AM-SincNet, nas bases de dados *TIMIT* e *MIT*.

	TIMIT		MIT	
	FER	CER	FER	CER
SincNet	44,64	1,15	34,65	1,27
AM-SincNet ($m = 0,05$)	27,06	0,50	29,29	0,52
AM-SincNet ($m = 0,15$)	28,91	0,57	32,13	0,92
AM-SincNet ($m = 0,20$)	**26,98**	0,36	29,54	0,57
AM-SincNet ($m = 0,25$)	29,30	0,64	29,57	0,81
AM-SincNet ($m = 0,30$)	28,44	0,43	29,19	0,52
AM-SincNet ($m = 0,35$)	29,22	0,50	29,45	0,46
AM-SincNet ($m = 0,40$)	27,57	0,43	29,63	0,52
AM-SincNet ($m = 0,45$)	27,07	0,43	**29,11**	**0,34**
AM-SincNet ($m = 0,50$)	27,86	0,50	30,30	0,52
AM-SincNet ($m = 0,55$)	27,81	0,36	29,55	0,46
AM-SincNet ($m = 0,60$)	28,28	0,50	29,89	0,40
AM-SincNet ($m = 0,65$)	27,92	0,50	29,49	**0,34**
AM-SincNet ($m = 0,70$)	29,76	0,64	29,46	0,69
AM-SincNet ($m = 0,75$)	26,95	**0,28**	29,21	0,46
AM-SincNet ($m = 0,80$)	30,85	0,50	29,65	0,69
AM-SincNet ($m = 0,85$)	28,32	0,50	30,37	0,69
AM-SincNet ($m = 0,90$)	29,23	0,43	29,44	0,52
AM-SincNet ($m = 0,95$)	27,81	0,57	29,35	0,52

base de dados *TIMIT*, o *SincNet* ficou com um FER de $44,64$, enquanto o método proposto convergiu pra um valor aproximado de $28,00$. Na mesma base de dados, o CER calculado para o *SincNet* foi de $1,15$ contra um valor aproximado de $0,5$ para o AM-SincNet. Utilizando ambas as medidas de erro, na base de dados *TIMIT* o método proposto foi melhor que o *SincNet,* tanto na média quanto em qualquer variação do parâmetro m utilizado. Do mesmo modo, na base de dados *MIT* o método proposto obteve melhores resultados em FER e CER para todas as variações de m testadas. Nela, o método proposto ficou com um valor médio de FER por volta de

$29,00$ contra $34,65$ do *SincNet*. Ainda, o CER observado para o método proposto na base de dados *MIT* variou por volta de $0,5$, enquanto o do *SincNet* ficou em $1,27$. Em ambas as bases de dados, o método proposto mostrou um CER médio de menos da metade do calculado no *SincNet*, enquanto apresenta uma queda considerável do FER calculado para base de dados *TIMIT*.

6.2 *AM-MOBILENET1D*

Os métodos propostos, *MobileNet1D* e AM-MobileNet1D, descritos na Seção 4.2, são uma adaptação da MobileNet V2 (Sandler et al., 2018) que visa trabalhar com sinais de áudio, mais precisamente com a tarefa de identificação de locutor. Para avaliar a eficácia dos métodos propostos, foram realizados diversos experimentos nas bases de dados *TIMIT* e *MIT* analisando duas medidas de erros comuns na tarefa de reconhecimento de locutor, são elas: *Frame Error Rate* (FER) e *Classification Error Rate* (CER). Nos experimentos realizados com o modelo proposto AM-MobileNet1D o parâmetro m da AM-Softmax foi fixado em $0,5$, uma vez que os experimentos com a AM-SincNet mostraram uma certa tendência de convergência para um único valor de FER e CER independente do m utilizado.

Tabela 5 – Comparação dos resultados do *SincNet* com os modelos propostos, *MobileNet1D* e AM-MobileNet1D, nas bases de dados *TIMIT* e *MIT*.

	TIMIT		MIT	
	FER	CER	FER	CER
SincNet	44,64	1,15	**34,65**	**1,27**
MobileNet1D	26,50	0,57	38,75	2,54
AM-MobileNet1D	**21,30**	**0,43**	35,55	2,19

A Tabela 5 apresenta os resultados de FER e CER em porcentagem para o modelo base *SincNet* e os modelos propostos *MobileNet1D* e AM-MobileNet1D, os resultados foram obtidos a partir de experimentos realizados nas bases de dados *TIMIT* e *MIT*. Como foi visto na Seção 3.3, a AM-Softmax introduz o novo parâmetro m para controlar o tamanho da margem aditiva, porém, com base nos experimentos da seção anterior, o valor de m utilizado aqui foi fixado para $0,5$. Na tabela é possível ver que os métodos propostos obtiveram resultados de FER e CER menor (melhor) que o calculado utilizando o *SincNet* na base de dados *TIMIT*. Por outro lado, na base de dados *MIT* o *SincNet* aparenta ter conseguido melhores resultados de FER e CER que ambos os métodos propostos. Analisando o resultado dos méto-

dos propostos em ambas as bases de dados, a utilização da AM-Softmax aparenta ter trazido alguma melhora para as taxas de FER e CER calculadas.

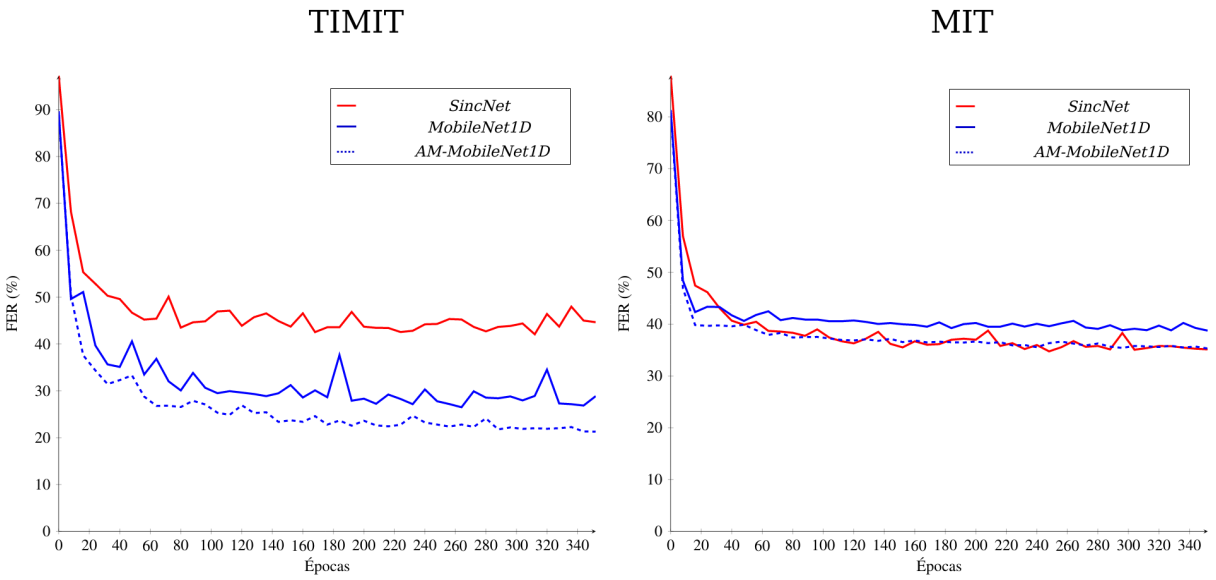


Figura 35 – Análise da evolução do *Frame Error Rate* (%) durante as épocas de treinamento com os métodos *SincNet*, *MobileNet1D* e AM-MobileNet1D nas bases de dados *TIMIT* e *MIT*.

Na Figura 35 é possível ver a evolução do FER durante as épocas de treinamento para o *SincNet* e os métodos propostos, *MobileNet1D* e AM-MobileNet1D, nas bases de dados *TIMIT* e *MIT*. Nela é possível ver que para a base de dados *TIMIT*, os métodos propostos obtiveram FER menor (melhor) que o calculado no *SincNet*. Por outro lado, a situação é um pouco diferente na base de dados *MIT*. Nela, o *SincNet* mostrou resultados melhores de FER do que o *MobileNet1D*. Para o AM-MobileNet1D, o FER calculado foi equivalente ao da *SincNet* e melhor que o da *MobileNet1D*. Nas duas bases de dados o uso da AM-Softmax ajudou a diminuir o erro calculado. O resultado obtido na base de dados *MIT* pode ser um indício de que os métodos propostos não se adequem bem a bases de dados com presença de ruídos, uma vez que as amostras da base de dados *MIT* foram gravadas em ambientes com ruído natural, como foi descrito na Seção 5.1. No entanto, em ambientes sem a presença de ruído, os métodos propostos mostraram resultados melhores que o *SincNet*, mesmo quando há poucas amostras disponíveis para o treinamento dos modelos, que é o caso da base de dados *TIMIT*.

6.3 *SINC AM-MOBILENET*

Os métodos propostos *Sinc MobileNet1D* e *Sinc AM-MobileNet1D* (Seção 4.3) são uma combinação do *MobileNet1D* e *AM-MobileNet1D* (Seção 4.2), respectivamente,

com as funções *sinc* (Subseção 2.2.1) do *SincNet* (Seção 3.2). Essa combinação é feita utilizando a camada *sinc* da *SincNet* como extrator de características inicial para o modelo, sendo assim, os sinais de áudio são processados primeiro pela camada *sinc* para depois servirem de entrada para o restante da rede. Para testar a eficácia dos métodos propostos em comparação ao *SincNet,* foram realizados diversos experimentos nas bases de dados *TIMIT* e *MIT* avaliando os métodos nas métricas de FER e CER.

Tabela 6 – Comparação dos resultados do *SincNet* com os modelos propostos, *Sinc MobileNet1D* e *Sinc AM-MobileNet1D*, nas bases de dados *TIMIT* e *MIT*.

	TIMIT		MIT	
	FER	CER	FER	CER
SincNet	44,64	1,15	**34,65**	1,27
Sinc MobileNet1D	41,75	**1,01**	43,36	1,44
Sinc AM-MobileNet1D	**41,61**	1,15	41,21	**1,09**

A Tabela 6 mostra os resultados de FER e CER obtidos nas últimas épocas de treinamento do *SincNet* e dos métodos propostos *Sinc MobileNet1D* e *Sinc AM-MobileNet1D* nos experimentos realizados nas bases de dados *TIMIT* e *MIT*. A princípio os métodos propostos mostram uma singela redução do FER calculado na base de dados *TIMIT*. Na mesma base de dados, o CER calculado para o *SincNet* e para o *Sinc AM-MobileNet1D* com $m = 0,5$ se mostram iguais, já o calculado para o *Sinc MobileNet1D* obteve uma pequena redução do erro. Na base de dados *MIT*, o *SincNet* mostrou melhores resultados de FER que ambos os métodos propostos. Por outro lado, o CER calculado pelo *SincNet* ficou entre os resultados calculados pelos métodos propostos, estando o *Sinc AM-MobileNet1D* com o menor (melhor) erro.

Para entender melhor os resultados de FER obtidos na duas bases de dados utilizando os métodos dispostos na Tabela 6, a Figura 36 apresenta a evolução do FER calculado com cada um dos modelos durante as épocas de treinamento. Nela é possível ver que apesar da *SincNet* apresentar uma melhor curva de aprendizagem que ambos os modelos propostos na base de dados *TIMIT*, no final do treinamento, todos convergem para o mesmo lugar, e, ao contrário do que os resultados da Tabela 6 sugerem, a diferença entre o FER do *SincNet* e dos métodos propostos aparenta ser irrelevante dado a variação do erro calculado em cada época. Por outro lado, considerando a evolução do FER calculado na base de dados *MIT*, o *SincNet* obteve melhores resultados mostrando um erro menor que os calculados nos métodos pro-

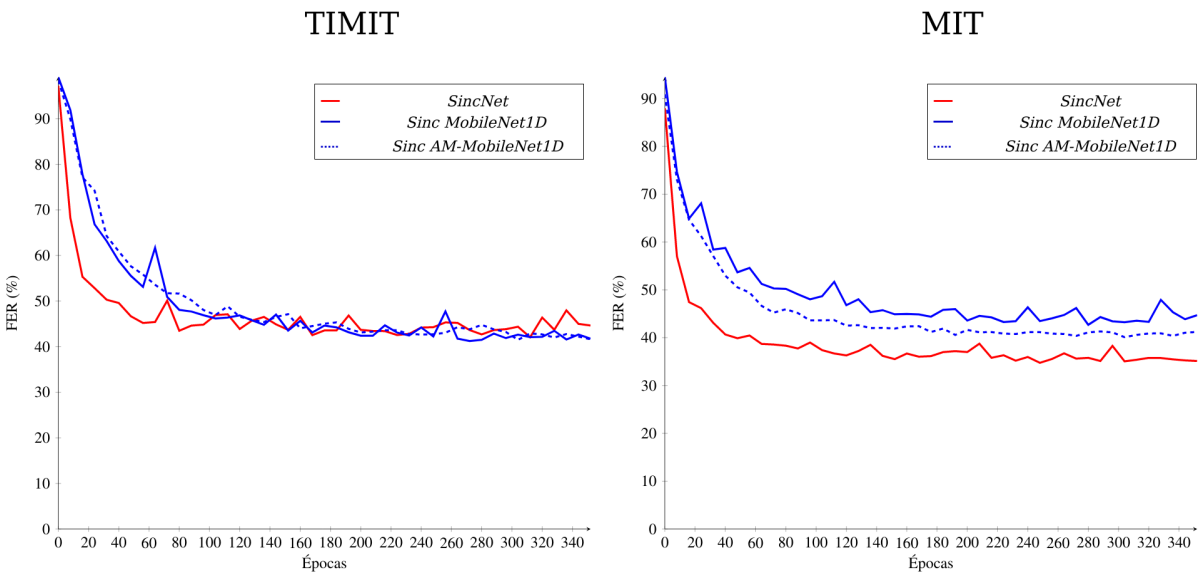


Figura 36 – Análise da evolução do *Frame Error Rate* (%) durante as épocas de treinamento com os métodos *SincNet*, *Sinc MobileNet1D* e *Sinc AM-MobileNet1D* nas bases de dados *TIMIT* e *MIT*.

postos. Ainda, o *Sinc AM-MobileNet1D* apresentou um erro menor do que o *Sinc MobileNet1D* para essa base de dados.

6.4 *GATED ACTIVATION NET*

O método proposto *Gated Activation Net*, descrito em detalhes na Seção 4.4, é um modelo de rede neural para reconhecimento de locutor inspirado na *SwishNet* e suas operações de *Gated Activation* (GA). Nesta seção serão apresentados os resultados de FER e CER obtidos nos experimentos realizados nas bases de dados *TIMIT* e *MIT*. Assim como os modelos das seções anteriores, o *Gated Activation Net* também foi testado com a função de ativação melhorada *Additive Margin Softmax* (AM-Softmax), utilizando $m = 0,5$.

Tabela 7 – Comparação dos resultados do *SincNet* com o modelo proposto, *Gated Activation Net*, nas bases de dados *TIMIT* e *MIT*.

	TIMIT		MIT	
	FER	CER	FER	CER
SincNet	44,64	**1,15**	**34,65**	**1,27**
Gated Activation Net (*Softmax*)	**40,75**	2,30	36,26	4,39
Gated Activation Net (AM-Softmax $m = 0,5$)	55,08	18,25	35,69	3,35

A Tabela 7 apresenta os resultados dos experimentos realizados com o *SincNet*

e o método proposto *Gated Activation Net*. Nela é possível ver os resultados de FER e CER nas bases de dados *TIMIT* e *MIT*. Para os experimentos realizados com o método proposto, foram testadas duas funções de ativação diferentes: a *Softmax* e a AM-Softmax. Os resultados refletem o erro calculado na última época de treinamento de cada modelo, sendo possível ver neles que para a base de dados *TIMIT* o *Gated Activation Net* com a *Softmax* obteve o menor (melhor) FER, embora tenha ficado próximo do calculado para o *SincNet*. Na mesma base de dados, o *SincNet* apresentou menor (melhor) CER que ambas as variações do *Gated Activation Net*. É importante destacar que os resultados de FER e CER calculados na base de dados *TIMIT* para o método proposto com AM-Softmax ficaram consideravelmente maiores (piores) que os calculados com os demais modelos desta seção, porém, isso não mostra que esse modelo tenha sido necessariamente pior que os demais nessa base de dados. Esse resultado calculado mostra apenas que o *Gated Activation Net* com AM-Softmax acabou o treinamento em um momento de pico, como pode ser visto melhor na Figura 37. Os resultados obtidos na base de dados *MIT* mostram que a diferença do FER calculado no final do treinamento de ambos os modelos é miníma, embora o *SincNet* tenha obtido o melhor resultado. Já para o CER calculado na mesma base de dados a diferença é mais significativa, com o *SincNet* mantendo o melhor resultado.

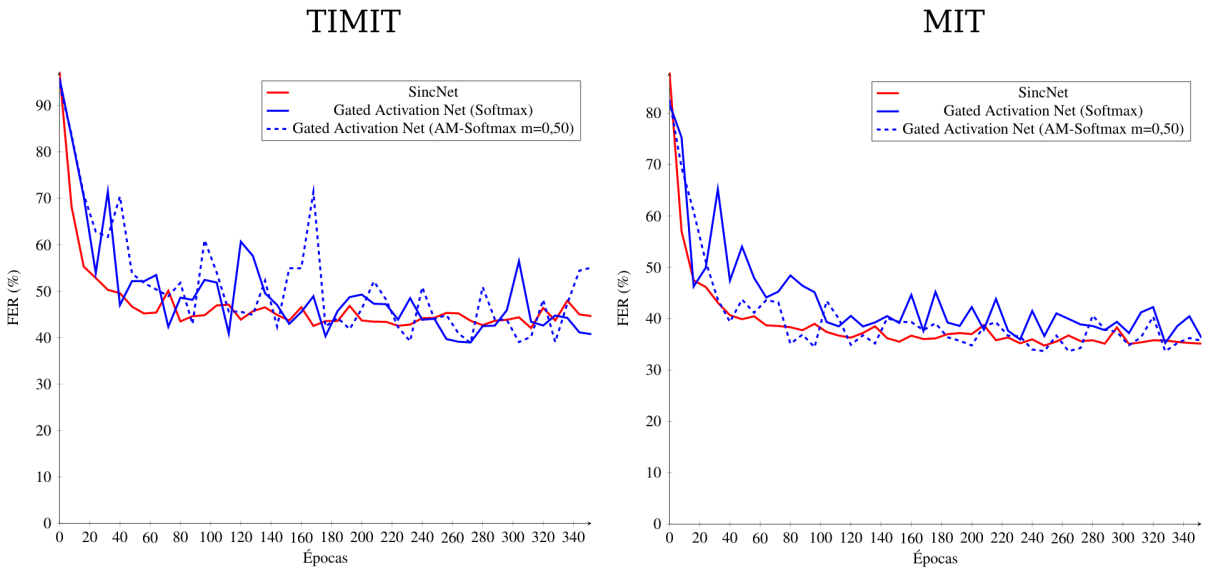


Figura 37 – Análise da evolução do *Frame Error Rate* (%) durante as épocas de treinamento com os métodos *SincNet* e *Gated Activation Net* nas bases de dados *TIMIT* e *MIT*.

A Figura 37, diferente da Tabela 7, oferece uma análise evolutiva do FER dos modelos observados nesta seção durante as épocas de treinamento. Nela é possível ver que ambas as variações do modelo proposto tem uma oscilação considerável

do FER durante as épocas de treinamento quando comparado com o *SincNet*. Essa oscilação é vista com maior clareza na base de dados *TIMIT* na qual o modelo proposto, em suas duas variações, tem picos de aumento de FER maiores que os vistos na base de dados *MIT*. Em ambas as bases de dados o FER calculado durante as épocas de treinamento para todos os modelos aparenta convergir para o mesmo local, embora os resultados do *SincNet* se mostrem mais estáveis.

6.5 *SINC GATED ACTIVATION NET*

O *Sinc Gated Activation Net* é uma variação do modelo proposto *Gated Activation Net* (Seção 4.4) na qual foi adicionado na entrada do modelo uma camada de convolução *sinc*, assim como foi feito com o *Sinc MobileNet* (Seção 4.3). Desta forma, as amostras de áudio passam primeiro pela camada *sinc* para depois seguirem para o *Gated Activation Net* e continuar o fluxo normal do modelo. Para avaliar a eficiência desse modelo foram realizados testes nas bases de dados *TIMIT* e *MIT*, utilizando as métricas de erro FER e CER. Ainda, foram testadas duas variações do modelo proposto, uma utilizando a *Softmax* como função de ativação e a outra utilizando a AM-Softmax.

Tabela 8 – Comparação dos resultados do *SincNet* com o modelo proposto, *Sinc Gated Activation Net*, nas bases de dados *TIMIT* e *MIT*.

	TIMIT		MIT	
	FER	CER	FER	CER
SincNet	44,64	1,15	**34,65**	1,27
Sinc Gated Activation Net (*Softmax*)	47,30	1,22	38,45	**0,46**
Sinc Gated Activation Net (AM-Softmax $m = 0,5$)	**43,37**	**1,08**	40,49	0,86

A Tabela 8 mostra os resultados de FER e CER para as bases de dados *TIMIT* e *MIT* calculados na última época de treinamento de cada modelo. Nela é possível ver que nos resultados relacionados a base de dados *TIMIT*, o método proposto com AM-Softmax obteve os melhores resultados tanto na taxa de FER quanto na de CER, ficando o *SincNet* com o segundo melhor resultado apesar da diferença de resultados calculados entre os três métodos não ser tão grande. Por outro lado, na base de dados *MIT* o *SincNet* ficou com o melhor resultado de FER enquanto obteve o pior resultado de CER. O método que ficou com o melhor resultado de CER na base de dados *MIT* foi o *Sinc Gated Activation Net* utilizando a *Softmax*, seu resultado equivale a um pouco mais da metade do resultado do segundo lugar,

o *Sinc Gated Activation Net* com AM-Softmax, e a um pouco mais que um terço do CER do *SincNet*.

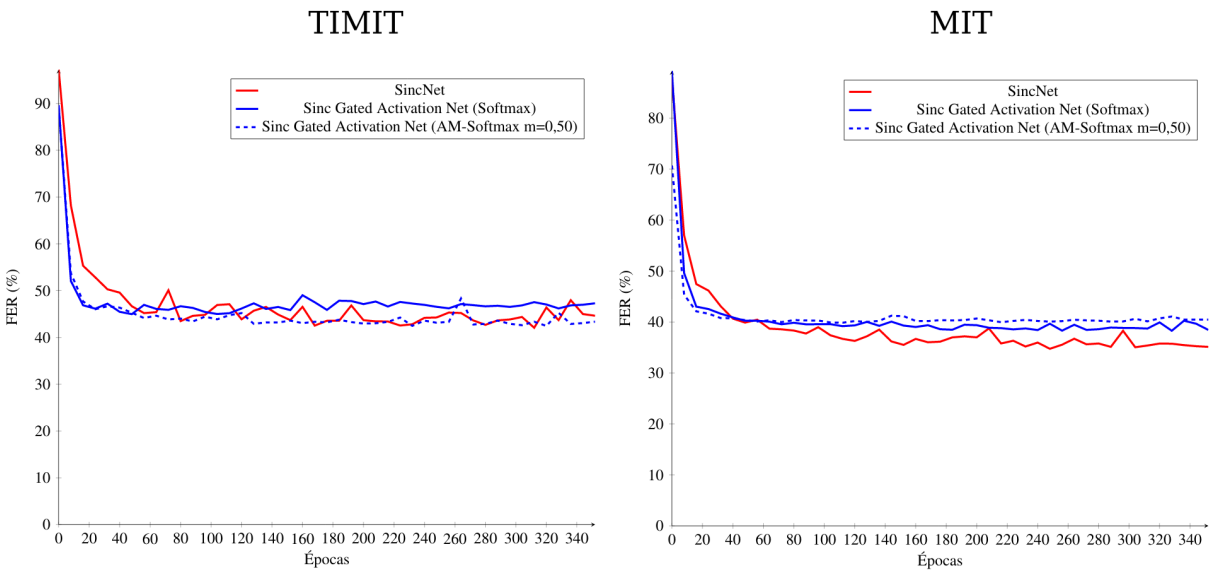


Figura 38 – Análise da evolução do *Frame Error Rate* (%) durante as épocas de treinamento com os métodos *SincNet* e *Sinc Gated Activation Net* nas bases de dados *TIMIT* e *MIT*.

A Figura 38 apresenta a evolução do FER durante as épocas de treinamento para cada um dos modelos nas bases de dados *TIMIT* e *MIT*. Nela é possível ver que a diferença do FER entre os modelos nas duas bases de dados é pequena tornando o *SincNet* e o *Gated Activation Net* com AM-Softmax praticamente equivalentes em termos de FER na base de dados *TIMIT*, enquanto o *Gated Activation Net* com *Softmax* obteve resultados de FER um pouco inferior na mesma base de dados. Na base de dados *MIT* o método proposto obteve resultados muito próximos em suas duas variações, enquanto o *SincNet* mostrou resultados um pouco melhores. Um fato importante a se notar na Figura 38 é que o uso da camada *sinc* no método proposto o tornou mais estável nas duas bases de dados, deixando o modelo mais confiável.

6.6 COMPARAÇÃO DOS MODELOS

Esta seção oferece uma análise comparativa dos resultados obtidos nos experimentos realizados com os modelos descritos no Capítulo 4 e o *SincNet* (Seção 3.2). A comparação é feita com base nos resultados das seções anteriores que mostram as métricas de *Frame Error Rate* (FER) e *Classification Error Rate* (CER) calculadas nos experimentos realizados nas bases de dados *TIMIT* e *MIT*. Além disso, será analisado o tempo de inferência, o número de parâmetros e o tamanho de cada mo-

delo para verificar o impacto das alterações feitas, como a adição das camadas *sinc* e AM-Softmax. O tempo de execução apresentado aqui é referente ao tempo médio de execução de 6.561 *batches* de tamanho 128 na máquina com as configurações listadas na Tabela 1, como foi explicado na Seção 5.4.

A Tabela 9 apresenta os resultados de FER e CER calculados nas bases de dados *TIMIT* e *MIT*, além do tempo de execução (milissegundos), o número de parâmetros em Milhão e o tamanho em *megabytes* de cada modelo, nela os melhores resultados em cada métrica foram destacados em negrito. O tempo de inferência vem na forma *tempo* $\pm$ *desvio-padrão*. Com base nela e nos gráficos da evolução do FER visto nas seções anteriores é possível notar que o uso da função melhorada *Additive Margin Softmax* (AM-Softmax) quando não melhorou os resultados de erro observados em cada modelo, os manteve em um valor muito próximo aos constatados com o uso da *Softmax*. Além da melhora ou equivalência dos resultados obtidos com o uso da AM-Softmax em comparação a *Softmax* tradicional, foi visto que seu uso não mostrou impacto negativo relevante no tempo de inferência, número de parâmetros e tamanho dos modelos.

A Figura 39 apresenta o resultado do tempo de inferência dos modelos, na qual é possível ver que o uso da AM-Softmax nos modelos propostos não prejudica o tempo de inferência dos modelos. Desta forma, fica claro o benefício da AM-Softmax nos experimentos realizados com as bases de dados *TIMIT* e *MIT* e os modelos para reconhecimento de locutor propostos nesse trabalho.

Tabela 9 – Comparação dos resultados do *SincNet* com os modelos propostos no Capítulo 4 nas bases de dados *TIMIT* e *MIT* utilizando as métricas de FER, CER e Tempo em milissegundo.

	TIMIT		MIT		Tempo de Inferência	Parâmetros	Tamanho
	FER	CER	FER	CER	(ms)	(M)	(MB)
SincNet	44,64	1,15	34,65	1,27	42,60 ± 1,5766	22,7	91,2
AM-SincNet ($m = 0,5$)	27,86	0,50	**30,30**	0,52	41,17 ± 0,7529	22,7	91,2
MobileNet1D	26,50	0,57	38,75	2,54	5,88 ± 0,1584	**2,8**	**11,6**
AM-MobileNet1D ($m = 0,5$)	**21,30**	**0,43**	35,55	2,19	5,85 ± 0,1673	**2,8**	**11,6**
Sinc MobileNet1D	41,75	1,01	43,36	1,44	46,44 ± 1,0075	3,0	12,6
Sinc AM-MobileNet1D ($m = 0,5$)	41,61	1,15	41,21	1,09	46,84 ± 0,8295	3,0	12,6
Gated Activation Net (*Softmax*)	40,75	2,30	36,26	4,39	**4,13 ± 0,3630**	25,6	102,9
Gated Activation Net (AM-Softmax $m = 0,5$)	55,08	18,25	35,69	3,35	4,15 ± 0,1700	25,6	102,9
Sinc Gated Activation Net (*Softmax*)	47,30	1,22	38,45	**0,46**	47,43 ± 1,1598	25,9	103,9
Sinc Gated Activation Net (AM-Softmax $m = 0,5$)	43,37	1,08	40,49	0,86	47,02 ± 7,5399	25,9	103,9

Em contrapartida ao uso da AM-Softmax que melhorou os resultados de erro de cada modelo ou os deixou equivalentes aos encontrados com o uso da *Softmax* tradicional, o uso da camada *sinc* não se mostrou tão promissor em todas as bases

de dados. De fato, o uso da camada *sinc* conseguiu melhorar o resultado de CER dos modelos na base de dados *MIT*, enquanto manteve ou piorou o CER calculado na base de dados *TIMIT*. Em relação ao FER, a Figura 40 apresenta a evolução deste erro durante as épocas de treinamento nas bases de dados *TIMIT* e *MIT* para o modelo proposto *MobileNet1D* nas suas quatro variações: *MobileNet1D*, *AM-MobileNet1D*, *Sinc MobileNet1D* e *Sinc AM-MobileNet1D*. Os resultados sem o uso da camada *sinc* estão em vermelho, enquanto os com o uso da camada *sinc* estão em azul. Nesta figura é possível ver que o uso da camada *sinc* teve um impacto negativo no desenvolvimento dos modelos durante as épocas de treinamento. Esse impacto é mais sútil na base de dados *MIT*, uma vez que o CER disposto na Tabela 9 é reduzido com o uso das camadas *sinc*.

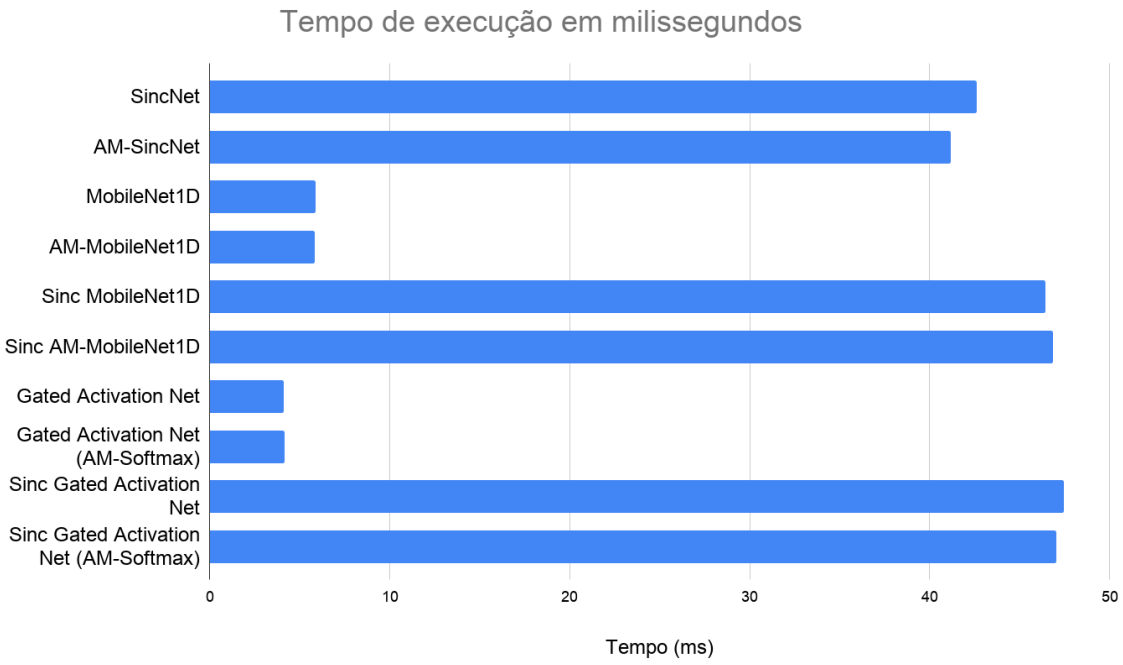


Figura 39 – Análise comparativa do tempo de execução dos modelos em milissegundo.

A mesma análise da evolução do FER com e sem o uso das camadas *sinc* pode ser vista na Figura 41 para as variações do modelo proposto *Gated Activation Net*. Assim como na Figura 40, os resultados utilizando a camada *sinc* estão plotados na cor azul, enquanto os sem o uso da camada *sinc* estão em vermelho. Ao contrário do que aconteceu com o *MobileNet1D*, para o método proposto *Gated Activation Net* o uso da camada *sinc* melhorou os resultados de CER em ambas as bases de dados, como pode ser visto na Tabela 9. Ainda, como demonstrado na Figura 41, o uso da camada *sinc* estabilizou o FER calculado $\pm$ na mesma média de valores obtidos sem o uso dela. O grande ponto negativo observado em todos os modelos testados que

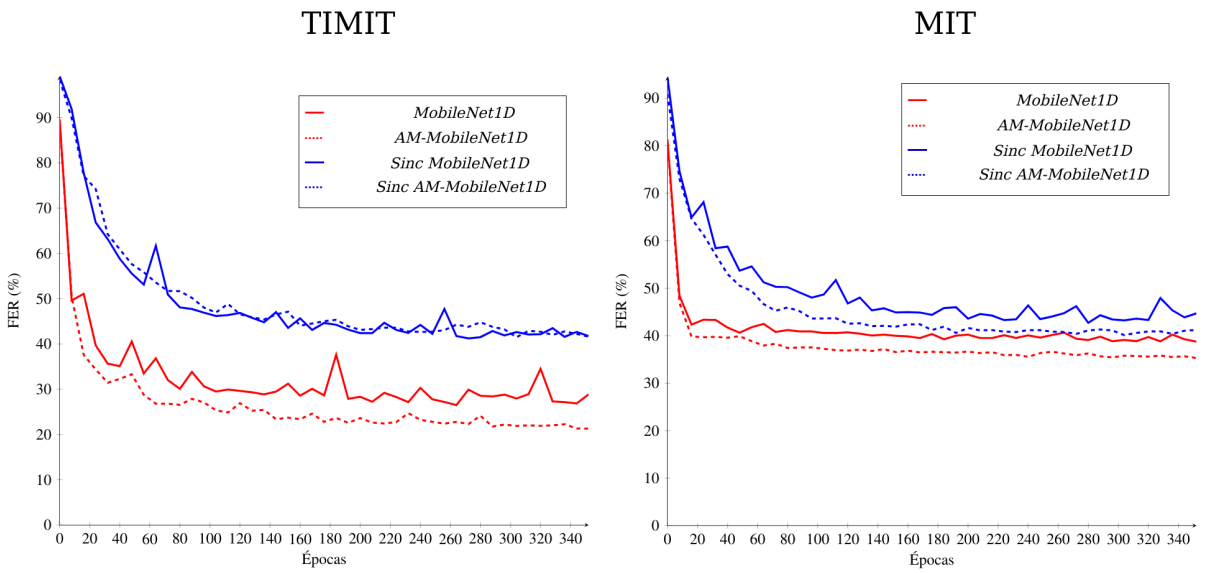


Figura 40 – Análise da evolução do *Frame Error Rate* (%) durante as épocas de treinamento com os modelos *MobileNet* e *Sinc MobileNet*.

fazem uso da camada *sinc* é o aumento no tempo de execução do modelo quando comparado aos que não usam a camada *sinc*. Essa diferença pode ser vista melhor na Figura 39 na qual o tempo de execução dos modelos que se utilizam da camada *sinc* é, em média, 9 vezes maior do que nos modelos que não usam.

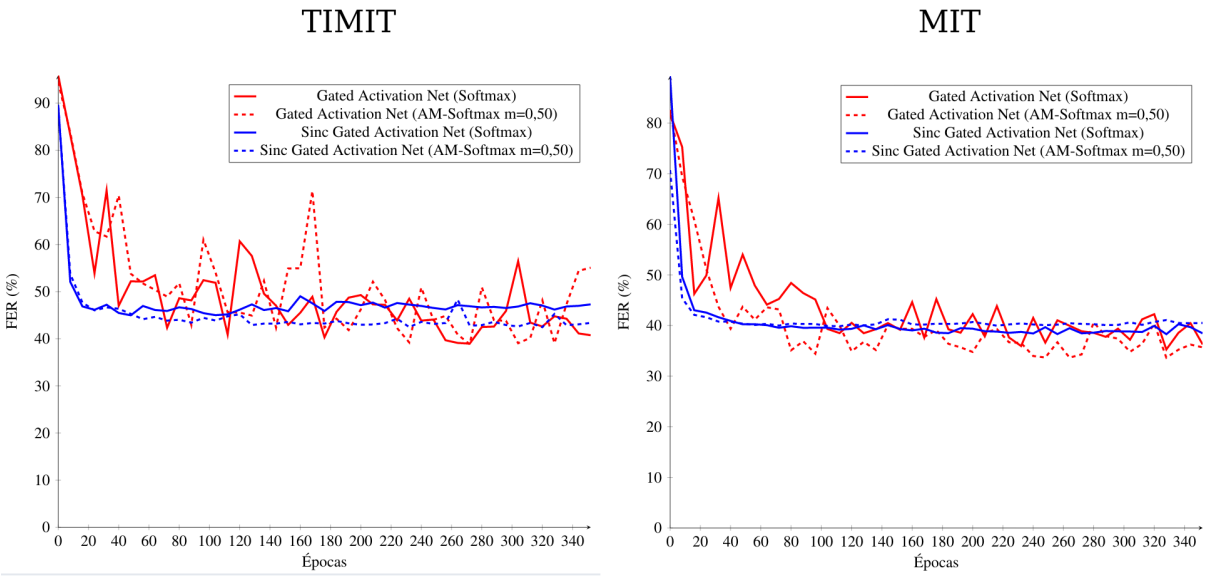


Figura 41 – Análise da evolução do *Frame Error Rate* (%) durante as épocas de treinamento com os modelos *Gated Activation Net* e *Sinc Gated Activation Net*.

Fazendo uma avaliação comparativa dos modelos propostos com o modelo base *SincNet*, é possível ver que na base de dados *TIMIT* em relação às medidas de erro todos os modelos propostos tiveram performance melhor ou equivalente ao *Sinc-*

Net. Embora isso não fique claro na Tabela 9 para o modelo *Gated Activation Net*, a Figura 37 mostra que apesar da oscilação de FER do método proposto, pode se dizer que seu resultado médio é equivalente ao do *SincNet*. Ainda vale considerar que o modelo que se saiu melhor nessa base de dados foi o *AM-MobileNet1D* seguido pelo AM-SincNet, seus resultados de FER e CER equivalem $\pm$ a metade do obtido com o *SincNet* tradicional. Por outro lado, na base de dados *MIT*, os modelos que fazem o uso da camada *sinc* tiveram resultados melhores ou equivalentes ao obtido pelo *SincNet*, ficando o *MobileNet1D* e o *Gated Activation Net* com resultados inferiores ao modelo base. Nessa base de dados, os modelos que tiveram os melhores resultados foram o AM-SincNet e o *Sinc Gated Activation Net* que tiveram resultado de CER menor que a metade calculado com o *SincNet*.

6.7 CONSIDERAÇÕES FINAIS

Em termos gerais, o método proposto *Additive Margin SincNet* (AM-SincNet) foi o que obteve melhores resultados de FER e CER conseguindo diminuir o erro calculado no *SincNet* de forma significativa em ambas as bases de dados. Além disso, o método proposto AM-SincNet não mostrou aumento significativo de complexidade quando comparado ao *SincNet* tradicional, mantendo seu tempo de execução equivalente ao do *SincNet* como pode Figura 39. Ainda, dependendo do contexto no qual a aplicação vai rodar, o método proposto *AM-MobileNet1D* pode ser uma boa escolha. Ele conseguiu diminuir significativamente o erro na base de dados *TIMIT* em relação ao *SincNet*, com uma evolução considerável na base de dados *MIT*. De fato, a grande vantagem do *AM-MobileNet1D* é a redução no tempo de processamento dos *frames*, onde ficou em torno de 7 vezes mais rápido que o método base *SincNet*, além do tamanho do modelo que chega a ser em média 8 vezes menor que o *SincNet*, ocupando apenas $11,6$ *megabytes* em disco, o que o torna uma alternativa para rodar em aparelhos *mobile*.

7 CONCLUSÕES

Este trabalho propôs modelos de *Deep Learning* para a tarefa de reconhecimento de locutor. O objetivo do trabalho foi propor métodos de *deep learning* com desempenho melhor que o obtidos pelo modelo base proposto por (Ravanelli; Bengio, 2018), o *SincNet* (Seção 3.2). Para isso, foram propostos diferentes modelos, os quais foram avaliados nas bases de dados *TIMIT* e *MIT* (Seção 5.1) utilizando as métricas de erro *Frame Error Rate* (FER) e *Classification Error Rate* (CER), além do tempo de inferência, número de parâmetros e tamanho de cada modelo. Dentre os modelos propostos no Capítulo 4 se destacaram o *Additive Margin SincNet* (AM-SincNet) (Seção 4.1) e o AM-MobileNet1D (Seção 4.2).

O método proposto *Additive Margin SincNet* (AM-SincNet) (Chagas Nunes; Macêdo; Zanchettin, 2019) é a junção do *SincNet* proposto por (Ravanelli; Bengio, 2018), que é uma rede neural que se faz uso de uma camada de convolução com funções *sinc* (Subseção 2.2.1), e a *Additive Margin Softmax* (AM-Softmax) (Seção 3.3) proposta por (WANG et al., 2018), que é uma versão da função *Softmax* com uma margem aditiva para separação das classes. (WANG et al., 2018) primeiro propôs a AM-Softmax para o problema de reconhecimento facial, porém, neste trabalho foi verificado que a AM-Softmax também oferece melhoras para o problema de reconhecimento de locutor. O AM-SincNet conseguiu resultados de FER e CER que superam o tradicional *SincNet* em ambas as bases de dados avaliadas, *TIMIT* e *MIT*, sem aumentar a complexidade do modelo, mantendo seu tempo de inferência compatível com o medido na *SincNet*, como pode ser visto na Figura 39.

O método proposto AM-MobileNet1D, descrito na Seção 4.2, é uma adaptação do *MobileNet V2* (Sandler et al., 2018) para tratar sinais de áudio. Nele as operações de convolução foram modificadas para se adaptarem a dimensionalidade dos sinais de áudio, o que diminuiu um pouco a complexidade do método proposto em relação a *MobileNet V2*. O AM-MobileNet1D conseguiu diminuir os resultados de FER e CER em mais da metade em relação ao *SincNet* na base de dados *TIMIT*. Ainda, na base de dados *MIT* o método proposto conseguiu resultados de FER equivalentes ao modelo base enquanto teve um aumento de menos de 1% no CER calculado. Apesar dos bons resultados de FER e CER obtidos pelo AM-MobileNet1D nas bases de dados *TIMIT* e *MIT*, a principal vantagem do método proposto é seu tempo de inferência que conseguiu ser em média 7 vezes mais rápido que o método base, *SincNet*, além da redução no número de parâmetros e no tamanho do modelo que conseguiu ser 8 vezes menor que o *SincNet*.

Além dos modelos propostos nesse trabalho, também foram realizados experimentos para verificar a influência do uso da *Additive Margin Softmax* (AM-Softmax) nos modelos de reconhecimento de locutor, assim como a utilização das camadas *sinc* do *SincNet*. O AM-Softmax introduz 2 novos parâmetros em comparação à camada *Softmax* tradicional, são eles: o parâmetro s para controlar a escala das amostras e o parâmetro m para controlar o tamanho da margem aditiva. Nos experimentos realizados neste trabalho, o parâmetro s foi configurado para um valor fixo $s = 30$, assim como proposto em (WANG et al., 2018). Por outro lado, com o parâmetro m que controla o tamanho da margem aditiva, foram realizados diversos experimentos com o modelo proposto AM-SincNet nas bases de dados *TIMIT* e *MIT* variando m no intervalo $0,05 \leq m \leq 0,95$, os resultados desses experimentos levam a crer que não existe um valor de m que maximize os resultados, pois para todos os valores de m testados os modelos acabaram convergindo para o mesmo limiar de erro. Em relação a utilização da AM-Softmax nos modelos de reconhecimento de locutor, para todos os experimentos realizados com os modelos propostos no Capítulo 4, o uso da AM-Softmax melhorou os resultados dos modelos ou os manteve equivalentes ao obtido sem o uso da AM-Softmax. Para os experimentos realizados para testar a influência da camada *sinc* nos modelos, foi verificado que o uso dela melhorou os resultados de CER dos modelos na base de dados *MIT*, conseguindo ainda estabilizar os resultados do modelo proposto *Gated Activation Net* em ambas as bases de dados. Também foi verificado que o uso da camada *sinc* aumentou em até 11 vezes o tempo de inferência dos modelos como pode ser visto na Figura 39.

Dentre os resultados expostos no Capítulo 6, aqueles referentes ao método proposto *Additive Margin SincNet* (AM-SincNet) (Seção 6.1) foram parcialmente publicados na *International Joint Conference on Neural Networks (IJCNN)* sob o nome *"Additive Margin SincNet for Speaker Recognition"*, (Chagas Nunes; Macêdo; Zanchettin, 2019).

7.1 TRABALHOS FUTUROS

Como trabalho futuro pretende-se testar os modelos propostos em outras bases de dados, como por exemplo na *VoxCeleb2* (CHUNG; NAGRANI; ZISSERMAN, 2018) que tem mais de 1 milhão de amostras divididas em torno de 6 mil locutores. Valendo ressaltar que seria interessante investigar melhor o motivo da oscilação do erro no método proposto *Gated Activation Net*, uma vez que ele mostrou ser promissor quanto ao tempo de execução. Outra ação importante seria a otimização da implementação das camadas de convolução *sinc* feitas por (Ravanelli; Bengio, 2018), dado que foi constatado o grande aumento no tempo de execução dos modelos que se

utilizavam dela. Além disso, é também interessante adaptar os métodos para o problema de verificação de locutor, uma vez que foram testados apenas no problema de identificação de locutor. Também é válido testar outras funções de erro derivadas da *Softmax*.

REFERÊNCIAS

Albawi, S.; Mohammed, T. A.; Al-Zawi, S. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*. [S.l.: s.n.], 2017. p. 1–6. ISSN null.

ALOM, M. Z.; TAHA, T. M.; YAKOPCIC, C.; WESTBERG, S.; SIDIKE, P.; NASRIN, M. S.; HASAN, M.; ESSEN, B. C. V.; AWWAL, A.; ASARI, V. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, v. 8, p. 292, 03 2019.

BA, J.; KIROS, J. R.; HINTON, G. E. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Bansal, P.; Imam, S. A.; Bharti, R. Speaker recognition using mfcc, shifted mfcc with vector quantization and fuzzy. In: *2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)*. [S.l.: s.n.], 2015. p. 41–44. ISSN null.

BEIGI, H. *Fundamentals of Speaker Recognition*. [S.l.]: Springer Publishing Company, Incorporated, 2011. ISBN 0387775919, 9780387775913.

BENGIO, Y. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, Now Publishers Inc., Hanover, MA, USA, v. 2, n. 1, p. 1–127, jan. 2009. ISSN 1935-8237. Disponível em: <http://dx.doi.org/10.1561/2200000006>.

CAMPBELL, W.; STURIM, D.; REYNOLDS, D.; SOLOMONOFF, A. Svm based speaker verification using a gmm supervector kernel and nap variability compensation. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2006*, v. 1, p. I – I, 06 2006.

Chagas Nunes, J. A.; Macêdo, D.; Zanchettin, C. Additive margin sincnet for speaker recognition. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2019. p. 1–5.

CHUNG, J. S.; NAGRANI, A.; ZISSERMAN, A. Voxceleb2: Deep speaker recognition. *Interspeech 2018*, ISCA, Sep 2018. Disponível em: <http://dx.doi.org/10.21437/Interspeech.2018-1929>.

CUMANI, S.; PLCHOT, O.; LAFACE, P. Probabilistic linear discriminant analysis of i-vector posterior distributions. In: *ICASSP*. IEEE, 2013. p. 7644–7648. Disponível em: <http://dblp.uni-trier.de/db/conf/icassp/icassp2013.html#CumaniPL13>.

CUMANI, S.; PLCHOT, O.; LAFACE, P. Probabilistic linear discriminant analysis of i-vector posterior distributions. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. [S.l.: s.n.], 2013. p. 7644–7648. ISSN 1520-6149.

DASH, K.; PADHI, D.; PANDA, B.; MOHANTY, S. Speaker identification using mel frequency cepstral coefficient and bpnn. 04 2012.

DEHAK, N.; KENNY, P. J.; DEHAK, R.; DUMOUCHEL, P.; OUELLET, P. Front end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech and Language Processing*, 2010.

Dehak, N.; Kenny, P. J.; Dehak, R.; Dumouchel, P.; Ouellet, P. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, v. 19, n. 4, p. 788–798, May 2011. ISSN 1558-7924.

FAN, C. Survey of convolutional neural network. In: . [S.l.: s.n.], 2016.

GAROFOLO, J. S.; LAMEL, L. F.; FISHER, W. M.; FISCUS, J. G.; PALLETT, D. S.; DAHLGREN, N. L. *DARPA TIMIT Acoustic Phonetic Continuous Speech Corpus CDROM*. [S.l.]: NIST, 1993.

GE, Z.; IYER, A. N.; CHELUVARAJA, S.; SUNDARAM, R.; GANAPATHIRAJU, A. Neural network based speaker classification and verification systems with enhanced features. *2017 Intelligent Systems Conference (IntelliSys)*, p. 1089–1094, 2017.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

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.], 2016. p. 770–778. ISSN 1063-6919.

HEIGOLD, G.; MORENO, I.; BENGIO, S.; SHAZEER, N. End-to-end text-dependent speaker verification. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Mar 2016. Disponível em: <http://dx.doi.org/10.1109/ICASSP.2016.7472652>.

HINTON, G. E.; SRIVASTAVA, N.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, 2012.

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*. 2017.

HUSSAIN, M. S.; HAQUE, M. A. Swishnet: A fast convolutional neural network for speech, music and noise classification and segmentation. *CoRR*, abs/1812.00149, 2018. Disponível em: <http://arxiv.org/abs/1812.00149>.

IANDOLA, F. N.; HAN, S.; MOSKEWICZ, M. W.; ASHRAF, K.; DALLY, W. J.; KEUTZER, K. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size*. 2016.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: BACH, F.; BLEI, D. (Ed.). *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: PMLR, 2015. (Proceedings of Machine Learning Research, v. 37), p. 448–456. Disponível em: <http://proceedings.mlr.press/v37/ioffe15.html>.

JIN, J.; DUNDAR, A.; CULURCIELLO, E. *Flattened Convolutional Neural Networks for Feedforward Acceleration*. 2014.

JUNG, J.-W.; HEO, H.-S.; YANG, I.-H.; SHIM, H.-j.; YU, H.-J. A complete end-to-end speaker verification system using deep neural networks: From raw signals to verification result. In: . [S.l.: s.n.], 2018.

KENNY, P.; BOULIANNE, G.; OUELLET, P.; DUMOUCHEL, P. Joint factor analysis versus eigenchannels in speaker recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, v. 15, p. 1435 – 1447, 06 2007.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. USA: Curran Associates Inc., 2012. (NIPS'12), p. 1097–1105. Disponível em: <http://dl.acm.org/citation.cfm?id=2999134.2999257>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. USA: Curran Associates Inc., 2012. (NIPS'12), p. 1097–1105. Disponível em: <http://dl.acm.org/citation.cfm?id=2999134.2999257>.

LECUN, Y.; BENGIO, Y. The handbook of brain theory and neural networks. In: ARBIB, M. A. (Ed.). Cambridge, MA, USA: MIT Press, 1998. cap. Convolutional Networks for Images, Speech, and Time Series, p. 255–258. ISBN 0-262-51102-9. Disponível em: <http://dl.acm.org/citation.cfm?id=303568.303704>.

LOGAN, B. Mel frequency cepstral coefficients for music modeling. In: *In International Symposium on Music Information Retrieval*. [S.l.: s.n.], 2000.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. [S.l.: s.n.], 2013.

MATEJKA, P.; GLEMBEK, O.; CASTALDO, F.; ALAM, M. J.; PLCHOT, O.; KENNY, P.; BURGET, L.; CERNOCKÝ, J. Full-covariance ubm and heavy-tailed plda in i-vector speaker verification. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, p. 4828–4831, 05 2011.

MCLAREN, M.; LEI, Y.; FERRER, L. Advances in deep neural network approaches to speaker recognition. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2015. p. 4814–4818. ISSN 1520-6149.

Muckenhirn, H.; Magimai.-Doss, M.; Marcell, S. Towards directly modeling raw speech signal for speaker verification using cnns. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2018. p. 4884–4888. ISSN 2379-190X.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: FüRNKRANZ, J.; JOACHIMS, T. (Ed.). *ICML*. Omnipress, 2010. p.

807–814. Disponível em: <http://dblp.uni-trier.de/db/conf/icml/icml2010.html#NairH10>.

OORD, A. van den; DIELEMAN, S.; ZEN, H.; SIMONYAN, K.; VINYALS, O.; GRAVES, A.; KALCHBRENNER, N.; SENIOR, A.; KAVUKCUOGLU, K. *WaveNet: A Generative Model for Raw Audio*. 2016.

OORD, A. van den; KALCHBRENNER, N.; VINYALS, O.; ESPEHOLT, L.; GRAVES, A.; KAVUKCUOGLU, K. *Conditional Image Generation with PixelCNN Decoders*. 2016.

PEEMEN, M.; MESMAN, B.; CORPORAAL, H. Speed sign detection and recognition by convolutional neural networks. In: *Proceedings of the 8th International Automotive Congress (IAC 2011), 16-17 May 2011, Eindhoven, The Netherlands*. [S.l.: s.n.], 2011. p. 162–170.

PRINCE, S.; ELDER, J. H. Probabilistic linear discriminant analysis for inferences about identity. *2007 IEEE 11th International Conference on Computer Vision*, p. 1–8, 2007.

RABINER, L.; SCHAFER, R. *Theory and Applications of Digital Speech Processing*. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN 0136034284, 9780136034285.

RASTEGARI, M.; ORDONEZ, V.; REDMON, J.; FARHADI, A. Xnor-net: Imagenet classification using binary convolutional neural networks. *Lecture Notes in Computer Science*, Springer International Publishing, p. 525–542, 2016. ISSN 1611-3349. Disponível em: <http://dx.doi.org/10.1007/978-3-319-46493-0_32>.

Ravanelli, M.; Bengio, Y. Speaker recognition from raw waveform with sincnet. In: *2018 IEEE Spoken Language Technology Workshop (SLT)*. [S.l.: s.n.], 2018. p. 1021–1028.

RAVANELLI, M.; BRAKEL, P.; OMOLOGO, M.; BENGIO, Y. A network of deep neural networks for distant speech recognition. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 4880–4884, 2017.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 39, 06 2015.

RICHARDSON, F.; REYNOLDS, D.; DEHAK, N. Deep neural network approaches to speaker and language recognition. *IEEE Signal Processing Letters*, v. 22, p. 1–1, 10 2015.

Sadewa, R. A.; Wirayuda, T. A. B.; Sa'adah, S. Speaker recognition implementation for authentication using filtered mfcc — vq and a thresholding method. In: *2015 3rd International Conference on Information and Communication Technology (ICoICT)*. [S.l.: s.n.], 2015. p. 261–265. ISSN null.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2018. p. 4510–4520.

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

SNYDER, D.; CHEN, G.; POVEY, D. *MUSAN: A Music, Speech, and Noise Corpus*. 2015.

Snyder, D.; Garcia-Romero, D.; Sell, G.; Povey, D.; Khudanpur, S. X-vectors: Robust dnn embeddings for speaker recognition. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2018. p. 5329–5333. ISSN 2379-190X.

SNYDER, D.; GHAHREMANI, P.; POVEY, D.; GARCIA-ROMERO, D.; CARMIEL, Y.; KHUDANPUR, S. Deep neural network-based speaker embeddings for end-to-end speaker verification. In: *2016 IEEE Spoken Language Technology Workshop (SLT)*. [S.l.: s.n.], 2016. p. 165–170.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, JMLR.org, v. 15, n. 1, p. 1929–1958, jan. 2014. ISSN 1532-4435.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S. E.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. *CoRR*, 2014.

Tang, Z.; Li, L.; Wang, D.; Vipperla, R. Collaborative joint training with multitask recurrent model for speech and speaker recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, v. 25, n. 3, p. 493–504, March 2017. ISSN 2329-9304.

TOMPSON, J.; GOROSHIN, R.; JAIN, A.; LECUN, Y.; BREGLER, C. Efficient object localization using convolutional networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun 2015. Disponível em: <http://dx.doi.org/10.1109/CVPR.2015.7298664>.

Triefenbach, F.; Martens, J. Can non-linear readout nodes enhance the performance of reservoir-based speech recognizers? In: *2011 First International Conference on Informatics and Computational Intelligence*. [S.l.: s.n.], 2011. p. 262–267. ISSN null.

TRIGEORGIS, G.; RINGEVAL, F.; BRUECKNER, R.; MARCHI, E.; NICOLAOU, M. A.; SCHULLER, B.; ZAFEIRIOU, S. Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2016. p. 5200–5204. ISSN 2379-190X.

Tzanetakis, G.; Cook, P. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, v. 10, n. 5, p. 293–302, July 2002.

Vijayan, A.; Mathai, B. M.; Valsalan, K.; Johnson, R. R.; Mathew, L. R.; Gopakumar, K. Throat microphone speech recognition using mfcc. In: *2017 International Conference on Networks Advances in Computational Technologies (NetACT)*. [S.l.: s.n.], 2017. p. 392–395. ISSN null.

WANG, F.; CHENG, J.; LIU, W.; LIU, H. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, v. 25, n. 7, p. 926–930, July 2018. ISSN 1070-9908.

WANG, F.; JIANG, M.; QIAN, C.; YANG, S.; LI, C.; ZHANG, H.; WANG, X.; TANG, X. Residual attention network for image classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul 2017. Disponível em: <http://dx.doi.org/10.1109/CVPR.2017.683>.

Wang, M.; Liu, B.; Foroosh, H. Factorized convolutional neural networks. In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. [S.l.: s.n.], 2017. p. 545–553.

Winursito, A.; Hidayat, R.; Bejo, A. Improvement of mfcc feature extraction accuracy using pca in indonesian speech recognition. In: *2018 International Conference on Information and Communications Technology (ICOIACT)*. [S.l.: s.n.], 2018. p. 379–383. ISSN null.

Woo, R. H.; Park, A.; Hazen, T. J. The mit mobile device speaker verification corpus: Data collection and preliminary experiments. In: *2006 IEEE Odyssey - The Speaker and Language Recognition Workshop*. [S.l.: s.n.], 2006. p. 1–6. ISSN null.

WU, J.; LENG, C.; WANG, Y.; HU, Q.; CHENG, J. Quantized convolutional neural networks for mobile devices. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun 2016. Disponível em: <http://dx.doi.org/10.1109/CVPR.2016.521>.

YANDONG HAO ZONGBO, L. H. L. Survey of convolutional neural network. *Journal of Computer Applications*, Journal of Computer Applications, v. 36, n. 9, p. 2508, 2016.