UNIVERSIDADE FEDERAL DO MARANHÃO

Programa de Pós-Graduação em Ciência da Computação

Mateus Barros Frota de Carvalho

RECONHECIMENTO AUTOMÁTICO DE FONEMAS VIA RNA PROFUNDA

São Luís
2020

MATEUS BARROS FROTA DE CARVALHO

RECONHECIMENTO AUTOMÁTICO DE FONEMAS VIA RNA PROFUNDA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Areolino de Almeida Neto

São Luís - MA

2020

Mateus Barros Frota de Carvalho

Reconhecimento Automático de Fonemas via RNA Profunda

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 11 de dezembro de 2020

Prof. Dr. Areolino de Almeida Neto
Universidade Federal do Maranhão - UFMA
Orientador

Prof. Dr. Alexandre César Muniz de Oliveira
Universidade Federal do Maranhão - UFMA
Examinador Interno

Prof. Dr. Rogério Moreira Lima Silva
Universidade Estadual do Maranhão - UEMA
Examinador Externo

São Luís - MA
2020

AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer a Deus. Gostaria de agradecer a minha família, em especial a minha mãe, Silvanira de Jesus Barros, e a minha avó, Maria José, que estiveram sempre presentes me apoiando ao longo de toda a minha vida e sempre me fizeram acreditar que a educação é o bem mais precioso que temos em nossas vidas. Obrigado, essa paixão pelo conhecimento hoje é o que me move e faz-me ampliar meus horizontes.

Também gostaria de agradecer ao Prof. Areolino de Almeida Neto, que me auxiliou neste estudo desafiador, sempre acreditou em mim e viabilizou formas para que eu pudesse continuar a desenvolver o trabalho. A sua paciência, experiência e dedicação foram decisivos para o desenvolvimento deste trabalho.

Aos professores do PPGCC-UFMA e da Engenharia de Computação, pela formação proporcionada através das disciplinas, conversas e conselhos. Sem dúvidas, todo o conhecimento passado foi imprescindível para a conclusão desta dissertação. Agradeço a UFMA por ter proporcionado toda a minha formação acadêmica, nos cursos de graduação de Ciência e Tecnologia, Engenharia de Computação e mestrado em Ciência da Computação. Agradeço também ao IFMA por ter me proporcionado uma oportunidade de trabalho neste meio tempo e proporcionado que eu continuasse a desenvolver este projeto.

Agradeço à amizade dos colegas de laboratório e do trabalho. Em muitas situações, vocês foram meu ponto de apoio. Em especial agradeço aos amigos Moisés Rocha, Joaquim Scavone, Antônio Mourão, Rodrigo Garcês, Cláudia Vieira e Beatriz Carvalho.

Quem desiste nunca vence e só vence quem nunca desiste.
(Napoleon Hill)

RESUMO

Este trabalho apresenta um modelo de reconhecimento de fonemas utilizando técnicas de detecção de objetos. Utilizou-se o detector *Single Shot Detection* em conjunto com a arquitetura de rede convolucional *MobileNet*. As bases de dados empregadas para treinar o modelo foram a TIMIT e a *LibriSpeech*, ambas são constituídas por áudios da língua inglesa. Para criar uma representação gráfica dos áudios das bases, para cada amostra de áudio, calculou-se o seu espectrograma na escala de Mel e para treinar o algoritmo de detecção de localização dos fonemas, anotou-se a posição temporal da ocorrência de cada fonema no seu respectivo espectrograma. Adicionalmente, foi necessário aumentar o conjunto de dados de treino, de forma a proporcionar melhora na generalização do modelo e para isso, juntaram-se as duas bases de dados e aplicaram-se técnicas de aumento de dados para áudios. Os resultados deste trabalho ficaram próximos dos resultados obtidos em importantes trabalhos recentemente publicados. Esta pesquisa usou dois modelos com arquiteturas diferentes: a arquitetura $MobileNet - Large$, a qual obteve uma acurácia de 0,72 mAP@0.5IOU e uma taxa de erro por fonema de 19,47% e a arquitetura $MobileNet - Small$, a qual obteve uma acurácia de 0,63 mAP@0.5IOU e taxa de erro por fonema igual a 31,02%.

Palavras-chave: Detecção de objetos, reconhecimento de fala, reconhecimento de fonemas.

ABSTRACT

This work presents a phoneme recognition model using object detection techniques. The Single Shot Detection detector was used in conjunction with the MobileNet convolutional network architecture. The databases used in model training were TIMIT and LibriSpeech, both have spoken audios in English. To generate a graphical representation using the audiobases, for each audio, its spectrogram was calculated on the Mel scale and to train the algorithm of phoneme location detection, the temporal position of the occurrence of each phoneme in its respective was noted for its spectrogram. Additionally, it was necessary to increase the training data set, in order to provide improvement in the generalization of the model and for that, the two databases were joined and data augmentation techniques were applied to audios. The results of this work were close to the results obtained in other state of the art works. This research used two models with different architectures: the MobileNet-Large architecture, which obtained an accuracy of 0.72 mAP@0.5IOU and an error rate per phoneme of 19.47 % and the MobileNet-Small architecture, which obtained an accuracy of 0.63 mAP@0.5IOU and error rate per phoneme equal to 31.02 %.

Keywords: Object detection, voice recognition, phoneme recognition.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

HMM	*Hidden Markov Model*
DBN	*Deep Belief Network*
RNA	*Rede Neural Artificial*
CNN	*Convolutional Neural Network*
MLP	*Perceptrons* de Múltiplas Camadas
MELSPEC	*Mel Filter Bank Coefficients*
MFCC	*Mel Frequency Cepstral Coefficients*
ASR	*Automatic Speech Recognition*
STFT	*Short Time Fourier Transform*
SNR	*Signal Noise Ratio*
mAP	*Mean Average Precision*
AP	*Average Precision*
IoU	*Intersection over Union*
PER	*Phone Error Rate*
SSD	*Single ShotMultiDetector*
YOLO	*You Only Look Once*
VP	Verdadeiros Positivos
VN	Verdadeiros Negativos
FP	Falsos Positivos
FN	Falsos Negativos

SUMÁRIO

1 INTRODUÇÃO

A utilização de computadores tem tornado-se cada vez mais generalizada e a forma como se interage com essa máquina sempre foi uma preocupação no setor da informática. Nas primeiras aplicações computacionais, as interações eram realizadas somente através do teclado, utilizando linhas de comando e blocos de texto. Com o passar dos anos, evoluiu-se para uma interação mais intuitiva, surgiram as interfaces gráficas, as quais proporcionaram maiores facilidades no manuseio das máquinas. Atualmente, é cada vez mais comum a utilização de *softwares* que trabalham diretamente através de comandos por voz.

A aplicabilidade de interagir com a máquina através de comandos de áudio proporciona uma maior facilidade de utilização das máquinas e amplia as possibilidades de uso, acarretando maior acessibilidade para todos, em especial para indivíduos que possuem limitações físicas ou deficiências. Vide os aplicativos de mensagens instantâneas, comumente utilizados, os quais permitem que um usuário com pouca ou nenhuma instrução escolar envie mensagem de voz e comunique-se livremente.

Seguindo essa mesma concepção, uma aplicação ou sistema que opere através de comandos de áudio, possibilitará maior simplicidade durante sua utilização, pois o usuário não precisará executar os comandos diretamente, através da digitação de textos ou de cliques em botões, bastará expressar verbalmente as instruções e a máquina reconhecerá o comando. Essa capacidade de identificar comandos verbais possibilita que o usuário opere o sistema sem precisar interagir visualmente com ele, permitindo interação com o sistema mesmo em situações em que sua atenção esteja direcionada à outra tarefa, como dirigir ou cozinhar.

Do ponto de vista técnico, a identificação da fala por máquinas não é uma tarefa trivial. Os desafios ocorrem principalmente porque os sinais obtidos no processo de produção da fala são muito variados, devido à volumosa quantidade de atributos inerentes à voz humana, bem como às características envolvidas na fala, como sotaques, velocidade da fala, timbre e possíveis ruídos de fundo.

O reconhecimento de padrões é uma habilidade inerente dos seres humanos. Essa habilidade permite identificar características relevantes a determinado elemento ou situação, as quais definem uma classe. Tendo isso em vista, existe uma área de estudos cujo objetivo é estudar e reproduzir essa característica humana em máquinas eletrônicas.

Diversos são os padrões que podem ser reconhecidos. Entre eles, está o reconhecimento de voz. No sistema auditivo humano, o som é captado pelo canal auditivo e passa por um complexo mecanismo de estruturas responsáveis por converter as ondas mecânicas propagadas pelo ar em sinais cerebrais, os quais são processados pelo cérebro. Embora trate-se de um procedimento complexo, o corpo humano executa-o de forma natural e automática.

Contudo, a atividade de reconhecimento não é facilmente reproduzida por uma máquina digital. O reconhecimento de padrões por máquinas envolve técnicas de atribuição de padrões às suas respectivas classificações ou rótulos. Esse processo deve acontecer de forma automática e com a menor intervenção humana possível (GONZALEZ; WOODS, 2009).

A técnica de reconhecimento envolve uma análise estocástica de um conjunto de exemplos e o objetivo deste processo é identificar as distinções entre os conjuntos existentes e gerar um modelo consistente que consiga classificar com eficiência não só exemplos já vistos, mas também exemplos inéditos.

As primeiras pesquisas nessa área objetivaram identificar um sinal sonoro e convertê-lo em dígitos numéricos (FURUI, 1981). A partir desse primeiro marco, diversos modelos foram implementados.

Por volta dos anos 80, algumas linhas de pesquisa começaram a aplicar a teoria probabilística dos modelos ocultos de Markov (HMM - *Hidden Markov Models*, em inglês) em sistema de reconhecimento de fala e deram origem aos primeiros sistemas comerciais de reconhecimento de fala (RABINER, 1989; JUANG; LEVINSON; SONDHI, 1986).

A utilização de HMM em tarefas de reconhecimento de fala culminou no surgimento de uma nova abordagem do problema, a qual combinava modelos acústicos, léxicos e linguísticos utilizando um algoritmo probabilístico. Essa nova abordagem possibilitou o desenvolvimento de sistemas mais robustos, capazes de reconhecer vocabulários mais extensos e independentes de locutor. Por volta dos anos 90, sistemas comerciais popularizaram-se e como exemplos, têm-se o IBM Via Voice e o Dragon Systems (CONIAM, 1999).

Atualmente, uma vasta literatura e diversas aplicações no domínio de reconhecimento de voz têm sido desenvolvidas (HUANG; DENG, 2010; MENG; ZHANG; ZHAO, 2012), tais como serviços de atendimento automático em centrais telefônicas, sistemas de acessibilidade para pessoas com deficiência motora, sistemas de assistência virtual, entre outros variados exemplos.

1.1 Objetivos

O objetivo principal deste trabalho é implementar uma abordagem de identificação de fonemas em palavras utilizando representações de áudio através de reconhecimento de padrões aplicando a técnica de redes neurais profundas.

Os objetivos específicos são:

1. Transformar os dados de formato de áudio em representação visual através do espectrograma;

2. Categorizar os fonemas contidos nas palavras dos enunciados das frases gravadas;
3. Construir um modelo baseado em redes neurais para identificação de fonemas;
4. Comparar o modelo produzido com outros modelos presentes na literatura.

1.2 Justificativas e motivações

A principal justificativa ao pesquisar e desenvolver soluções na área de reconhecimento de voz é possibilitar ao usuário uma comunicação onidirecional com a máquina. Tendo em vista que a interação via áudio não necessita de contato físico ou visual, essa abordagem possibilita uma maior acessibilidade e facilidade na comunicação. Um fator muito importante a considerar-se é a exclusão educacional, pois infelizmente, muitas pessoas não possuem formação suficiente para comunicarem-se através da escrita. Ainda assim, isso não impede sua comunicação com outros indivíduos, que ocorre através da fala ou mesmo através de gestos. Dessa forma, o reconhecimento de voz poderá permitir uma interação homem-máquina, mesmo para pessoas com essas deficiências.

A possibilidade de interação homem-computador através de áudio possibilita muitas vantagens, entre elas, a comunicação mais eficiente, a qual ocorre sem necessidade de contato visual ou físico. Adicionalmente, além de permitir a inclusão digital de pessoas com baixa escolaridade, indivíduos com restrições motoras poderão ter melhor qualidade de vida, a partir da utilização de dispositivos digitais através de comandos de fala.

Atualmente, existem diversas soluções de reconhecimento de fala e de comandos através de voz. Ainda assim, o desenvolvimento de novas tecnologias nessa área, cada vez mais se mostra de extrema importância, haja vista que muitas dessas soluções apresentam muitos erros de tradução de áudio para texto. Outra adversidade tipicamente envolvida nesses serviços é a necessidade de conexão com a internet, uma vez que os seus algoritmos funcionam em servidores em nuvem e necessitam que o cliente esteja conectado à rede para que possa efetuar as solicitações e receber os resultados dos processamentos.

1.3 Trabalhos Relacionados

As primeiras pesquisas relacionadas ao tema de reconhecimento automático de fala iniciaram-se cerca de 70 anos. Nessa ocasião, foi desenvolvido um reconhecedor de voz capaz de decodificar um sinal sonoro e convertê-los para dígitos numéricos (FURUI, 1981). A partir desse marco, diversos modelos foram desenvolvidos. No entanto, apresentavam muitas restrições, por exemplo: vocabulário extremamente limitado, alta sensibilidade em relação à distinção de locutores e a ruídos do ambiente, etc. Por volta dos anos 60, surgiu a teoria probabilística dos modelos ocultos de Markov (*Hidden Markov Models*, HMM), mas só por volta dos anos 80, começou-se a aplicar essa técnica para reconhecimento de

fala. Essa utilização deu origem as primeiras aplicações de sistemas de reconhecimento de fala (RABINER, 1989; JUANG; LEVINSON; SONDHI, 1986).

Até então, os trabalhos desenvolvidos na área de reconhecimento de fala haviam alcançado um certo patamar de desempenho, contudo, não apresentavam melhorias expressivas quando comparados ao modelo estado da arte da época, que era totalmente baseado em métodos probabilísticos. Mais recentemente, por volta de 2012, um grupo de pesquisadores propuseram uma nova abordagem, que consistiu em substituir o modelo acústico, o qual na maioria dos sistemas era representado por um modelo Gaussiano, por um novo modelo baseado em redes neurais profundas (*Deep Neural Networks*, DNNs, em inglês). Esse modelo foi utilizado como extrator de características e era interligado a outro módulo baseado em cadeias ocultas de Markov. Essa alteração ultrapassou o desempenho do anterior modelo estado da arte em mais de 30% (HINTON *et al.*, 2012).

Após esse marco, diversas iniciativas de pesquisas vêm tentando melhorar ainda mais o desempenho dos atuais ASR (*Automatic Speech Recognition*). Alguns pesquisadores buscavam construir um sistema fim-a-fim totalmente baseado em redes neurais. Durante suas pesquisas, alguns grupos de pesquisadores obtiveram resultados satisfatórios substituindo o módulo estatístico, baseado em HMM e responsável por modelar a estrutura sequencial da fala, por modelos baseado em Redes Neurais Recorrentes (*Recurrent Neural Networks*, RNNs) (GRAVES *et al.*, 2006; GRAVES; MOHAMED; HINTON, 2013; SAK; SENIOR; BEAUFAYS, 2014; SAK *et al.*, 2014).

Aplicar alguma técnica no áudio para criar uma representação de dados costuma apresentar bons resultados. Isso ocorre devido aos dados de áudio brutos apresentarem uma alta quantidade de informações, sendo assim, a aplicação de pré-tratamentos ou utilização de métodos de extração de características reduz o volume de informações a serem identificados.

Ainda assim, alguns trabalhos na literatura obtiveram resultados promissores aplicando técnicas de aprendizado profundo aos sinais de áudios brutos, sem nenhum tipo de pré-processamento (Muckenhirn; Magimai.-Doss; Marcell, 2018; Dai *et al.*, 2017; Palaz; Magimai.-Doss; Collobert, 2015). Essa estratégia é interessante, pois apresenta uma metodologia totalmente *data-driven*, onde não é necessário um especialista para efetuar o tratamento dos dados, pois qualquer pessoa que domine a técnica de redes neurais, mas não tenha pleno domínio sobre técnicas de processamento de áudio, poderia aplicá-la sem grandes dificuldades.

Contudo, a utilização de recursos de pré-processamento costuma potencializar a extração de características representativas, as quais são relevantes ao processo de reconhecimento de padrões, direcionando os esforços do modelo para a compreensão das informações realmente úteis e abstendo-se de informações irrelevantes, como os ruídos de fundo ou espectros inexpressivos. Tendo isso em vista, trabalhos recentes dividem a

tarefa de reconhecimento de áudio em dois módulos, o primeiro módulo busca elaborar um modelo acústico, o qual é responsável por extrair características representativas do espectro de áudio. E então, essas características são posteriormente enviadas a um segundo módulo, que é composto por um classificador, o qual deverá correlacionar as informações de forma a identificar as sequências analisadas.

Em Meftah, Alotaibi e Selouani (2016), é feito um estudo analisando alguns dos principais extratores de características utilizados na literatura, utilizando como classificador um modelo baseado em HMM. Esse estudo é feito utilizando uma base de fonemas na linguagem arábica. Nesse trabalho, são comparados os extratores *Linear Predictive Coding* (LPC), *Mel Frequency Cepstral Coefficients* (MFCC), *Perceptual Linear Prediction* (PLP), *Logarithmic Mel-Filter Bank Coefficients* (FBANK), *Mel-Filter Bank Coefficients* (MELSPEC) e *Linear Prediction Reflection Coefficients* (LPREFC). O sistema que obteve melhor acurácia foi baseado em FBANKs, seguido pelo segundo melhor sistema, o qual utilizou espectrograma na escala de Mel (MELSPEC). Os estudos também comprovaram que o MFCC e o PLP apresentaram acurácia semelhante, já o LPC não obteve bons resultados para a tarefa de reconhecimento de fonemas.

Em Ding, Xie e Zhu (2015), foi realizado um experimento comparativo entre os extratores: *Logarithmic Mel-Filter Bank Coefficients* (FBANK), *Mel-Filter Bank Coefficients* (MFCC) e *Linear Prediction Coefficients* (LPC). Os modelos acústicos fornecidos pelos extratores de áudio são utilizados como entrada de uma DBN (Deep Belief Network). Durante o experimento, o modelo acústico utilizando FBANKs apresentou desempenho superior aos demais.

Diversos trabalhos têm proposto modelos acústicos utilizando a combinação de alguma técnica de transformação e técnicas de aprendizado profundo. No trabalho de Quintanilha, Biscainho e Netto (2017), utilizaram-se CNN e RNNs. O sinal de áudio de um trecho de fala é transformado em MFCCs e utilizado como entrada do sistema. No trabalho de Santos *et al.* (2015), além de usar CNNs e RNNs, foi usado como entrada do sistema o sinal de áudio transformado através da aplicação de FBANKs. Já em FAN e LIU (2018), é apresentada uma abordagem combinando CNNs e RNNs e é feita uma comparação entre as entradas de áudio transformadas em espectrograma em escala de Mel e FBANKs.

Além dos diferentes tipos de arquiteturas de sistemas de reconhecimento, também é necessário considerar o objetivo do reconhecimento. Se é para identificação de linguagem, reconhecimento de sentimentos, de locutor, ou de palavras. Quando se trata do último caso, ainda é possível subdividir a categoria em sistemas de reconhecimento de fala contínua ou palavras isoladas. Para ambos sistemas de reconhecimento de fala, é necessário definir qual será a unidade de saída. A saída do sistema pode ser tanto a palavra inteira, como a palavra subdividida em unidades fonéticas menores, por exemplo, fonemas, sílabas,

difones, trifones ou mesmo caracteres. Essas unidades são comumente chamadas de *tokens* (BRESOLIN, 2008).

Os sistemas de reconhecimento de fala contínua são sistemas que comumente apresentam uma alta complexidade quando comparados aos demais. Isso ocorre principalmente porque na fala humana não costumam existir pausas entre os *tokens*. Dessa forma, o contexto das palavras ajuda na tomada de decisão. As ferramentas comumente utilizadas na literatura para considerar o contexto na tomada de decisão são os HMMs (RABINER, 1989; JUANG; LEVINSON; SONDHI, 1986). Esses sistemas vêm sendo gradativamente substituídos por modelos que utilizam redes neurais recorrentes (GRAVES *et al.*, 2006; GRAVES; MOHAMED; HINTON, 2013; SAK; SENIOR; BEAUFAYS, 2014; SAK *et al.*, 2014).

Um fator de fundamental importância para o incentivo a evolução desses sistemas são os diversos desafios constantemente propostos à comunidade. Esses desafios possuem o objetivo de estimular o desenvolvimento de novas soluções, que consigam detectar de forma precisa uma grande diversidade de classes de objetos distintas. Como exemplos desses desafios, podem-se citar o Google AI Open Images Object Detection Track 2018, PASCAL Visual Object Classes Challenge 2007 e 2012 (VOC2007, VOC2012), Microsoft COCO: Common Objects in Context (MS COCO), ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (KUZNETSOVA *et al.*, 2018; DENG *et al.*, 2009; RUSSAKOVSKY *et al.*, 2015a; EVERINGHAM *et al.*, 2010; LIN *et al.*, 2014).

Em virtude do empenho da comunidade de visão computacional em superar esses desafios, surgiram algumas ferramentas muito interessantes, como o You Only Look Once (YOLO) (REDMON *et al.*, 2016) e Single-Shot MultiBox Detector (SSD) (LIU *et al.*, 2016). O YOLO é um detector de um estágio e sua principal contribuição é proporcionar a detecção de objetos em imagens em tempo real. Atualmente, o sistema YOLO está na sua terceira versão, a qual alcançou um resultado de 57,9% mAP (mean Average Precision) no desafio MS COCO. O detector SSD opera de maneira similar, em um único estágio, e detecta múltiplas categorias diretamente, predizendo suas categorias e sua posição através de um conjunto predefinido de quadros utilizando diferentes escalas. Esse sistema atingiu a marca de 53,3% no desafio MS COCO. Apesar de o resultado obtido pelo sistema SDD ter sido inferior em relação à precisão das detecções, esse detector obteve tempo de processamento menor e maior taxa de quadros por segundo, quando comparados ao sistema YOLO (JIAO *et al.*, 2019).

Os primeiros passos para investigar a aplicação das técnicas de detecção de objetos para classificar fonemas em espectrogramas de sinais de áudio foram dados por Algabri *et al.* (2020). No seu trabalho, os autores propuseram a utilização de reconhecimento de fonemas através de espectrogramas da base inglesa TIMIT e de uma base arábica. Eles utilizaram a arquitetura YOLO e também uma arquitetura chamada de CenterNet, em ambas obtiveram

resultados comparáveis a artigos recentemente publicados sobre reconhecimento de fonemas. O trabalho ainda investigou a utilização da técnica de transferência de conhecimento inter e intra-linguagem, isto é, os autores dividiram e avaliaram a tarefa de transferência de conhecimento em duas classificações. Uma delas foi chamada de tarefa de transferência intra-linguagem. Nessa abordagem, treinou-se previamente o modelo com uma base de dados e congelaram-se os seus pesos, posteriormente, carregaram-se os pesos obtidos durante o primeiro treinamento e treinou-se a rede utilizando uma nova base de dados de áudio da língua inglesa. O segundo teste é feito para investigar a utilização da transferência inter-linguagem, ou seja, a metodologia é semelhante ao processo anterior, no entanto, as bases utilizadas durante os treinamentos são de linguagens diferentes. Nessa ocasião, foram utilizadas a língua inglesa e arábica. Ambos processos apresentaram benefícios ao aplicar a técnica de transferência de conhecimento.

1.4 Contribuições

As principais contribuições deste trabalho são:

- Investigar a eficiência de um modelo de reconhecimento de fonemas utilizando técnicas de detecção de objetos através de redes neurais;
- Apresentar métodos de *Data Augmentation* para dados de áudio;
- Comparar o desempenho do modelo de identificação de fonemas com outros modelos propostos na literatura.

1.5 Organização do Trabalho

O trabalho está dividido em cinco capítulos. Além deste capítulo, no Capítulo 2, é apresentada a fundamentação teórica visando a identificar os principais módulos de um sistema de reconhecimento de fala e explicar como ocorre o seu funcionamento. Esse capítulo visa, ainda, a assegurar o entendimento dos conceitos que serão aplicados na metodologia e nas análises dos resultados deste trabalho. No Capítulo 3, apresentam-se as ferramentas que serão utilizadas para construir o modelo proposto, bem como avaliá-lo. O Capítulo 4 é destinado aos resultados obtidos nos experimentos aplicados e discussões. O Capítulo 5 apresenta o conjunto das conclusões mais importantes, destacando os resultados obtidos e propondo perspectivas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Funcionamento do sistema de fala

A voz é uma característica inerentemente humana baseada na produção de sons articulados, os quais são representados por uma linguagem. Através dessa ferramenta, pode-se estabelecer a comunicação entre os indivíduos. Os sons produzidos pela fala possuem interpretação gráfica através de letras e cada linguagem possui um alfabeto próprio.

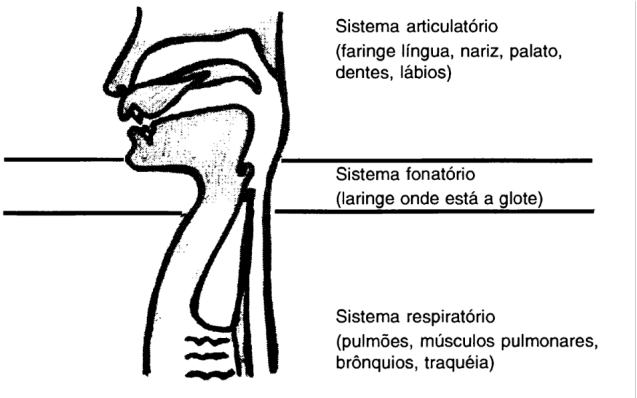
Ainda assim, é difícil encontrar sons representados exclusivamente por apenas uma letra isolada. Em decorrência disso, costuma-se utilizar outra unidade sonora considerada mais primitiva para representar os sons presentes na pronúncia das palavras. Essa unidade é conhecida como fonema.

A forma sistemática como cada língua organiza os sons é objetivo de estudo de uma área conhecida como *fonologia*. É responsabilidade dessa ciência distinguir os significados dos fonemas e a forma como se combinam para formar unidades linguísticas maiores, como as palavras. Essa área também define as variações que os fonemas podem apresentar (SILVA, 1999).

Em relação à produção dos sons, no sentido fisiológico de quais órgãos do sistema articulatório são utilizados, é responsabilidade de outra área, denominada de *fonética*. Essa ciência é responsável pela descrição, classificação e transcrição dos sons de fala.

O sistema responsável para a produção da fala é composto por diversos órgãos, que não possuem como responsabilidade primária a fala. Na verdade, esses órgãos têm outras funcionalidades como mastigar, engolir, respirar ou cheirar. Apesar desses órgãos não desempenharem exclusivamente a função de fala, um ser humano em suas capacidades normais é capaz de emitir sons em qualquer linguagem. Para executar essa tarefa, o homem utiliza um sistema do corpo humano chamado de aparelho fonador.

Figura 1 – Aparelho fonador.



Fonte: Silva (1999).

Na Figura 1, está ilustrado o aparelho fonador humano, o qual é composto por três sistemas: o sistema respiratório, o fonatório e o articulatório. Cada um desses sistemas possui uma função primária. O sistema respiratório, composto pelos pulmões, músculos pulmonares, brônquios e traqueia, tem como função primária a respiração. Quando esse órgão é ativado para a produção de sons, atua emitindo correntes de ar, as quais são modeladas em ondas sonoras pelos demais sistemas.

O sistema fonatório é composto pela laringe. Esse órgão é composto por uma série de cartilagens, articulações e músculos. No que diz respeito a fala, esse órgão é responsável por controlar a passagem de ar através de uma estrutura de músculos estriados, denominados de cordas vocais. A função primária desse sistema é atuar como válvula bloqueando a entrada de sólidos nos pulmões. O ato de engasgar-se é decorrente de quando a epligote, que é um dos músculos da faringe, não obstrui a passagem de sólidos para o sistema respiratório, por isso o corpo produz um reflexo e o sistema respiratório expele ar de forma a impedir a entrada do corpo estranho no sistema respiratório.

Já o sistema articulatório é composto pela faringe, língua, nariz, dentes e lábios. Sua função primária está principalmente relacionada à alimentação do indivíduo, atos como morder, mastigar, sentir o paladar, cheirar, sugar e engolir.

O aparelho fonador possui limitações fisiológicas, por isso é correto afirmar que há um número limitado de sons possíveis de ocorrer nas línguas naturais. Por exemplo, é possível pronunciar um som que necessite que a ponta da língua toque nos dentes incisivos superiores, porém é fisiologicamente impossível com a ponta da língua tocar a ponta do nariz. Na verdade, um conjunto de cerca de 120 símbolos é suficiente para modelar os sons consonantais e vocálicos que ocorrem nas línguas naturais (SILVA, 1999).

2.2 Processamento digital de sinais

Para fins didáticos, os sinais costumam ser categorizados em diversos aspectos, em relação à periodicidade, a energia, a potência, se são contínuos ou discretos em relação ao tempo, analógicos ou digitais e, ainda, se são determinísticos ou probabilísticos. Quando se trata dos termos contínuo no tempo e discreto no tempo, estes qualificam o sinal ao longo do eixo de tempo, geralmente representado no eixo horizontal.

Por outro lado, os termos analógico e digital qualificam a amplitude do sinal, geralmente representado no eixo vertical. A fala é um sinal analógico, contínuo e não periódico. Entretanto, quando se executa a tarefa de aquisição do sinal de voz, as ondas de pressão emitidas pelo locutor são transmitidas através do ar e convertidas em ondas elétricas através dos microfones e amplificadores.

Por conta de limitações técnicas, os computadores digitais não permitem que o sinal seja armazenado considerando um conjunto infinito de pontos, por isso é necessário

realizar amostragens periódicas. É de extrema importância manter a taxa de amostragem do sinal suficientemente alta, para que seja possível a reconstrução sem perda significativa da informação do sinal original. O fundamento quantitativo necessário para esse propósito é fornecido pelo teorema da amostragem (LATHI, 2006).

2.3 Transformações de sinais

Pré-enfase

Após gravar e armazenar o sinal digitalmente, costumam-se fazer algumas transformações no sinal. Historicamente, uma transformação aplicada é chamada de pré-ênfase, em resumo, é um tipo de normalização. Entre as principais vantagens desse método, pode-se citar o balanceamento do espectro das frequências, visto que altas frequências costumam ter magnitudes menores quando comparadas às magnitudes das frequências baixas. Outro benefício importante é a melhora da taxa de sinal-ruido (SNR).

O filtro de pré-ênfase pode ser aplicado a um sinal x usando um filtro de primeira ordem, como na Equação 2.1:

$$y(t) = x(t) - \alpha * x(t-1) \tag{2.1}$$

O valor adotado para α geralmente fica entre 0,95 e 0,97.

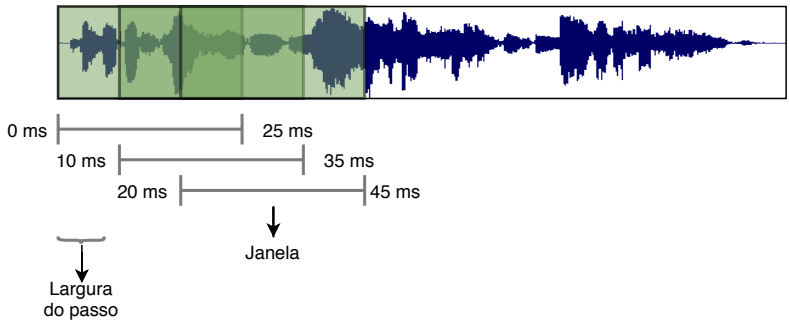
Janela deslizante

Para entender o próximo passo do processo de transformação, é importante ter em mente que as frequências de um sinal modificam-se temporalmente. Geralmente, calcular a Transformada de Fourier, considerando o sinal completo, irá gerar perda na representação do sinal.

Para contornar esse problema, assume-se que ao considerar um curto período de tempo, as frequências do sinal são estacionárias e dessa forma, é possível aplicar a Transformada de Fourier nesse curto período de tempo e obter uma boa representação do sinal.

Esse processo consiste em deslizar ao longo do sinal uma janela, cujo comprimento costuma limitar-se entre 20 a 40 ms. Para evitar perda de informação, a distância entre uma janela e a sua sucessora costuma ser menor que a largura total da janela, por isso, as janelas sobrepõem-se. Um valor comumente adotado para a largura da janela é de 25 ms e para a largura do passo de 10 ms. Isso gera uma sobreposição de 15 ms. A Figura 2 ilustra esse processo.

Figura 2 – Exemplo de janela deslizante.



Fonte: Autor.

Janelamento

O processo anterior gera sobreposição do sinal. Com objetivo de suavizar esse efeito, aplica-se um filtro de janelamento para suavizar as bordas do sinal. Existem exemplos de diversos filtros na literatura, sendo o janelamento de *Hamming* um filtro muito utilizado, que pode ser representado pela Equação 2.2:

$$w[n] = 0,54 - 0,46 * \cos(\frac{2\pi n}{N-1}) \tag{2.2}$$

Transformada de Fourier

Para cada janela extraída anteriormente, aplica-se a Transformada de Fourier. Essa transformada é originada a partir da transformada clássica. O procedimento também é chamado de Short-Time-Fourier-Transform (STFT). A STFT é recomendada para sinais que possuem componentes que variam muito temporalmente, como a fala.

A transformada de Fourier clássica calcula o espectro das frequências considerando toda a extensão do sinal. Já a STFT trabalha em áreas localizadas, proporcionando informação de frequência mais precisa, distinguindo as variações das frequências do sinal ao longo do tempo. A STFT é dada pela Equação 2.3:

$$\begin{aligned} X_{STFT}[m,n] &= \sum_{k=0}^{L-1} x[k]g[k-m]e^{\frac{-j2\pi nk}{L}}, \\ X[k] &= \sum_{m}\sum_{n} X_{STFT}[m,n]g[k-m]e^{\frac{-j2\pi nk}{L}} \end{aligned} \tag{2.3}$$

Onde $x[k]$ representa o sinal e $g[k]$ representa um dos L janelamentos. A partir da Equação 2.3, a STFT de $x[k]$ pode ser interpretada como a transformada de Fourier do produto de $x[k]g[k-m]$.

Espectro de potência

Com o resultado da etapa anterior, calcula-se a potência espectral do sinal através da Equação 2.4:

$$S[k] = |X[k]|^2 = (X[k])^2 + (jX[k])^2 \tag{2.4}$$

O objetivo desse procedimento é intensificar as quantidades de energia nas diferentes faixas de frequência. Esse processo é importante, pois considera as altas frequências do sinal, uma vez que essa área do espectro contém informações as quais podem ter relevância para aprimorar o desempenho dos classificadores.

Considerando o caso dos sons vocálicos como exemplo, esses sons costumam apresentar maior concentração de energia em áreas de baixa frequência. No entanto, muitas vezes as informações contidas nas faixas de frequências mais altas são de fundamental importância para a diferenciação de algumas vogais, as quais possuem pronúncia semelhante.

Escala Mel

O processo de transformação do espectro de potência para a chamada escala Mel é um método que modela a resposta em frequência de maneira semelhante ao que ocorre no sistema auditivo humano (FAN; LIU, 2018; GORDILLO, 2013).

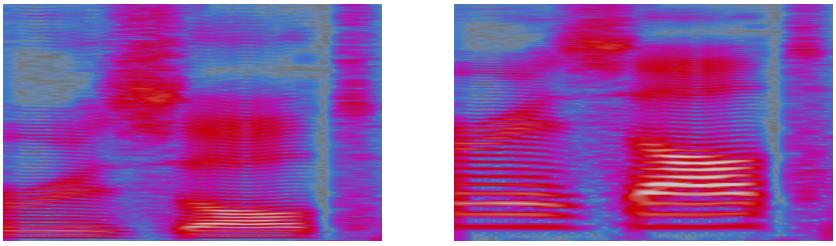
Para transformar o espectro de potência na escala Mel, deve-se aplicar um banco de K filtros à potência espectral $S[k]$. O banco de filtros é composto por filtros espaçados de acordo com a escala de frequência Mel, representada pela Equação 2.5:

$$Mel(f) = 1125 \ln(1 + \frac{f}{700}) \tag{2.5}$$

Na prática, o efeito de utilizar o espectrograma na escala Mel é promover um espaçamento entre as frequências de baixa magnitude localizadas na parte inferior do espectrograma. Enquanto as frequências de alta magnitude são levemente compactadas, conforme a Figura 3.

Figura 3 – Comparativo ilustrando o efeito da aplicação da escala Mel ao sinal.

a) Espectrograma na escala linear b) Espectrograma na escala Mel



Fonte: Autor.

2.4 Redes Neurais

A busca pela construção de sistemas inteligentes, ou pelo menos com o comportamento inteligente, tem sido motivada pelo reconhecimento do funcionamento do cérebro humano e como esse mecanismo processa as informações de maneira diferente a de um computador convencional. O cérebro pode realizar diversas tarefas, desde as mais simples às mais complexas, de forma sutil.

Diversas tarefas, consideradas de alta complexidade, computacionalmente falando, são executadas de forma elementar pelo cérebro humano. Como exemplo dessas tarefas, têm-se o reconhecimento de padrões, o controle motor, o armazenamento de informações, entre outras. A execução dessas e muitas outras tarefas só é possível graças à complexa estrutura biológica humana (GAMA *et al.*, 2011).

O desenvolvimento da teoria das redes neurais artificiais (RNAs) teve como principal inspiração a estrutura e o funcionamento do sistema nervoso cerebral mamífero. Esse sistema possui alta capacidade de generalização do aprendizado entre tarefas, bem como, grande facilidade para a aquisição de novos conhecimentos. A estrutura nervosa cerebral pode ser referenciada por um emaranhado complexo de células, denominadas por neurônios.

Os neurônios são um tipo de estrutura cujas conexões estão em constante aprimoramento, o que permite que eles adaptem-se ao meio ambiente. Essa característica de adaptação é fundamental para o funcionamento das unidades de processamento de informação do cérebro humano. Não é diferente em relação às redes neurais construídas com neurônios artificiais. Em um panorama, uma RNA é uma máquina projetada para imitar a maneira como o cérebro realiza uma tarefa particular.

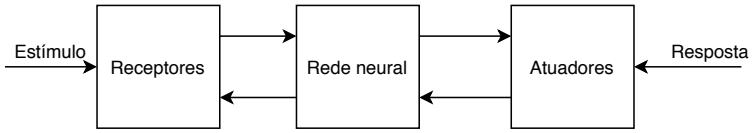
As RNAs são sistemas computacionais distribuídos compostos por unidades de processamento simples, densamente interconectadas. Essas unidades são conhecidas como neurônios artificiais e estão dispostas em uma ou mais camadas, interligadas por conexões, geralmente unidimensionais (GAMA *et al.*, 2011). Outra definição, que segue a mesma linha, diz que uma rede neural é um processador densa e paralelamente distribuído. Ela é constituída por unidades de processamento simples, que tem a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso. Essa estrutura assemelha-se ao cérebro em dois aspectos principais: o conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem; e as forças de conexões entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido (HAYKIN, 2007).

A estratégia utilizada para concretizar o processo de aprendizagem funciona através da modificação dos pesos sinápticos entre os neurônios de uma rede, de forma ordenada, para alcançar o objetivo desejado. Além da modificação dos pesos, também é possível configurar a rede para modificar sua própria topologia, o que é motivado pelo fato que os

neurônios naturais podem morrer e novas conexões sinápticas podem vir a surgir. Esse fato contribui diretamente com o processo de generalização, que nada mais é do que a característica da rede neural de produzir saídas adequadas para as entradas que não estavam presentes durante o treinamento. Isso indica que o aprendizado foi consolidado.

O sistema nervoso humano pode ser representado de forma resumida a um sistema de três estágios, similar a um modelo de processamento computacional simples. Esses sistemas são compostos por um bloco de entrada, um de processamento e um de saída. Quando se trata do sistema nervoso humano, o centro de processamento do sistema é o cérebro. Na Figura 4, o cérebro está representado pelo bloco da rede neural e trabalha recebendo informações continuamente. Os receptores convertem estímulos do corpo humano ou do ambiente em impulsos elétricos, os quais transmitem informação para a rede neural. Já os atuadores convertem os impulsos elétricos gerados pela rede neural em saídas do sistema.

Figura 4 – Representação em diagrama de blocos do sistema nervoso.



Fonte: Haykin (2007).

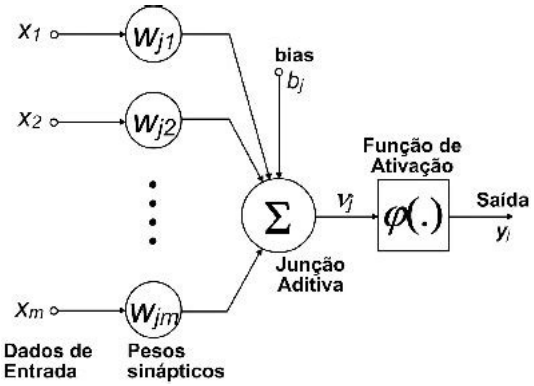
Analisando a Figura 4, pode-se imaginar o neurônio como uma caixa, a qual recebe um ou vários sinais elétricos de outros neurônios, efetua um determinado processamento e devolve uma ou mais saídas, as quais estarão conectadas a outros neurônios ou mesmo diretamente à saída do sistema. Essas interações entre os neurônios ocorrem em uma unidade estrutural e elementar, denominada sinapse. Nas descrições clássicas de organização neural, define-se que a sinapse é uma conexão simples que impõe ao neurônio receptor um estado de excitação ou inibição.

Estrutura do Neurônio

Assim como no cérebro biológico, o neurônio artificial também é uma unidade de processamento de informações e possui características semelhantes, como as sinapses. No entanto, no neurônio artificial, o conjunto de sinapses ou conexões é caracterizado por um peso, que representa a sua "força". Observando a Figura 5, para cada um dos sinais de entrada m representados por x_m, existe um peso w_j que representa a força da conexão daquela entrada com o neurônio j. No interior do neurônio j, há uma junção aditiva, que tem o objetivo de somar os sinais de entradas ponderados pelos seus respectivos pesos. Ao final do neurônio, há uma função de ativação, cujo objetivo é restringir a amplitude da saída de um neurônio a um valor finito. É comum o intervalo normalizado da amplitude ser descrito pelo intervalo unitário [0,1] ou [-1,1]. O modelo apresentado na Figura 5 inclui

o recurso do *bias* representado por b_j, sua função é aumentar ou diminuir a intensidade da entrada da função de ativação (HAYKIN, 2007).

Figura 5 – Modelo não-linear de um neurônio.



Fonte: Haykin (2007).

O neurônio artificial representado na Figura 5 também pode ser descrito matematicamente através das Equações 2.6 e 2.7:

$$v_j = \sum_{k=1}^{m} w_{jk} x_k \tag{2.6}$$

$$y_j = \varphi(v_j + b_j) \tag{2.7}$$

Na Equação 2.6, define-se a entrada líquida do neurônio v_j. Onde x_1 a x_m representam os sinais de entrada, enquanto w_{j1} a w_{jm} representam os pesos sinápticos dessas entradas. Na Equação 2.7, enquanto a saída do neurônio é representada por y_j, a entrada líquida do neurônio é representada por v_j e o *bias* por b_j, já a função de ativação é representada por φ.

A partir do entendimento do funcionamento básico de um neurônio, o processo de construção de uma rede neural ocorre através do agrupamento de diversos neurônios em estruturas denominadas de camadas. A quantidade de camadas de uma rede definirá sua profundidade, já a forma como essas camadas estão conectadas define a topologia da rede.

Entre as várias topologias de redes disponíveis na literatura, estão as redes convolucionais, em inglês *Convolutional Neural Network* (CNN). Essa categoria de rede tem como característica principal a utilização de convoluções entre suas entradas, produzindo mapas de características, os quais compartilham pesos entre os neurônios das camadas.

Back-propagation

Esse algoritmo ou suas variações são comumente utilizadas para o treinamento de redes neurais. O *back-propagation* é um tipo de algoritmo de treinamento supervisionado

que utiliza pares de entrada e saída desejada. O algoritmo ajusta o peso da rede através de um mecanismo de correção de erro. O treino ocorre em duas fases, a primeira fase é chamada de $foward$, onde o algoritmo percorre a rede da entrada até a saída e ao final, produz um resultado. Esse resultado é comparado com a saída desejada. A segunda fase, chamada de *backward*, utiliza a diferença entre a saída desejada e a saída fornecida pela rede para ajustar os pesos das conexões.

O erro calculado na saída do neurônio j na iteração n pode ser definido por:

$$e_j(n) = d_j(n) - y_j(n) \tag{2.8}$$

Onde $d_j(n)$ é o resultado desejado para o n-ésimo exemplo de treino e $y_j(n)$, o resultado calculado pela rede. É comum considerar intensidade do erro através da operação $\frac{1}{2}e_j^2$. Tendo isso em mente, para calcular o erro de um determinado exemplo n, efetua-se o somatório da potência do erro de todos os neurônios j da camada de saída, conforme a Equação 2.9:

$$\varepsilon(n) = \frac{1}{2} \sum_{j \ \epsilon \ C} e_j^2(n) \tag{2.9}$$

Onde C é o conjunto de todos os neurônios de saída. Para adquirir o erro de um ciclo de treinamento, que pode considerar o conjunto inteiro de treino ou apenas um lote desse conjunto, calcula-se o erro quadrático médio, o qual é obtido através da soma de todos os erros $\varepsilon(n)$, considerando as iterações n. A Equação 2.10 representa essa operação, onde N é o número total de elementos do conjunto de treino ou daquele lote de treino.

$$\varepsilon_{AV} = \frac{1}{N} \sum_{n=1}^{N} \varepsilon(n) \tag{2.10}$$

2.4.1 Redes Neurais Convolucionais

O funcionamento das redes neurais convolucionais ocorre através dos mesmos elementos básicos que as redes tradicionais. Ambas são constituídas de camadas de neurônios que possuem ligações sinápticas com outros neurônios e estas ligações possuem pesos e *bias*. Assim como as redes tradicionais, as redes convolucionais necessitam de treinamento e utilizam funções de ativação. Se bem observado, as últimas camadas da rede convolucional são exatamente iguais as de uma rede tradicional. Contudo, essa arquitetura de rede diferencia-se das redes tradicionais principalmente pela forma como concebe as suas entradas, as quais são tipicamente imagens.

Nesse momento, o leitor pode questionar-se sobre redes tradicionais também receberem imagens como entrada. É possível adaptar uma imagem e inseri-la em uma rede linear. No entanto, arquiteturas de redes tradicionais ao utilizar imagens costumam precisar de

diversos neurônios, gerando um alto número de pesos a serem treinados. Enquanto a rede convolucional utiliza uma estratégia de pesos compartilhados, na qual um conjunto de pesos, representados por um filtro, são os mesmos para todas as regiões da imagem e essa característica reduz consideravelmente o número de parâmetros utilizados por esse tipo de rede (SIMONYAN; ZISSERMAN, 2014).

As redes convolucionais costumam possuir três estruturas básicas bem definidas: a camada convolucional, a camada de *pooling* e a camada totalmente conectada. A seguir, é tratado sobre a operação de convolução e em seguida, sobre as camadas citadas.

Operação de convolução

A operação de convolução é um procedimento matemático que relaciona duas matrizes, f e g. Na área de processamento de imagens, uma imagem é definida como uma matriz bidimensional. Nesse domínio, a operação de convolução é amplamente utilizada para diversas tarefas como operações morfológicas, detecção de bordas, suavização de imagem, extração de atributos, entre diversas outras aplicações (GONZALEZ; WOODS, 2009).

Sejam f e g funções contínuas no domínio de uma variável x, a operação de convolução é definida pela Equação 2.11, onde $*$ é chamado de operador de convolução e g é um filtro, que é comumente chamado de *kernel*.

$$f(x) * g(x) = \int_{\tau=-\infty}^{\infty} f(\tau) \cdot g(x-\tau)\mathrm{d}\tau \tag{2.11}$$

Quando são considerados sinais discretos, a operação de convolução é definida pela Equação 2.12:

$$f(x) * g(x) = \sum_{n=-\infty}^{\infty} f[n] \cdot g[x-n] \tag{2.12}$$

Para aplicar esses conceitos em processamento de imagem é necessário transformar a definição de convolução unidimensional para um formato bidimensional, nesse caso, a função receberá duas variáveis. Efetuando essa modificação, obtêm-se as Equações 2.13 e 2.14, para sinais contínuos e discretos, respectivamente.

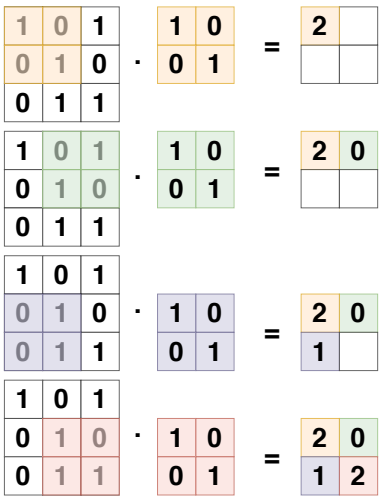
$$f(x,y) * g(x,y) = \int_{\tau_1=-\infty}^{\infty} \int_{\tau_2=-\infty}^{\infty} f(\tau_1,\tau_2) \cdot g(x-\tau_1, y-\tau_2)\mathrm{d}\tau_1\mathrm{d}\tau_2 \tag{2.13}$$

$$f(x,y) * g(x,y) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2] \tag{2.14}$$

Uma imagem é um exemplo de sinal discreto que é definido através de uma função bidimensional. Na prática, utiliza-se uma matriz de elementos, os quais recebem o nome

de *pixel*. Na Equação 2.14, a função f é a imagem a ser analisada, enquanto a função g é o filtro ou *kernel*, que também pode ser entendido como imagem, já que também é formado por uma matriz de elementos. Portanto, a operação de convolução via Equação 2.14 é o resultado do somatório da multiplicação de cada elemento da imagem por cada elemento do *kernel*. Na Figura 6, é ilustrado um exemplo da aplicação da operação de convolução. Observando da esquerda para a direita, têm-se: a imagem, o filtro e o resultado da convolução. Percebe-se que após efetuar a operação, o tamanho da imagem diminuiu. Contudo, existem algumas técnicas, como adicionar bordas com zeros à imagem original antes de efetuar a operação. Esse procedimento conservará o tamanho inicial da imagem.

Figura 6 – Exemplo de convolução: a matriz 3×3 à esquerda representa a imagem, a matriz 2×2 representa o *kernel* e a direita é o resultado da convolução.



Fonte: Autor.

Camada convolucional

Essa camada é o fator principal que diferencia essa arquitetura de rede neural das demais. Nessa parte da rede, é realizado um procedimento que ocorre através da aplicação da técnica de convolução na imagem de entrada através de um conjunto de filtros. A técnica de convolução é derivada do processamento de sinais, conforme explicado anteriormente. No entanto, os filtros na rede convolucional são compostos por valores reais que representam os pesos dessa camada da rede, por isso os valores que compõem a matriz dos filtros na rede convolucional são alterados ao longo do treinamento. O objetivo principal dessa camada é extrair características da imagem e produzir mapas de características que servirão posteriormente para auxiliar na tomada de decisão ao final do processo.

Na Figura 6, é ilustrado um exemplo de convolução utilizando uma imagem hipotética de dimensões 3×3 e um filtro de dimensões 2×2. Nessa demonstração, considera-se apenas um canal da imagem, que na prática significa uma imagem em preto e branco. No entanto, é muito comum durante o processamento utilizar imagens coloridas,

as quais geralmente são compostas por três canais. O padrão de cores mais comumente utilizado é o sistema RGB. Caso a imagem seja colorida e utilize o sistema RGB, será composta por três matrizes e, durante o processamento, será utilizado um filtro para cada canal.

Outro parâmetro muito importante que deve ser considerado é o deslizamento, também chamado de *stride*. Na Figura 6, é utilizado o valor um, ou seja, o filtro é deslizado deslocando um *pixel* por vez, seja no sentido horizontal ou vertical. O valor do *stride* fica a critério do projetista da rede, contudo, deve-se respeitar as dimensões da imagem e do filtro.

Após efetuar o processo de convolução, é muito comum aplicar uma função de ativação. Uma função amplamente utilizada por projetistas é a função ReLU, do inglês *Rectified Linear Units*. Essa função é definida pela Equação 2.15 e é aplicada em cada elemento resultante do processo convolucional. Seu papel consiste em verificar se o elemento é maior que zero e em caso positivo, o valor permanece o mesmo, caso contrário, o valor do elemento é substituído pelo valor zero.

$$f(x) = max(0, x) \tag{2.15}$$

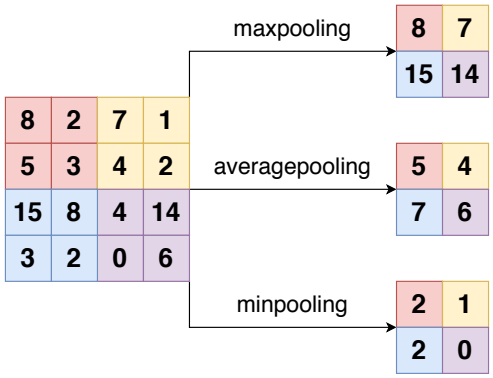
Camada de *pooling*

A camada de *pooling* é independente da camada de convolução e seu papel é semelhante ao processo de *down-sampling*, comum em processamento digital de sinais. O objetivo dessa função é reduzir espacialmente o tamanho das amostras. Em decorrência disso, ela reduz significativamente a quantidade de parâmetros a serem analisados pela rede. Semelhantemente à camada de convolução, na camada de *pooling*, também é necessário determinar o tamanho do filtro e o *stride* ou "unidade de deslizamento". As dimensões do filtro definem a extensão da vizinhança considerada para aplicar o procedimento.

Em redes convolucionais, é comum o uso do *maxpooling* e do *average pooling*, mas também existe o *minpooling*. O *maxpooling* transforma os n vizinhos no maior valor do grupo, o qual será responsável por representar aquela área, já o *average pooling* calcula a média da vizinhança e o resultado irá caracterizar aquela área e o *minpooling* considera o menor valor entre a vizinhança.

Na Figura 7 está ilustrado os três tipos de *pooling* citados, nesse exemplo foi considerado apenas um canal da imagem com dimensões 4×4, o filtro possui dimensões 2×2 e o *stride* utilizado foi 2. Vale ressaltar que, da mesma forma que ocorre na camada de convolução, as operações de *pooling* ocorrem independentes em cada um dos canais, sendo assim, caso o exemplo possuísse mais canais, a operação também seria aplicada nesses de forma independente.

Figura 7 – Exemplo da aplicação de *pooling* utilizando uma imagem de dimensões 4×4 e filtro de dimensões 2×2 com *stride* 2.



Fonte: Autor.

Camada totalmente conectada

A camada totalmente conectada é similar a uma rede neural linear tradicional, as entradas da camada totalmente conectada são os resultados obtidos anteriormente através das convoluções e *poolings*. Um fator importante a considerar-se é que nem sempre as camadas anteriores fornecem um resultado linear, de modo que possa ser introduzida diretamente nessa parte da rede. Por conta desse motivo, muitas vezes é necessário "achatar" os mapas de características obtidos através das camadas anteriores, esse processo também é chamado de *flatten*.

Para obter o número total de parâmetros necessários, deve-se multiplicar o número de mapas de características pelo produto de suas dimensões. Por exemplo, sejam 64 mapas de características de dimensões 3×3, o número de parâmetros passados para a camada totalmente conectada será $64 \times 3 \times 3 = 576$. Após criar a estrutura necessária para receber os dados da camada anterior, pode-se estender o número de camadas dessa parte da rede, conforme as necessidades do projeto.

Ao final da camada totalmente conectada, deve-se acoplar as saídas de seus neurônios à camada de saída da rede. Para a tarefa de classificação, essa camada possuirá o número de neurônios igual ao número de classes do problema. A camada de saída está acoplada uma função de ativação, dependendo do número de classes consideradas no problema utilizará-se um tipo de função.

Quando o problema envolve apenas duas classes, utiliza-se a função *sigmoid*, já quando o número de classes é maior que dois, costuma-se utilizar uma função de ativação chamada de *softmax*, representada pela Equação 2.16. Nessa equação, K representa o número de classes da rede e z_j representa a saída do neurônio j que corresponde a uma das classes do problema. Em resumo, essa função transforma a probabilidade de ativação em cada neurônio utilizando valores que variam de 0 a 1, e os divide pelo somatório de

das saídas de todos os neurônios.

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.16}$$

Após aplicar a função de ativação, a rede retornará uma indicação de a qual classe aquela entrada pertence e a partir disso, será calculado o erro através do algoritmo *backpropagation*. Posteriormente, esse erro é retro-propagado para as camadas anteriores e será responsável por ajustar os pesos dessas camadas, realizando o treinamento da rede.

2.4.2 Data Augmentation

Embora o ser humano possua extrema facilidade em designar tarefas visuais, como reconhecer um rosto ou um objeto específico e até mesmo uma cena de um ambiente, para o computador, tarefas que envolvem visão computacional costumam ser extremamente complicadas. Para o computador, processar um conjunto de *pixels* que compõem uma imagem e descobrir o que há na figura não é de resolução trivial. Na prática, embora não seja verdade para todos os tipos de algoritmos de aprendizado de máquina, especificamente para as redes convolucionais e redes profundas em geral, quanto mais dados distintos estiverem disponíveis para o treinamento, maior será a probabilidade de generalização.

No entanto, muitas vezes a obtenção de dados é custosa ou inviável por diversos motivos, e em decorrência disso, algumas práticas tornaram-se comuns para produzir novos exemplos a partir do conjunto de dados original. Na literatura existem diversos exemplos de práticas que costumam gerar bons resultados, entre elas estão: o espelhamento horizontal ou vertical da imagem; a rotação da imagem em torno do seu próprio eixo; a modificação das intensidades dos canais da imagem, isto é, incrementar ou diminuir os valores dos *píxels* de um canal, ou mesmo misturar as intensidades entre os canais. Contudo, é necessário ter cautela e verificar o domínio do problema ao aplicar essa técnica. Pois, dependendo do problema, o resultado desse processo pode ter efeito contrário e prejudicar o desempenho do modelo.

2.4.3 Transfer Learning

Uma prática que tem ganhado muito destaque nos últimos anos é a reutilização dos conhecimentos de um modelo pré-treinado para uma determinada tarefa. Através dessa técnica, é possível reutilizar não só a topologia de rede, mas também reutilizar o conhecimento adquirido ao longo de um treinamento anterior, mesmo para tarefas diferentes. Uma forma muito comum de aplicar essa técnica é adquirindo um conjunto de pesos pré-treinados para outro problema.

Adquirir pesos treinados para outro problema não é difícil, uma vez que habitualmente os desenvolvedores disponibilizam os pesos de suas arquiteturas treinadas para

determinados desafios, como ImageNet (ILSVRC) (DENG *et al.*, 2009), Microsoft COCO detection (LIN *et al.*, 2014), PASCAL Visual Object Classes Challenge (EVERINGHAM *et al.*, 2015).

Resumidamente, esse processo ocorre através do congelamento dos pesos de uma rede treinada para determinada tarefa, por exemplo, identificar semáforos ou identificar gatos e cachorros. Deve-se inicializar o modelo da rede e carregar os pesos com o conjunto de pesos pré-treinados adquiridos previamente. O próximo passo é adaptar a última camada do modelo carregado e substituir sua última camada por uma camada que contenha o número de neurônios necessários para resolver o seu problema. Por fim, treina-se apenas a última camada da rede, mas também é possível descongelar os pesos da rede e continuar o treinamento a partir de um determinado ponto.

Um grande benefício dessa técnica é que é possível reaproveitar todo o esforço de outros pesquisadores através dos conhecimentos de sua rede, teoricamente o que ocorre é o seguinte, ao treinar a rede para uma determinada tarefa, por exemplo, o desafio da ImageNet, a rede aprende a encontrar mapas de características que forneçam informações suficientes à camada totalmente conectada permitindo a sua tomada de decisão correta. Dessa forma, ao substituir apenas a última camada, você só estará remapeando as conexões para as últimas camadas, mas em tese a rede já possui conhecimento de como identificar aqueles objetos na imagem.

2.4.4 Arquitetura de Redes Convolucionais

Entendendo o básico de como funciona uma rede convolucional, um tópico importante a discutir-se é sobre as arquiteturas de redes neurais disponíveis na literatura. É importante ressaltar que qualquer pessoa pode construir sua própria rede convolucional, da forma que desejar. É possível definir a ordem entre de camadas convolucionais e de *pooling*, bem como a quantidade de cada uma dessas camadas, o número de neurônios, o tamanho dos filtros, os parâmetros de uma rede são todos customizáveis e podem ser arquitetado da forma que o projetista desejar.

Contudo, um fator significativo a considerar-se é que existem diversos exemplos de arquiteturas na literatura que se mostraram muito eficientes ao aprender padrões. Comumente, são lançados desafios a comunidade de visão computacional com o objetivo de promover desenvolvimento de novas propostas de arquiteturas de redes que consigam solucionar os mais diversos problemas de reconhecimento de imagem. Alguns exemplos de destaque são os desafios Google AI Open Images Object Detection Track 2018 , PASCAL Visual Object Classes Challenge 2007 e 2012 (VOC2007, VOC2012), Microsoft COCO: Common Objects in Context (MS COCO), ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (KUZNETSOVA *et al.*, 2018; DENG *et al.*, 2009; RUSSAKOVSKY *et al.*, 2015a; EVERINGHAM *et al.*, 2010; LIN *et al.*, 2014).

Por volta de 2012, as redes convolucionais começaram a ser amplamente difundidas na área de visão computacional, quando Alex Krizhevsky venceu o desafio ImageNet (ILSVRC 2012) (RUSSAKOVSKY *et al.*, 2015b) com sua arquitetura de rede AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Desde então, diversas arquiteturas de redes têm sido propostas nos últimos anos. As modificações são muito diversas, como aumentar o número de camadas ou número de neurônios, ou mesmo aplicar outras técnicas mais sofisticadas que são discutidas mais a frente. Em geral, quanto maior a rede e mais técnicas sofisticadas são utilizadas, obtém-se mais acurácia, no entanto, essa eficiência não se reflete em relação ao tamanho do modelo e a sua velocidade de processamento.

Analisando da perspectiva de aplicações do mundo real, onde o usuário nem sempre terá a sua disposição uma máquina de alto desempenho, é extremamente difícil utilizar estes modelos mais sofisticados em um dispositivo cujo poder computacional é restrito. Por isso, é muito comum utilizar a comunicação com a internet para enviar os dados para serem processados em um *datacenter*. No entanto, em algumas situações não há acesso à internet, portanto o processamento precisa ser feito *offline*.

A relação entre acurácia e eficiência das arquiteturas é um fator importante em um projeto envolvendo redes neurais. Existem diversas opções de arquiteturas densas e profundas disponíveis na literatura. No entanto, a demanda por redes que funcionem *offline* em dispositivos de baixa capacidade computacional tem feito surgir algumas arquiteturas mais reduzidas como a MobileNet, ShuffleNet e SqueezeNet (HOWARD *et al.*, 2017; IANDOLA *et al.*, 2016; ZHANG *et al.*, 2018).

Anteriormente, ao abordar um pouco sobre as operações utilizando filtros e convoluções, comentou-se sobre a possibilidade de redução do tamanho da imagem através da técnica de *pooling*. Adicionalmente, também é possível diminuir o tamanho de uma imagem utilizando o valor do *stride* maior que um. Essas técnicas são muito utilizadas em diversas aplicações. O resultado da sua aplicação é a redução das dimensões das entradas da rede e com isso se reduz o número total de parâmetros a serem processados.

Embora essa técnica seja muito eficiente, ainda assim, geralmente as redes convolucionais costumam efetuar muitas operações matemáticas e demandar um alto poder de processamento. Pensando nesse aspecto, alguns pesquisadores começaram a estudar diferentes formas de aplicar as operações de convolução. De modo que, desenvolveu-se uma técnica chamada de convolução *pointwise* ou convolução 1×1 (LIN; CHEN; YAN, 2013).

Uma imagem geralmente é representada pelas suas dimensões de altura e largura, mas também possui um elemento de volume que, geralmente é definido pelos seus canais, por exemplo, os canais RGB. Pensando em uma imagem de dimensões $D_f \times D_f \times M$, onde D_f é a dimensão da imagem e M, o número de canais dessa imagem, ao aplicar uma operação de convolução utilizando o valor do *stride* um e um filtro de dimensões $D_k \times D_k$, o resultado ao final do processo será uma imagem de dimensões D_f - $D_k + 1$. Considerando

apenas um mapa de características produzido a partir de uma imagem inicial, a qual possui três canais, o filtro utilizado na operação também deverá possuir três canais. Portanto, para cada deslizamento que o filtro realizar, serão realizadas $D_k \times D_k \times M$ multiplicações. Sendo assim, essa operação deverá ser repetida N vezes, de acordo com a quantidade de mapas de características que se deseja obter ao fim do processo. Tendo isso em vista, o custo computacional da operação de convolução padrão é representado pela Equação 2.17:

$$D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f \tag{2.17}$$

Onde D_f representa as dimensões da imagem original, D_k representa o tamanho do filtro, M e N representam o número de canais de entrada e o número de canais de saída, respectivamente.

Com o propósito de reduzir o custo computacional envolvido nessa operação, surgiu a técnica de convolução de *Depthwise Separable Convolution*. Essa técnica basicamente divide o processo anterior em duas etapas. A primeira etapa efetua uma convolução de maneira semelhante a anterior, cada canal é tratado individualmente. A diferença é que, ao final do processo, o número inicial de canais é preservado e a operação é aplicada apenas uma vez, dessa forma, obtém-se a Equação 2.18:

$$D_k \cdot D_k \cdot M \cdot D_f \cdot D_f \tag{2.18}$$

A convolução *Depthwise* é mais eficiente em relação a convolução padrão, no entanto, essa etapa é responsável apenas por aplicar os filtros aos canais e não os combina para extrair novas características. Por isso, é necessária uma segunda etapa que calcula uma combinação linear entre a saída da etapa anterior. Essa parte do processo é chamado de convolução 1×1 ou *Pointwise convolution*. Nesse ponto, utiliza-se um filtro 1× 1× M, onde M é o número de canais da imagem. O resultado dessa segunda etapa é um mapa de característica que possui as dimensões $D_f \times D_f \times 1$, logo para N mapas, obtêm-se as dimensões $D_f \times D_f \times N$. A combinação entre a convolução *Depthwise* e a convolução 1×1 (*Pointwise*) possuem o custo computacional descrito pela Equação 2.19:

$$D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + N \cdot M \cdot D_f \cdot D_f \tag{2.19}$$

Nesse momento, é normal o leitor questionar-se qual a vantagem desse procedimento. O entendimento dessa operação não surge de forma intuitiva e muitas vezes, a primeira impressão ao analisar essa técnica é de que não tem muita eficácia ou serventia, no entanto, essa técnica é extremamente poderosa para condensar a informação ao longo dos canais das entradas da imagem ou dos filtros em uma determinada camada. Para ilustrar, pode-se utilizar o exemplo de uma imagem de tamanho $10 \times 10 \times 3$ e um conjunto de filtros

de tamanho $5 \times 5 \times 3$. Considerando o *stride* como unitário, utilizando a expressão D_f - D_k + 1, a imagem final terá o tamanho de 6×6 e o filtro será movido 6×6 vezes na imagem inicial. Supondo a extração de 256 mapas de características e aplicando a Equação 2.17, haverá o total de $5 \times 5 \times 6 \times 6 \times 3 \times 256 = 691.200$ multiplicações. No caso de dividir a operação em duas etapas, o que ocorre é o seguinte, utilizando os mesmos parâmetros, aplicando a Equação 2.18, teriam-se na primeira etapa $5 \times 5 \times 6 \times 6 \times 3 = 3600$ multiplicações. Já na segunda etapa, considerando que se desejam extrair 256 mapas de características, teriam-se $256 \times 3 \times 6 \times 6 = 2808$ multiplicações. Ou seja, para esse caso, somando o número de multiplicações necessárias nas duas etapas tem-se o total de 6408 operações, enquanto utilizando a técnica de convolução convencional, esse valor seria dez vezes superior.

Também é possível expressar a redução de operações computacionais entre os processos combinando as Equações 2.19 e 2.17 formando a Equação 2.20. Dessa forma, pode-se calcular a redução computacional envolvida no processo de acordo com as especificações do tamanho do filtro e do número de mapas de características.

$$\frac{D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + N \cdot M \cdot D_f \cdot D_f}{D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f} = \frac{1}{N} + \frac{1}{D_k^2} \tag{2.20}$$

Um exemplo de arquitetura que utiliza essa técnica apresentada é a $MobileNet$. Essa família de redes neurais opera através de poucas camadas, o que lhe garante um tempo de resposta rápido e com baixa latência. Além disso, essa família de redes foi projetada para funcionar em dispositivos móveis, por isso apresenta um baixo custo de processamento e consequentemente maior economia de energia para o dispositivo.

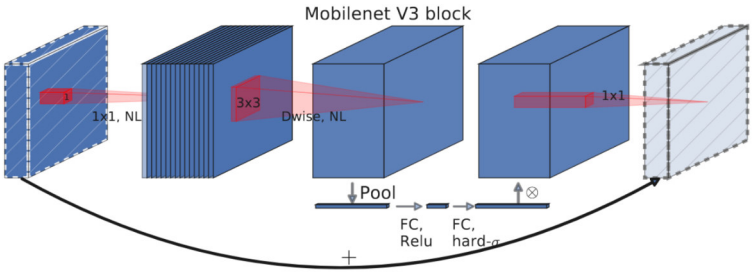
A arquitetura $MobileNet$ atualmente está na sua terceira versão. Ao longo de sua evolução, essa arquitetura tem sido modificada e na versão atual apresenta resultados com melhor acurácia e menor latência quando comparada aos modelos anteriores. Com o propósito de aprimorar a arquitetura, os autores propuseram um bloco convolucional que combina as técnicas convolucionais *pointwise* e *depthwise* utilizadas na primeira versão com as técnicas de conexões residuais (HE *et al.*, 2016).

O intuito dos autores ao combinar esses elementos é proporcionar uma arquitetura com camadas de baixa densidade. Entretanto, reduzir a dimensionalidade das camadas por si só não costuma apresentar bons resultados. Dessa forma, para obter uma boa precisão sem aumentar a densidade dos dados, os autores apresentaram um bloco de convoluções chamado de *Bottleneck Residual Block*.

A função desse bloco é reduzir a quantidade de dados que estão passando pela rede, conforme o nome sugere, o bloco gera um "gargalo" de informações em sua saída. Esse conjunto de convoluções é composto por três etapas: primeiro aplica-se uma camada de convolução *pointwise* aos dados recebidos, esse estágio também é chamado de camada de

expansão, nessa camada aumenta-se o volume dos dados. Em seguida, há uma camada *depthwise*, a qual efetua uma filtragem dos dados expandidos. Por fim na última parte do bloco, uma camada tipo *pointwise* "compacta" os dados e reduzindo a sua dimensionalidade. Além das operações mencionadas, o bloco conta com uma conexão residual entre a primeira e última etapa. Na Figura 8 está representado o bloco em questão. Para mais detalhes sobre o funcionamento do bloco e da arquitetura, consultar os artigos originais (SANDLER *et al.*, 2018; HOWARD *et al.*, 2019)

Figura 8 – Bottleneck Residual Block MobileNetV3 - Nessa Figura, é possível observar que quando os dados chegam ao bloco entre a primeira e segunda camada sofrem uma espécie de descompressão, aumentando seu volume. Da segunda para a terceira camada, os dados passam por um processo de filtragem com objetivo de identificar os padrões nos dados.E da terceira para a quarta camada, efetua-se um processo de *pooling*. Por fim, há a etapa de redução de dimensionalidade ou "compressão". Nessa última etapa, os dados voltam a ter as mesmas dimensões da entrada no bloco, a entrada e a saída do bloco são conectadas através de uma conexão residual.



Fonte: Howard *et al.* (2019).

Tabela 1 – Estrutura da MobileNetV3 Large.

Entrada	**Operador**	**Dimensões dos filtros**
224x224x3	conv2d	3x3x16
112x112x16	bneck block	3x3x16
112x112x16	bneck block	3x3x24
56x56x24	bneck block	3x3x24
56x56x24	bneck block	5x5x40
28x28x40	bneck block	5x5x40
28x28x40	bneck block	5x5x40
28x28x40	bneck block	3x3x80
14x14x80	bneck block	3x3x80
14x14x80	bneck block	3x3x80
14x14x80	bneck block	3x3x80
14x14x80	bneck block	3x3x112
14x14x112	bneck block	3x3x112
14x14x112	bneck block	5x5x160
7x7x160	bneck block	5x5x160
7x7x160	bneck block	5x5x160
7x7x160	conv2d	1x1x960
7x7x960	pool	7x7
1x1x960	conv2d	1x1x1280
1x1x1280	conv2d	1x1xk

Fonte: Howard *et al.* (2019).

Tabela 2 – Estrutura da MobileNetV3 Small.

Entrada	**Operador**	**Dimensões dos filtros**
224x224x3	conv2d	3x3x16
112x112x16	bneck block	3x3x16
56x56x16	bneck block	3x3x24
28x28x24	bneck block	3x3x24
28x28x24	bneck block	5x5x40
14x14x40	bneck block	5x5x40
14x14x40	bneck block	5x5x40
14x14x40	bneck block	5x5x48
14x14x48	bneck block	5x5x48
14x14x48	bneck block	5x5x96
7x7x96	bneck block	5x5x96
7x7x96	bneck block	5x5x96
7x7x96	conv2d	1x1x576
7x7x576	pool	7x7
1x1x576	conv2d	1x1x1024
1x1x1024	conv2d	1x1xk

Fonte: Howard *et al.* (2019).

2.4.5 Detecção de objetos

A detecção de objetos é um campo da visão computacional que tem por objetivo identificar a presença de objetos em imagens ou vídeos. Nas sessões anteriores, comentou-se sobre a tarefa de classificação de objetos utilizando redes convolucionais. Contudo, quando o assunto é referente à detecção de objetos, o sistema deve não só distinguir a classe de determinado objeto em uma imagem, mas, também localizar sua posição na imagem em questão. Isso posto, é necessário ressaltar que muitas vezes a imagem não contém apenas um único objeto, isto é, uma imagem pode conter diversos objetos da mesma classe ou mesmo de classes distintas.

O objetivo do processo de classificação convencional, por exemplo, utilizando redes convolucionais, é extrair características por meio das primeiras camadas e ao final do processo as últimas camadas da rede são responsáveis por efetuar uma classificação. Na tarefa de detecção de objetos funciona de maneira similar, contudo, ao final do processo, essa tarefa indica não somente a classificação do objeto, mas também sua localização na imagem. Por isso, além da camada $softmax$ que é comumente utilizada no método de classificação convencional, adiciona-se uma segunda camada de saída à rede. A primeira camada de saída tem a função de indicar a classificação de determinado objeto, enquanto a segunda camada de saída terá como função determinar a localização desse objeto. Por isso, essa primeira camada conterá informações relativas às classes do problema, enquanto a segunda camada conterá informação que representa a localização do objeto baseada nos $pixels$ x e y da imagem, essa localização é chamada de *bounding box*.

Como mencionado no primeiro parágrafo, nem sempre existe apenas um objeto na imagem. Nesse caso, é necessário tratar a possibilidade de existir mais de um objeto na imagem, por isso será necessário configurar a rede para realizar mais detecções. Para resolver essa problemática, repete-se a estrutura de detecção original de localização e de classificação de acordo com um determinado número máximo de detecções por imagem, a ser definido conforme o projeto.

Existem dois métodos muito difundidos para detecção de objetos em imagens, um baseado em janelas deslizantes e outro baseado em propostas de regiões. O primeiro método mencionado é comumente chamado de detector de uma etapa, enquanto o segundo, de detector de duas etapas. Como o próprio nome sugere, o detector de uma etapa executa a detecção em apenas um estágio. Já os detectores de duas etapas, precisam primeiro utilizar algum método para propor regiões e logo após esse processo, eles realizam as detecções. Os detectores de duas etapas apresentam altas taxas de acurácia considerando a localização e também a classificação, enquanto os detectores de um estágio destacam-se no quesito velocidade de detecção. O exemplo mais representativo na literatura de detector de dois estágios é o Faster R-CNN (REN *et al.*, 2015). Como exemplos de detectores de um estágio, podem-se citar os sistemas YOLO (You Only Look Once) e SSD (Single Shot

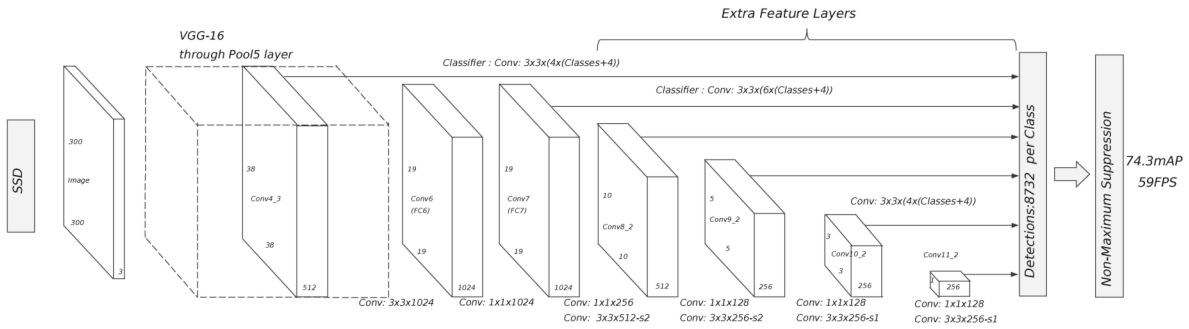
Multibox Detector) (REDMON *et al.*, 2016; LIU *et al.*, 2016).

Single Shot Multibox Detector

A abordagem utilizada no SSD é baseada em uma rede *feed-forward*. De acordo com o número de objetos a serem detectados, criam-se uma coleção de delimitadores, chamados de *bounding boxes*. Para cada delimitador, produz-se uma estimativa da presença de uma das classes do problema. Após as detecções, é aplicado o algoritmo de *non-maximum suppression* que descarta os *bounding boxes* irrelevantes e produz a detecção final. O modelo SSD utiliza como base uma arquitetura de Rede Neural Convolucional. No trabalho original, utiliza-se a VGG, no entanto, esse *backbone* pode ser substituído por outra arquitetura, como MobileNet ou ResNet, entre outras.

Esse detector conta com algumas particularidades que lhe proporcionam versatilidade e agilidade na detecção, sem comprometer a acurácia das suas predições. No trabalho em que se apresentou esse detector, os desenvolvedores ressaltaram algumas características do modelo. Entre elas, pode-se destacar a adição de algumas camadas extras que são acopladas a arquitetura base utilizada, no trabalho a arquitetura usada foi a VGG-16. Essas camadas extras seguem um padrão em que seu tamanho é progressivamente reduzido, de forma a proporcionar uma detecção em múltiplas escalas, conforme se pode observar na Figura 9. Segundo o autor, o objetivo desse método é reduzir o *overfitting*. Além de adicionar camadas extras a arquitetura original, o autor propõe fazer uma espécie de "*bypass*" entre as camadas. Para isso, ele interliga as saídas das camadas intermediárias diretamente ao classificador no final da rede.

Figura 9 – Esse exemplo apresentado pelo autor utiliza como *backbone* base a arquitetura VGG-16. Nessa base, adicionaram-se algumas camadas extras que, vão progressivamente diminuindo de tamanho para auxiliar a detecção. Observa-se que todas as saídas das camadas, tanto as da arquitetura base, quanto as saídas das camadas extras estão ligadas em paralelo diretamente com a camada de detecção, pulando algumas ligações intermediárias. Após a camada de detecção, aplicou-se a etapa de *non-maximum suppression*. Essa etapa suprime os *bounding-boxes* com *score* mais baixos.



Fonte: Liu *et al.* (2016).

Durante o treinamento do sistema SSD, é necessário determinar quais *bounding-boxes* aproximam-se mais da anotação da posição real do objeto e treinar a rede de acordo com isso. Para cada predição, varia-se o conjunto de delimitadores padrão de forma a diversificar aspectos relativos a sua forma, localização e a escala do delimitador de forma a abranger objetos em distintas posições. Por isso, para cada um dos delimitadores do conjunto padrão de *bounding-boxes*, calcula-se o índice de interseção sobre a união (IoU). Para isso, utiliza-se a anotação que corresponde a localização na imagem do objeto original e seleciona a predição que possui o maior índice. A interseção sobre a união também é chamada de índice de Jaccard e pode ser expressa pela Equação 2.21:

$$J(A, B) = \frac{A \cap B}{A \cup B} \tag{2.21}$$

Segundo o autor, a função objetivo utilizada no sistema SSD é derivada do sistema *MultiBox* (ERHAN *et al.*, 2014), no entanto, a função no SSD foi otimizada para lidar com múltiplas categorias. A função objetivo desse sistema é composta pelo *localization loss* (loc) e pelo *confidence loss* (conf), conforme a Equação 2.23. A *localization loss* (loc) é responsável por corrigir o erro entre a posição da anotação original e a posição predita pela rede através do *bounding-box*. Para isso, consideram-se apenas as combinações positivas, seja X_{ij}^{p} um indicador para a equivalência entre o i-ésimo *bounding-box* com a j-ésima anotação de uma classe p na imagem, caso o IoU seja maior que 0,5 $X_{ij}^{p} = 1$, caso contrario X_{ij}^{p}=0. O *localization loss* é o mesmo *Smooth L1 loss* utilizado na YOLO (REDMON *et*

al., 2016), conforme a Equação 2.22:

$$L_{loc}(x,l,g) = \sum_{i \in Posm \in ex,ey,w,h}^{N} \sum X_{ij}^{p} smooth_{L1}(l_i^m - \hat{g}_j^m)$$

$$\begin{aligned} \hat{g}_j^{cx} &= \frac{(g_j cx - d_i cx)}{d_i^w} \\ \hat{g}_j^{cy} &= \frac{(g_j cy - d_i cy)}{d_i^h} \\ \hat{g}_j^{w} &= log(\frac{g_j^w}{g_i^w}) \\ \hat{g}_j^{h} &= log(\frac{g_j^h}{g_i^h}) \end{aligned} \tag{2.22}$$

Onde o parâmetro l representa a marcação predita pela rede e o parâmetro g representa a marcação original na imagem, enquanto os parâmetros cx e cy correspondem ao desvio entre o *bounding-box* d em relação à largura w e a altura h.

Já a métrica *confidence loss* (conf) representa o erro calculado ao realizar uma predição. Para cada combinação positiva, a função objetivo é ajustada de acordo com o escore de avaliação para aquela respectiva classe. Já para as combinações negativas, penaliza-se a classe zero, que corresponde a detecção do "fundo" na imagem. Para isso, aplica-se a função $softmax$ considerando as pontuações obtidas na classificação conforme a Equação 2.23:

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} X_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \tag{2.23}$$

Onde $\hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p (c_i^p)}$ e N é o número total combinações positivas.

A função objetivo geral é descrita pela Equação 2.24:

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)) \tag{2.24}$$

Onde N é o número de *bounding-box* que representam uma combinação positiva e o parâmetro α é um peso a ser multiplicado pelo *localization loss*.

3 METODOLOGIA

3.1 Materiais

3.1.1 Bases de dados

TIMIT Acoustic-Phonetic Continuous Speech Corpus

A base TIMIT é uma base de áudios em inglês, projetada para fornecer dados para estudos acústicos-fonéticos e para auxiliar no desenvolvimento e avaliação de sistemas automáticos de reconhecimento de fala. A base consiste em áudios de 630 falantes, abrangendo os oito principais dialetos do inglês americano. Cada falante gravou dez frases foneticamente ricas, gerando um total de 6300 sentenças, ou 5,4 horas. Os enunciados foram gravados utilizando a resolução de 16 bits e as frequências vão até 16 kHz. Para cada enunciado gravado em áudio, existem transcrições ortográficas, fonéticas e de palavras, as quais todas possuem anotação temporal. Cada um dos falantes pronunciou dez sentenças e estas foram divididas em três grupos: sa, sx e si. Entre essas sentenças, duas são do grupo "sa" e correspondem a sentenças comuns a todos os participantes. Outras cinco são "sx" e consistem em um grupo com 450 sentenças foneticamente balanceadas selecionadas pelo Massachusetts Institute of Technology (MIT). Já as últimas três são do tipo "si", que correspondem a sentenças selecionadas aleatoriamente de fontes diversas, como obras da literatura.

Considerando o estado original da base, existem um total de 64 categorias de fonemas distintas. Contudo, alguns fonemas são imperceptíveis e podem ser considerados como silêncio e outros são muito semelhantes entre si, por isso podem ser agrupados. Objetivando reduzir a quantidade de fonemas, Lee e Hon (1989) propuseram um modelo onde foram selecionados 48 fonemas e removeram todos os fonemas da categoria "Q". Além disso, também foram identificados 15 alofones (sons que representam o mesmo fonema). Na Tabela 3, estão listados os 48 fonemas contidos originalmente na base, os fonemas selecionados e os fonemas agrupados.

Após realizar as remoções e agrupamentos necessários, restaram 39 categorias de fonemas distintas. Entre essas 39 categorias, os fonemas ainda devem ser divididos em sete grupos, nos quais as substituições dentro do grupo não são consideradas como erros, são eles: sil, cl, vcl,epi, el, l, en, n, sh, zh, ao, aa, ih, ix, ah, ax. No trabalho, os autores ainda sugerem que não sejam consideradas as sentenças identificadas com o tipo "SA", pois em alguns contextos, essas sentenças introduzem um viés em alguns fonemas, gerando um alto resultado, porém artificial (LEE; HON, 1989).

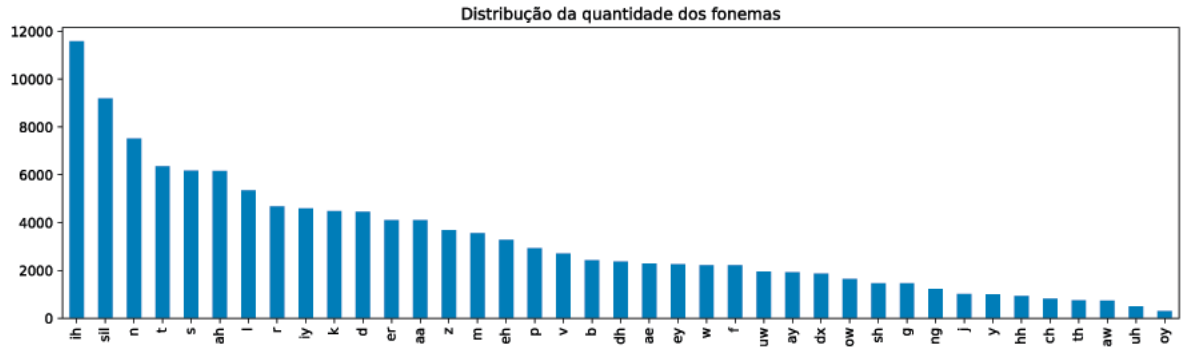
Tabela 3 – Tabela de fonemas da base TIMIT modificada.

Phone	**Example**	**Folded**	**Phone**	**Example**	**Folded**
iy	beat		en	button	
ih	bit		ng	sing	eng
eh	bet		ch	church	
ae	bat		jh	judge	
ix	roses		dh	they	
ax	the		b	bob	
ah	butt		d	dad	
uw	boot	ux	dx	(butter)	
uh	book		g	gag	
ao	about		p	pop	
aa	cot		t	tot	
ey	bait		k	kick	
ay	bite		z	zoo	
oy	boy		zh	measure	
aw	bough		v	very	
ow	boat		f	fief	
l	led		th	thief	
el	bottle		s	sis	
r	red		sh	shoe	
y	yet		hh	hay	hv
w	wet		cl(sil)	(unvoiced closure)	pcl,tcl,kcl,qcl
er	bird	axr	vcl(sil)	(voiced closure)	bcl,dcl,gcl
m	mom	em	epi(sil)	(epinthetic closure)	
n	non	nx	sil	(silence)	h, h, pau

Fonte: Lee e Hon (1989).

Após executar o processo mencionado anteriormente, reduz-se a quantidade total de fonemas para 39 categorias. A partir de um conjunto de fonemas, é possível contabilizar a quantidade de amostras de cada fonema, como ilustrado na Figura 10. É notável que as amostras estão desbalanceadas e isso pode acabar influenciando o desempenho do modelo do algoritmo de reconhecimento para determinadas classes.

Figura 10 – Distribuição da quantidade de amostras para cada fonema nos exemplos de treino da base TIMIT.

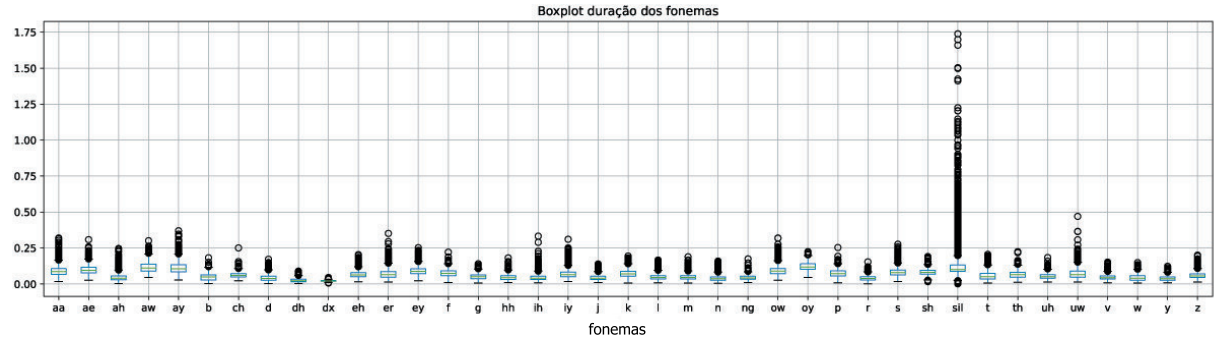


Fonte: Autor.

Outro fator importante a considerar-se é a duração dos fonemas. As sentenças são faladas por diversos locutores pertencentes a diversas localidades com diferentes sotaques. Esses fatores impactam diretamente na pronúncia dos fonemas. Dependendo do sotaque e da velocidade da fala do locutor, fonemas iguais podem ser pronunciados de maneira distinta.

Conforme se pode verificar na Figura 11, existem muitos *outliers* em relação à duração dos fonemas. Não é uma alternativa apenas descartá-los, considerando que estes *outliers* possuem propriedades representativas da fala de alguns dos locutores e seus sotaques. Sendo assim, sua remoção pode acarretar na construção de um modelo menos robusto, que identifica uma menor quantidade de sotaques e consequentemente atenderá um menor número de falantes.

Figura 11 – Distribuição da duração das amostras (em segundos) para cada fonema nos exemplos de treino da base TIMIT.



Fonte: Autor.

LibriSpeech: ASR Corpus based on public domain audiobooks

A LibriSpeech é uma base de dados gratuita construída a partir dos audiobooks

disponibilizados no projeto LibriVox[1]. O projeto LibriVox conta com uma vasta biblioteca de audiobooks contendo cerca de 8000 obras literárias disponibilizadas gratuitamente (PANAYOTOV *et al.*, 2015). A base de áudios LibriSpeech contém cerca de 1000 horas de fala com frequências na faixa de 16 kHz. Os arquivos estão divididos em três subsets, com tamanhos de 100, 360 e 500 horas. Diferente da base TIMIT, as anotações dessa base foram verificadas automaticamente utilizando algoritmos de alinhamento e de identificação de locutores. Os autores indicaram que através dos algoritmos puderam fazer uma rápida inspeção no alinhamento dos textos dos enunciados e também excluir áudios que possuíssem múltiplos locutores. Através dessas aplicações, também foi possível identificar informações relativas ao gênero e descartar uma pequena quantidade de gravações que possuíam problemas de qualidade. Os autores informam que balancearam os falantes de acordo com os dados disponíveis, dessa forma, garantindo o mesmo tempo de gravação para um cada um dos falantes, bem como balanceamento da quantidade de locutores de acordo com seus respectivos gêneros.

Essa base originalmente não possui alinhamento fonético conforme a base TIMIT. Por isso, para poder utilizar essa base neste trabalho, foi necessário utilizar uma ferramenta para forçar o alinhamento fonético (MCAULIFFE *et al.*, 2017). Esse algoritmo possui alguns modelos pré-treinados para determinadas linguagens[2]. Os arquivos contendo as informações relativas às anotações fonéticas e de palavras foram obtidos através do trabalho de Lugosch *et al.* (2019)[3]. Para mais detalhes sobre o algoritmo de alinhamento, consultar os trabalhos mencionados.

Neste trabalho, optou-se por utilizar o subconjunto da base o qual contém 100 horas de gravações. Ainda assim, selecionou-se apenas uma parte das gravações para complementar as bases de amostras utilizadas durante o treino.

3.1.2 Classificador

Para efetuar os experimentos neste trabalho, optou-se por utilizar o detector de objetos SSD em combinação com a arquitetura de rede MobileNet. Essa foi uma decisão de projeto, levando em consideração que o modelo após o treinamento possui um baixo custo de processamento e isso permite que esse funcione em um dispositivo com baixa capacidade computacional e de modo *offline*, sem necessitar de um servidor externo para efetuar os processamentos.

Vale lembrar que esse algoritmo foi desenvolvido anteriormente em outro trabalho (HOWARD *et al.*, 2019) tendo sido aplicado ao desafio de classificação ImageNet (RUSSAKOVSKY *et al.*, 2015b). Neste trabalho, aproveitou-se essa arquitetura aplicada a tarefa

[1] https://librivox.org
[2] https://montreal-forced-aligner.readthedocs.io/en/latest/pretrained_models.html
[3] https://zenodo.org/record/2619474.X8MM4JZ7lhF

de reconhecimento de fonemas.

Conforme mencionado na Seção 3, a arquitetura MobileNet é responsável por efetuar a parte convolucional do processo e possui duas versões: uma chamada MobileNetV3-Large e outra, MobileNetV3-Small. A versão *large* consiste na arquitetura completa da rede e a versão *small*, que é uma versão reduzida da mesma arquitetura, a qual utiliza uma menor quantidade de parâmetros.

Para efetuar os experimentos, utilizou-se o *framework* TensorFlow e alguns algoritmos foram executados de forma *online* através da plataforma Google Colab e outros de forma *offline* em um computador do laboratório da UFMA.

O TensorFlow é uma plataforma de código aberto para soluções de aprendizado de máquina. Essa ferramenta possuí uma ampla variedade de recursos de última geração. Dessa forma, proporciona uma implementação de projetos de aprendizado de máquina de maneira mais fluída e ágil. O sistema é flexível e pode ser usado para implementar uma ampla variedade de algoritmos, incluindo algoritmos de treinamento e inferência para modelos de redes neurais profundas. Por isso, o TensorFlow tem sido utilizado para realizar pesquisas e implantar sistemas de aprendizado de máquina em produção em muitas áreas de ciência da computação e outros campos, incluindo reconhecimento de voz, visão computacional, robótica, recuperação de informações, processamento de linguagem natural, extração de informações geográficas e descoberta computacional de medicamentos (ABADI *et al.*, 2015).

O Google Colab é um dos produtos do pacote do Google Research que permite a execução de código em linguagem Python através do navegador. O Colab é um serviço que não requer configuração para uso e, ao mesmo tempo, fornece acesso gratuito a recursos de computação, incluindo GPUs. Embora essa ferramenta possibilite a execução de qualquer projeto (somente em linguagem Python), ela é especialmente recomendada para projetos de aprendizado de máquina, análise de dados e educação [4].

3.2 Métodos

Divisão em palavras

Originalmente os áudios foram gravados em formato de enunciado, isto é, o locutor fala uma frase inteira correspondente a uma sentença. Apesar disso, a base contém anotações temporais da posição temporal de cada palavra falada e de cada fonema ao longo de cada gravação. Dessa forma, é possível dividir a base em palavras ou até mesmo em fonemas isolados. Considerando essa possibilidade, desmembraram-se os enunciados iniciais em palavras individuais, dessa maneira se aumenta o número de amostras disponíveis

[4] https://colab.research.google.com/

para o treinamento e diminui a quantidade de fonemas em cada amostra. Além disso, também proporciona uma detecção mais ágil, pois utiliza unidades menores para efetuar a detecção, o que permitirá ao sistema fornecer uma resposta mais rápida quando comparada à abordagem de reconhecer o enunciado completo.

Isto posto, efetuou-se a divisão dos enunciados em palavras, para isso, construiu-se um algoritmo que verifica a anotação temporal das palavras para cada enunciado. Cada palavra da oração é recortada e salva em um arquivo distinto. Após isso, foi necessário ler o arquivo que contém a informação temporal de cada palavra e separá-lo de acordo com a nova palavra, também é necessário corrigir o tempo da anotação fonética. Essa correção foi necessária, pois ao separar a palavra do enunciado original, o tempo da ocorrência daquela palavra no arquivo original é alterado. Para exemplificar, suponha uma gravação hipotética da oração "Eu vou a escola" com duração de três segundos. A palavra "escola" é pronunciada durante os segundos 2 a 3, então a sílaba "es" seria pronunciada de 2,0 a 2,3 s, a sílaba "co" de 2,3 a 2,6 s e a "la" de 2,6 a 2,9 s. Ao recortar a palavra e criar um novo arquivo, a gravação iniciará no segundo 0, ou seja, nesse exemplo fictício, a palavra não será mais pronunciada ao final da gravação original, no tempo de 2 segundos, como mencionado. Após esse processo, a palavra "escola" fará parte de uma nova gravação, com início no tempo 0. Em decorrência disso, a sílaba "es" será pronunciada de 0 a 0,3 s, a sílaba "co" de 0,3 a 0,6 s e a sílaba "la" de 0,6 a 0,9 s.

3.2.1 Divisão fonética

Ao projetar um sistema de reconhecimento de voz, é necessário definir o tipo de unidade de saída. Para isso, deve-se considerar se o sistema será de amplo vocabulário ou identificará apenas um conjunto restrito de palavras. Para o primeiro caso, considerando o total de palavras que existem em uma linguagem, seria computacionalmente intratável criar um sistema que possua como saída a quantidade total das palavras existentes na linguagem. Em decorrência disso, opta-se por reconhecer unidades menores, como fonemas ou caracteres, que possuem uma quantidade menor de representações.

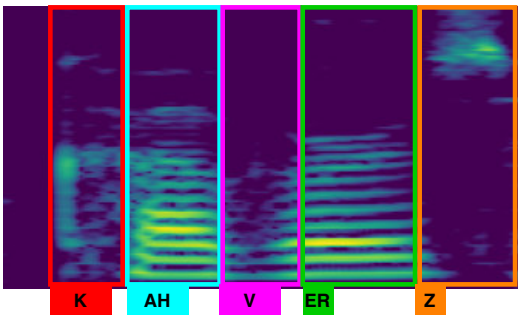
Um sistema de reconhecimento pode ser baseado em diversas unidades. Na abordagem de reconhecimento de palavras, costuma-se ter um sistema cujo vocabulário é restrito, pois essa categoria de sistema é condicionado a aprender apenas um grupo pré-determinado de palavras. Já as abordagens de amplo vocabulário utilizam unidades de reconhecimento mais granulares, como os fonemas ou os caracteres. Dessa forma, estes sistemas podem proporcionar o reconhecimento de diversas palavras e até mesmo extrapolar o modelo e identificar palavras que nunca foram apresentadas ao sistema.

Neste trabalho, optou-se por utilizar como unidade de reconhecimento os fonemas. A base de áudios TIMIT contém todos os enunciados de fala anotados foneticamente. Isto é, para cada amostra de áudio, há um documento em texto o qual contém anotações

relativas à posição temporal de cada fonema presente no enunciado. Dessa maneira, é possível construir um algoritmo que recebe as informações temporais de início e fim de cada fonema e posteriormente separar os áudios em fonemas ou palavras e agrupá-los de acordo com a sua classificação fonética.

Para cada arquivo de fala contendo uma palavra, foi criado um arquivo auxiliar que contém a informação temporal dos fonemas contidos naquela palavra. Ao realizar a conversão do formato de áudio para o espectrograma, também foi realizada a anotação fonética levando em consideração a posição dos fonemas em *pixels* na imagem do espectrograma. A Figura 12 ilustra a constituição das anotações fonéticas na palavra "covers". Para cada amostra de áudio, foi criado um espectrograma e um arquivo de marcação que contém a informação da posição de cada fonema naquela imagem. Dessa forma, foi possível visualizar as marcações através de $software$s utilizados para fazer anotação em imagens, como exemplo o $LabelImg$. Através do arquivo de marcações, facilitou-se o processo de conversão das imagens e o processo de anotações para o padrão utilizado durante o treinamento dos algoritmos de detecção de objetos.

Figura 12 – Exemplo de anotação fonética no espectrograma de uma gravação de áudio da palavra "covers".



Fonte: Autor.

3.2.2 Data Augmentation

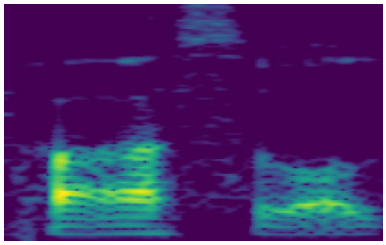
A maioria dos processos que envolvem a utilização de algoritmos de redes profundas necessitam de um número extenso de amostras para seu treinamento, tais amostras devem possuir a maior diversidade de exemplos possível. Contudo, muitas vezes o processo de obtenção de dados é muito custoso. Assim, uma forma de reduzir os custos na obtenção de dados é através de técnicas de aumento de dados (*Data Augmentation*). Para lidar com imagens convencionais, é muito comum aplicarem-se técnicas de deformação da imagem como esticar ou encolher seu tamanho, rotacioná-las no sentido horário ou anti-horário, ampliar ou retrair as imagens, modificar a intensidade e balanço das cores e até mesmo remover um pedaço aleatório das imagens.

Neste trabalho, no entanto, existe uma certa peculiaridade com relação às imagens, pois elas correspondem a uma distribuição dos sinais ao longo do tempo e em decorrência disso, nem todas as técnicas habitualmente utilizadas para aumento de dados em situações convencionais são válidas. Por exemplo, espelhar horizontalmente a imagem pode inverter totalmente o sentido da pronúncia de uma palavra. Para ilustrar, seja o exemplo das palavras "abacate" e sua versão invertida "tecaaba". Embora possuam o mesmo grupo de fonemas, não possuem a mesma pronúncia, por isso não são equivalentes.

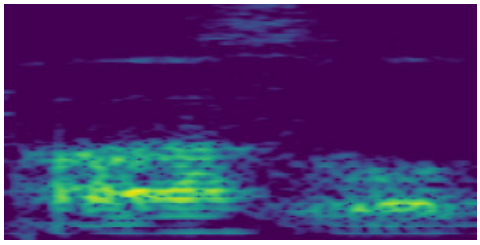
Em decorrência dessa problemática, foi necessário buscar por técnicas que sejam compatíveis com o aumento de dados no contexto de áudios. Para isso, inspirou-se em um tutorial disponibilizado pelo $Tensorflow$[5]. Os procedimentos de esticar ou encolher a imagem podem ser aplicados nesse contexto visto que seu efeito no áudio é semelhante a aumentar ou diminuir a velocidade da gravação. Na Figura 13, é possível observar o efeito de alterações no espectrograma em função do aumento e diminuição da velocidade da gravação do áudio original.

Figura 13 – Todas as Figuras (a), (b) e (c) são espectrogramas da palavra "possible". A Figura (b) ilustra o efeito da redução da velocidade da fala, assim o tempo de fala é maior. A Figura (c) ilustra o efeito do aumento da velocidade da fala, dessa forma, o tempo para pronunciar a mesma palavra é menor.

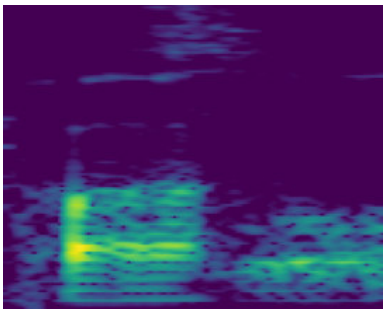
a) Espectrograma com a velocidade original.



b) Espectrograma em velocidade reduzida.



c) Espectrograma em velocidade acelerada.



Fonte: Autor.

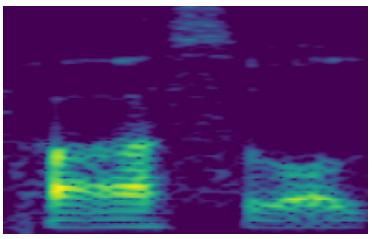
Outra técnica utilizada para o aumento de dados foi o deslocamento do áudio da gravação original por meio da adição de um trecho de ruído no início do áudio. Dessa

[5] https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/speech_commands

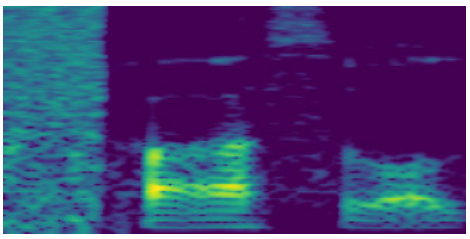
forma, a informação que corresponde a fala tem seu tempo atrasado em alguns segundos. A Figura 14 ilustra um exemplo que apresenta o efeito dessa técnica no espectrograma.

Figura 14 – As Figuras (a) e (b) são espectrogramas da palavra "possible". Enquanto a Figura (a) ilustra o espectrograma do áudio em sua forma original, a Figura (b) mostra o efeito da adição de um trecho de ruído no início da gravação de forma a deslocá-la no tempo.

a) Espectrograma original.



b) Espectrograma com áudio deslocado da origem.

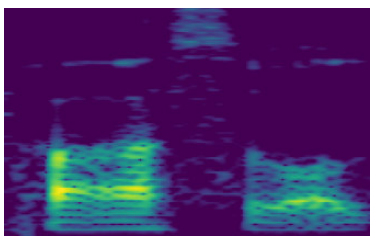


Fonte: Autor.

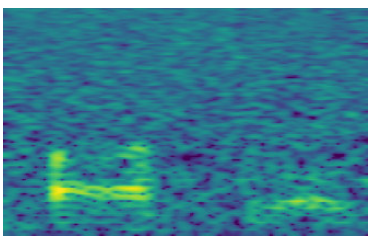
Também aplicou-se um ruído ao longo de toda a gravação original. Para isso, pode-se gerar um ruído branco e multiplicá-lo por um coeficiente de intensidade, por exemplo, 0,05 e então somá-lo a gravação inicial. Além de adicionar um ruído branco, também é possível utilizar um banco de ruídos, por exemplo, uma gravação de ruídos de fundo como pessoas sussurrando, ou de gravações que possuem elementos sonoros de ruas de uma cidade, contendo som de motores, buzinas de carros, sons de construções etc. A Figura 15 ilustra a aplicação de ruído branco na gravação de um áudio que contém a palavra "possible".

Figura 15 – As Figuras (a) e (b) são espectrogramas da palavra "possible". Enquanto a Figura (a) ilustra o espectrograma do áudio em sua forma original, a Figura (b) mostra o efeito da adição de um ruído branco em toda a extensão do áudio.

a) Espectrograma original.



b) Espectrograma adicionado de ruído branco.

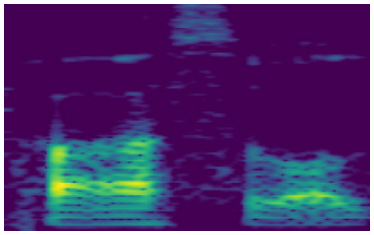


Fonte: Autor.

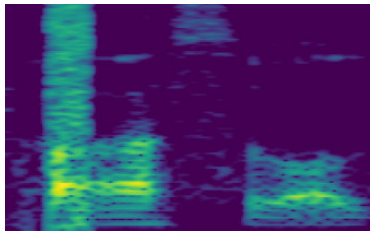
Outra técnica utilizada foi a adição de ruído localizado em determinados trechos das gravações de áudio. Nesse caso, de acordo com a posição dos fonemas no áudio e sua duração, aplicou-se um ruído localizado de forma a desfigurar um dos fonemas, conforme ilustrado na Figura 16.

Figura 16 – As Figuras (a) e (b) são espectrogramas da palavra “possible”. Enquanto a Figura (a) ilustra o espectrograma do áudio em sua forma original, a Figura (b) mostra o efeito da adição de um ruído localizado em um local aleatório do áudio.

a) Espectrograma original



b) Espectrograma adicionado de ruído localizado



Fonte: Autor.

3.2.3 Métricas

Phone Error Rate (PER)

A métrica *Phone Error Rate* (PER) é semelhante à métrica *Word Error Rate* (WER). Ambas são métricas amplamente utilizadas para avaliar o desempenho de sistemas de reconhecimento de fala ou de sistemas de tradução. A métrica PER considera quatro parâmetros importantes para mensurar a desempenho do sistema de reconhecimento, são elas as *substituições* (S), *exclusões* (D) e *inserções* (I), dividas pelo número total de predições (N), conforme ilustra a Equação 3.1. Para exemplificar, suponha que a palavra seja “escola” e o sistema indique durante o reconhecimento que a palavra correta é “iscola”, esse caso trata-se de uma *substituição* do “e” pelo “i”. Outra possibilidade seria caso a detecção fosse “escol”, nesse cenário teria-se uma *exclusão* da letra “a”. Um exemplo de *inserção* seria o caso do detector identificar a palavra “esxcola”.

$$PER = \frac{S + D + I}{N} \tag{3.1}$$

Mean Average Precision (mAP)

O mAP é uma métrica amplamente utilizada em detecção de objetos. Essa métrica é calculada através da interseção sobre a união (IoU) das áreas da marcação real do objeto na imagem e a marcação predita pelo algoritmo durante a classificação. Em geral, considera-se que a detecção da localização do objeto está correta caso o IoU seja maior ou igual a 0,5. Se houver detecção e o IoU for maior ou igual a 0,5, a detecção é considerada um Verdadeiro Positivo (TP), caso contrário será um Falso Positivo (FP). Caso não haja detecção, considera-se que o modelo falhou em detectar o objeto e dessa forma, classifica-se a detecção como Falso Negativo (FN). O mAP é a média da precisão do modelo considerando um determinado nível de confiança, representado pelo IoU. Para

calcular a precisão do modelo (*average precision*), utiliza-se a Equação 3.2:

$$AP = \frac{VerdadeirosPositivos}{VerdadeirosPositivos + FalsosPositivos} \tag{3.2}$$

Para calcular o mAP, considera-se a média das precisões (AP) de todas as classes, conforme a Equação 3.3:

$$mAP = \frac{1}{N} \sum_{j=1}^{N} AP_j \tag{3.3}$$

4 RESULTADOS E DISCUSSÃO

Neste capítulo, são apresentados os resultados obtidos durante este estudo. É importante salientar que os experimentos relatados nesta seção consistiram apenas em um subgrupo reduzido, considerando o total de experimentos realizados durante o trabalho.

4.1 Treinamento do modelo

Como apresentado na Seção 3, antes de efetuar qualquer procedimento envolvendo o uso de redes neurais, foi necessário efetuar uma série de pré-processamentos envolvendo os dados utilizados. Entre estes procedimentos, aplicou-se a técnica de aumento de dados aos arquivos de áudio e nessa ocasião, verificou-se que nem todas as técnicas propostas anteriormente foram eficientes para a melhora do desempenho do sistema.

Inicialmente, treinou-se o modelo durante 100 mil iterações, onde cada iteração foi responsável por executar um conjunto do *batch size*. Para fins explicativos, caso o *batch size* possuísse o tamanho de 128 e o *dataset* com tamanho de 512, cada época seria composta por quatro iterações. Para cada treinamento, aplicou-se uma das técnicas de aumento de dados individualmente com o objetivo de identificar qual sua influência e a eficácia no treinamento do modelo. A Tabela 4 mostra as técnicas de aumento de dados utilizadas neste trabalho e suas respectivas acurácias.

Para evitar alterações na base de teste, aplicaram-se as técnicas de aumento de dados (Subseção 3.2.2) apenas na base de treino, além disso, selecionaram-se aleatoriamente as amostras nas quais a técnica seria aplicada. Para as técnicas de ruído e de ruído localizado, o nível de ruído usado variou de 1% a 5%. Assim, cada amostra recebeu aleatoriamente um valor específico de ruído dentro dessa faixa. A técnica de ruído localizado consiste em selecionar apenas um trecho do áudio e aplicar o ruído localizado no intervalo definido. O tamanho do trecho também é escolhido aleatoriamente considerando cada amostra. Para a técnica de deslocamento de áudio, o tamanho do segmento que o áudio seria deslocado da origem também foi definido aleatoriamente para cada amostra. Já para a técnica de alteração de velocidade, usaram-se dois intervalos: até 20% ou até 30%, tanto para aumentar, quanto para reduzir o tempo de fala. Para cada amostra, o valor da redução e do aumento da velocidade foi sorteado considerando os limites definidos.

Conforme a Tabela 4, é notável que cada técnica de aumento de dados proporciona um efeito distinto no modelo. É importante salientar que houve variação nos resultados apresentados na Tabela 4, devido aos processos não determinísticos envolvidos tanto no treinamento das RNAs (HAYKIN, 2007), quanto nas técnicas de aumento de dados. Os valores apresentados nessa tabela correspondem aos resultados obtidos durante alguns dos testes, considerando a aplicação individual das técnicas de aumento de dados.

Tabela 4 – Resultados da aplicação isolada das técnicas de aumento de dados durante o treinamento considerando 100 mil iterações.

Técnica de aumento de dados	**mAP (0.5IOU)**
Base Original	0,59
Ruído	0,56 - 0,57
Deslocamento	0,65 - 0,67
Velocidade (+ /- 20%)	0,60 - 0,62
Velocidade (+ /- 30%)	0,60 - 0,61
Ruído localizado	0,61 - 0,63

Fonte: Autor.

Observando a Tabela 4, nota-se que os treinamentos utilizando os dados após as aplicações das técnicas de deslocamento, alteração de velocidade e ruído localizado apresentaram o índice mAP superior ao treinamento utilizando apenas os dados originais da base. Entre essas técnicas, a técnica de deslocamento foi a que gerou maior impacto, com o mAP de 0,67. Já a de ruído localizado, chegou até a 0,63 mAP, enquanto a técnica de alteração de velocidade, considerando a margem de até 20%, apresentou resultados superiores que a de velocidade com margem de até 30%.

Durante os experimentos, verificou-se que a aplicação de ruído ao segmento total dos sinais de áudio comprometeu o resultado do treinamento. No entanto, após uma reflexão, constatou-se que esse resultado deu-se por conta de que a base de teste não possui ruídos em suas amostras. Dessa forma, ainda que a aplicação de ruído aos sinais de áudio pudesse ser útil para aumentar o poder de generalização do modelo em situações reais, para fins de teste utilizando os exemplos da base de teste original, essa aplicação não apresentou bons resultados.

O uso das técnicas de aumento de dados individualmente gerou diferentes efeitos durante os treinamentos, sendo verificados os impactos durante os testes. Ainda assim, durante os experimentos, buscou-se combinar as técnicas de aumento de dados de forma a proporcionar maior generalização do modelo e assim garantir um resultado satisfatório.

Além das técnicas de aumento de dados, utilizaram-se também partes de outra base de áudio, a LibriSpeech. Durante os experimentos, optou-se por utilizar apenas um subconjunto dessa base, de forma a verificar o seu impacto no treinamento. A partir da utilização da agregação dessas novas amostras, pode-se obter um resultado satisfatório.

Na Figura 17, é possível verificar o progresso de um dos treinamentos do sistema. Durante esse treino, utilizou-se um *batch size* igual a 16, taxa de aprendizado inicial de 0,1 e termo *momentum* igual a 0,9. Nesse treinamento, utilizaram-se as combinações das técnicas de aumento de dados (ruído local, deslocamento e alteração de velocidade de 20%) aplicadas aos dados de treino da base TIMIT e adicionou-se um subconjunto da base LibriSpeech para aumentar a quantidade de amostras da base e favorecer a generalização

da rede.

Em ambos gráficos apresentados na Figura 17, as marcações ao lado esquerdo do gráfico representam a variação do *Loss* do treinamento, enquanto ao lado direito, verifica-se a escala correspondente à acurácia medida pelo mAP. Da mesma forma, a linha azul representa a variação do *Loss* ao longo dos treinamentos, enquanto a linha laranja corresponde a medida da acurácia.

O gráfico apresentado na Figura 17a representa o treinamento da versão completa da rede, a arquitetura *MobileNet-V3-Large*. O treinamento dessa versão ocorreu de forma pouco ruidosa, apresentando notável progresso e estabilidade. Entre as primeiras 10 mil iterações, houve uma queda expressiva do parâmetro *Loss*, enquanto a acurácia apresentou um crescimento significativo até próximo da iteração 80 mil. Ao final do processo, a acurácia apresentou um valor de 0,72 mAP, enquanto o *Loss* terminou próximo de 0,5.

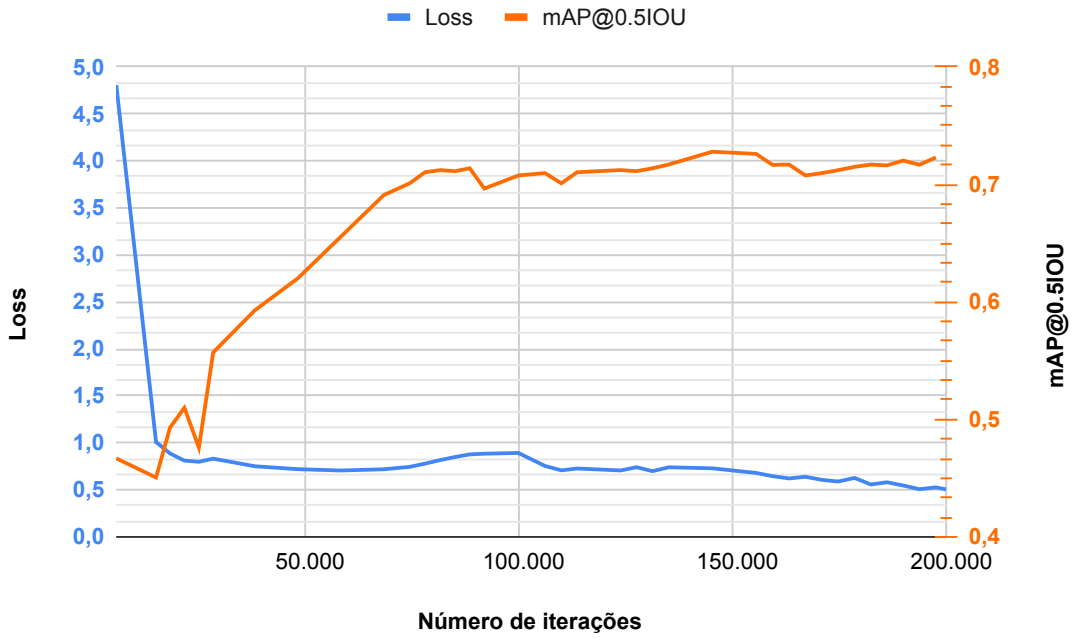
Por outro lado, o gráfico apresentado na Figura 17b representa o treinamento da arquitetura *MobileNet-V3-Small*. Esse treinamento, por sua vez, apresentou um alto nível de ruído, com muitas oscilações ao longo do processo. Essa característica expressou certa dificuldade do sistema em aprender os padrões dos dados. Nesse caso, isso deu-se principalmente por conta dos parâmetros reduzidos da arquitetura, fator que prejudicou a tarefa de aprendizado. Ao final do processo, a acurácia apresentou um valor de 0,63 mAP, enquanto o *Loss* terminou próximo de 1,0.

Na Figura 18a, está ilustrada a matriz de confusão do melhor modelo obtido ao longo dos experimentos. É possível verificar que a maioria das predições estão de acordo com as classes correspondentes, exceto a classe *sil*. Conforme comentado anteriormente, o sistema considerou a maior parte das ocorrências dessa classe como "fundo" da imagem, por isso desconsiderou a sua classificação. Considerando essa possibilidade, uma alternativa para contornar esse problema seria eliminar a classe do silêncio, visto que essa pode ser considerada fundo da imagem e descartá-la não acarretaria prejuízos ao modelo.

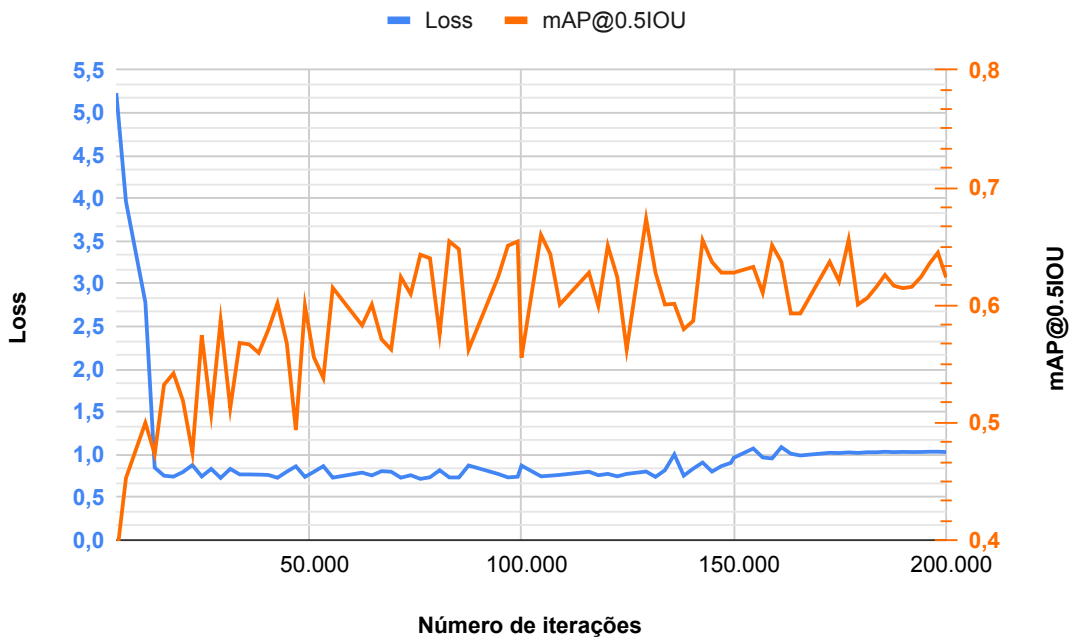
Na Figura 18b, está ilustrada a matriz de confusão do modelo *Small*. O fato deste modelo ser menor e menos robusto que o modelo apresentado anteriormente justifica os resultados menos efetivos. Diferente da matriz de confusão ilustrada na Figura 18a, nota-se que nesse caso existem algumas falhas na diagonal principal. Verifica-se que existem muitas predições imprecisas, mas ainda assim, analisando o panorama geral, o modelo apresentou resultados acima da média.

Figura 17 – Gráfico da evolução do treinamento das redes para 200 mil iterações

a) MobilenetV3-Large



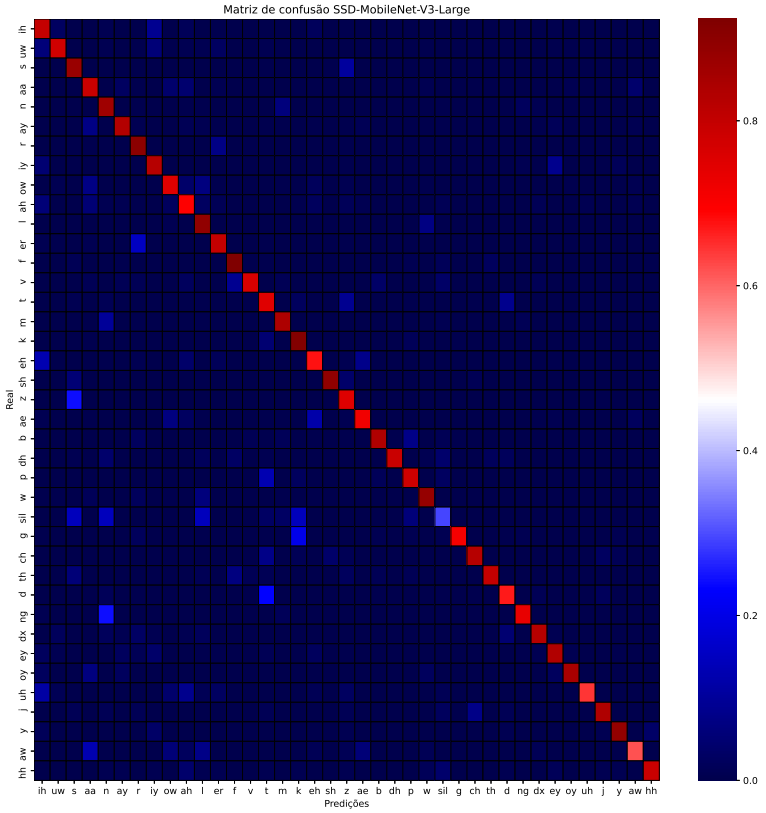
b) MobilenetV3-Small



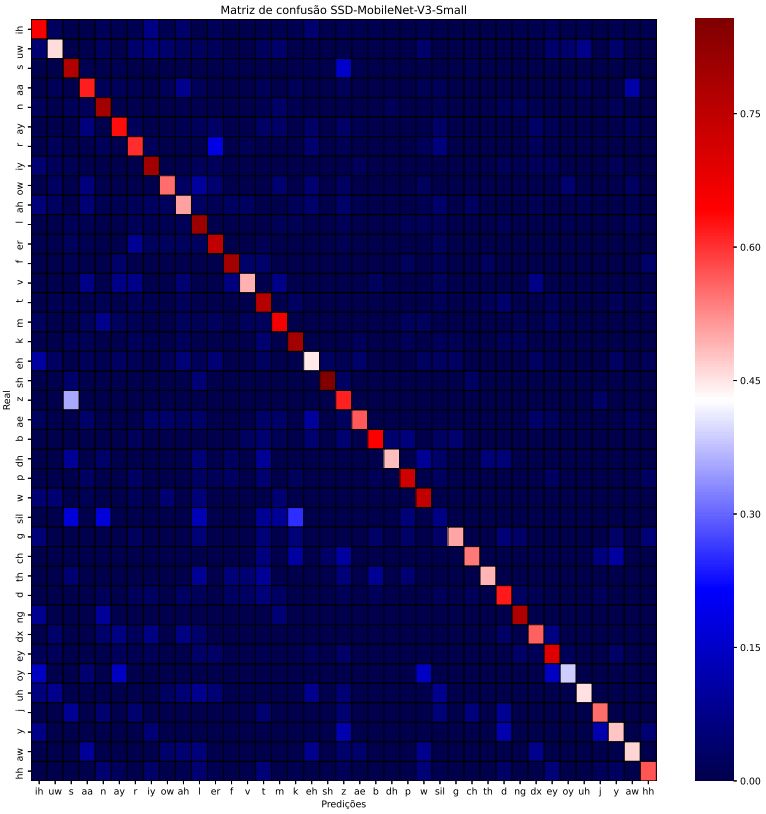
Fonte: Autor

Figura 18 – Gráfico da matriz de confusão das redes treinadas por 200 mil iterações.

a) Matriz de confusão contendo as 39 classes de fonemas do modelo utilizando a arquitetura *Large*.



b) Matriz de confusão contendo as 39 classes de fonemas do modelo utilizando a arquitetura *Small*.

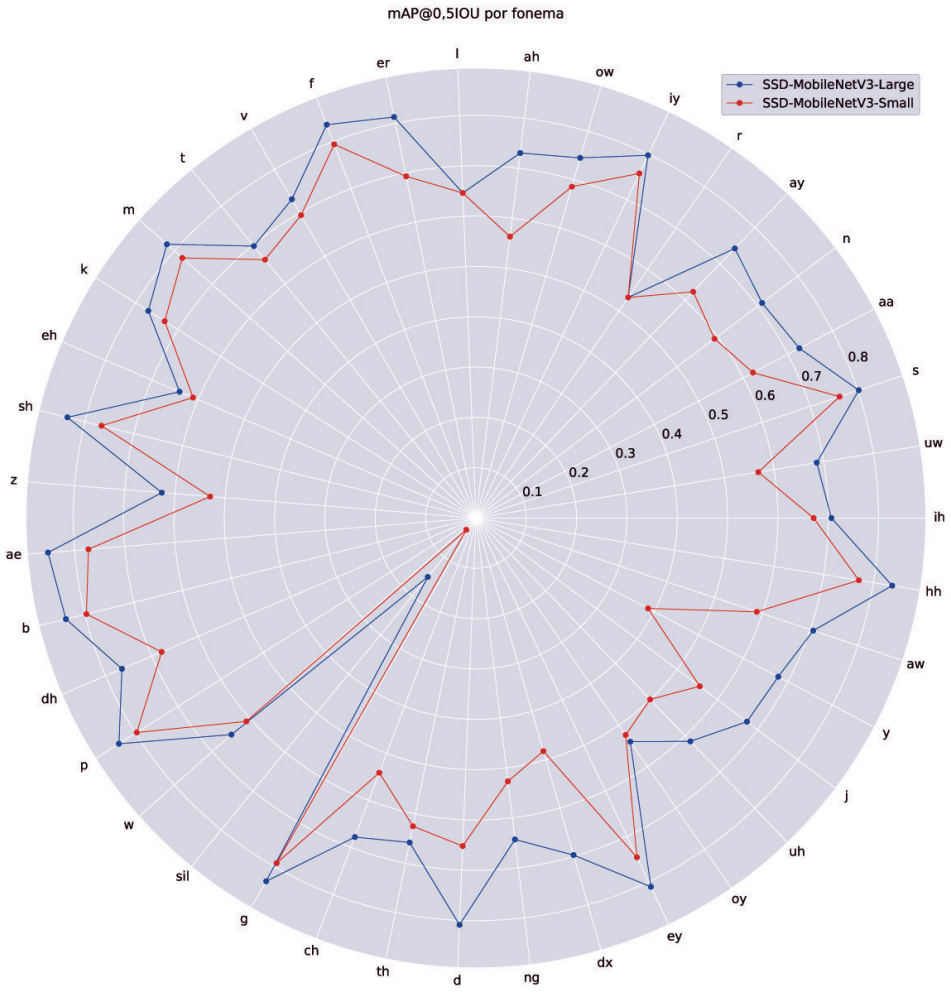


Fonte: Autor.

Na Figura 19, ilustra-se um gráfico circular com objetivo de expor o desempenho em relação ao mAP considerando as classes individualmente. Para isso, considerou-se o índice de interseção sobre união (IoU) igual a 0,5. No que diz respeito a rede mais robusta, pode-se verificar que houve um bom aproveitamento na maioria das classes, exceto para a classe *sil*, que corresponde às pausas e aos silêncios ao longo do sinal de áudio.

Verificou-se que esse problema ocorreu por conta dessa classe frequentemente ser confundida com o "fundo" da imagem. De acordo com a metodologia adotada pelo sistema de detecção de objetos, as detecções consideradas "fundo" da imagem são simplesmente descartadas para evitar que atrapalhe o classificador durante a tarefa de reconhecimento.

Figura 19 – Gráfico da circular comparando o mAP para cada classe de fonemas dos modelos *Large* e *Small*.



Fonte: Autor.

Comparação de resultados

Os resultados obtidos durante os experimentos deste trabalho são apresentados na Tabela 5. Foi feito um comparativo utilizando os dois modelos da mesma arquitetura utilizada, a versão completa (*large*) e a versão reduzida (*small*).

Tabela 5 – Resultados obtidos durante a fase de testes para os modelos da MobileNetV3 *Large* e *Small* utilizando dados de teste da base TIMIT.

Modelo	**mAP (0.5IOU)**	**PER (%)**	**Parâmetros**
SSD-MobileNet-V3-Large	0,729	19,47	5,4M
SSD-MobileNet-V3-Small	0,632	31,02	2,5M

Fonte: Autor.

É notável que a arquitetura de rede completa obteve melhores resultados. Isso ocorre em função de esta possuir mais atributos e consequentemente maior poder de generalização. Em contrapartida, a quantidade de parâmetros utilizadas pela rede reduzida é menor que a metade da quantidade utilizada pela rede maior.

Tabela 6 – Resultados obtidos durante os experimentos no trabalho de Algabri *et al.* (2020)

Modelo	**mAP (0.5IOU)**	**PER (%)**	**Parâmetros**
YOLO v3	0,785	16,34	65,9M
CenterNet	0,766	15,89	62,2M
YOLO V3 Tiny	0,68	25,57	5,5M

Fonte: Autor.

Na Tabela 6, estão expostos os resultados obtidos no trabalho de Algabri *et al.* (2020), que utiliza metodologia semelhante à utilizada neste trabalho. Os resultados são comparáveis, pois foram obtidos utilizando a mesma base de dados durante a fase de testes. Ao analisar as tabelas, observa-se que o produto deste trabalho apresentou resultados levemente inferiores em relação aos de Algabri *et al.* (2020). No entanto, quando se compara o número de parâmetros dos dois sistemas, verifica-se que os sistemas utilizados neste trabalho, embora apresentem acurácia menor, necessitam de uma quantidade de parâmetros expressivamente inferior. Ao comparar a arquitetura Yolo V3 Tiny e a SSD MobileNet V3 Large, verifica-se que a quantidade de parâmetros é equivalente, mas os resultados obtidos pela SSD MobileNet V3 Large são superiores.

5 CONCLUSÃO

O presente trabalho iniciou-se com uma revisão bibliográfica visando a compreender o atual estado de desenvolvimento de sistemas de reconhecimento de voz. Durante essa revisão, identificou-se que os sistemas de reconhecimento são divididos em diversos subgrupos, entre estes, os sistemas de amplo vocabulário e os de vocabulário reduzido. Essa classificação é definida de acordo com a capacidade de reconhecimento do sistema.

Os sistemas que reconhecem diretamente unidades maiores, como palavras ou sentenças, são caracterizados como sistemas de vocabulário reduzido, uma vez que reconhecem apenas um número restrito de comandos. Por outro lado, os sistemas de amplo vocabulário buscam reconhecer unidades menores, como fonemas ou caracteres. A partir desses elementos, compõem-se estruturas maiores, como palavras ou mesmo orações completas. A escolha da unidade de reconhecimento utilizada é fundamental para uma boa eficácia do sistema. Da mesma maneira, a escolha da base de dados é um fator fundamental para o projeto.

Ao longo do trabalho, identificou-se a necessidade de aumentar a quantidade de dados para o treinamento. Embora se tenham obtidos resultados razoáveis durante os experimentos iniciais, ainda não foram suficientes para atingir o resultado almejado. Geralmente, adquirir dados é uma tarefa custosa, e por isso se optaram por aplicar algumas técnicas de aumento de dados (Seção 3.2.2). Adicionalmente, de forma a produzir uma melhor generalização do modelo, buscou-se por alternativas e então decidiu-se incrementar os dados da base TIMIT com a base LibriSpeech.

Para adequar a LibriSpeech à metodologia deste trabalho, utilizaram-se os dados de alinhamento gerados através de um algoritmo de alinhamento automático, diferente da base TIMIT, na qual os dados foram manualmente verificados durante a sua construção. Para isso, selecionou-se um subconjunto da base de dados LibriSpeech e incorporou-o aos dados da base TIMIT original. Esse procedimento proporcionou um ganho na acurácia. Ainda será necessário treinar o modelo utilizando o conjunto de dados completo da base LibriSpeech. Esse procedimento permitirá extrair melhores conclusões sobre o impacto da adição de dados na generalização.

Este trabalho compreende a área de processamento de sinais e identificação de padrões para solucionar o problema em questão. Também aplicaram-se técnicas de visão computacional. Em decorrência disso e com base nos estudos adquiridos na revisão de literatura, optou-se por representar o sinal de forma gráfica, utilizando espectrogramas calculados com base nos sinais de áudio. Definiu-se que se utilizaria redes convolucionais para detecção de padrões e durante a execução do trabalho, verificou-se a possibilidade de aplicar as técnicas de detecção de objetos nas representações dos sinais visando à identificação dos fonemas nos espectrogramas.

Ao longo dos últimos anos, diversos algoritmos de detecção de objetos têm sido propostos e muitos deles apresentam resultados satisfatórios. Apesar disso, além do bom funcionamento do algoritmo no que diz respeito a precisão de suas predições, outros fatores importantes são a latência de resposta das predições e a capacidade de processamento exigida para o funcionamento desses sistemas. Em decorrência dessa preocupação, buscou-se por um modelo que não só apresentasse resultados satisfatórios, mas também não exigisse grande capacidade de processamento. E, consequentemente, pudesse ser utilizado em um dispositivo de baixo poder computacional, de forma independente, no próprio dispositivo, sem precisar empregar um servidor externo para efetuar os processamentos.

Como resultado do trabalho, o modelo desenvolvido apresentou resultados satisfatórios em relação à identificação de fonemas. O modelo oferece uma acurácia eficiente, dado o seu número reduzido de parâmetros. Diante dos resultados obtidos, verifica-se que é possível aplicar os algoritmos de reconhecimento de objetos em espectrogramas. A escolha da arquitetura para desenvolver este trabalho foi fundamental para permitir que o sistema seja utilizado em dispositivos móveis, os quais possuem baixa capacidade de processamento. Esse uso permitirá o uso do modelo de forma gratuita, ilimitada e *standalone*, sem a necessidade de utilizar um serviço externo que, muitas vezes é pago e exige pagamentos pelo número de requisições solicitadas.

Trabalhos Futuros

Dando continuidade a este trabalho, sugere-se implantar o modelo desenvolvido através de uma aplicação móvel e testar a sua utilização em um dispositivo desse tipo. Isso pode ser realizado através de ferramentas disponibilizadas no $framework\ Tensorflow$ (o mesmo utilizado para construir e treinar o modelo).

No presente trabalho, buscou-se construir um classificador de fonemas, mas nada impede de utilizar-se as mesmas estratégias para a tarefa de detecção de palavras, uma vez que a própria base de dados contém anotações referente às palavras. Para isso, bastaria selecionar um determinado conjunto de palavras para treinar o algoritmo de forma semelhante ao executado neste trabalho. Embora a abordagem de detecção de palavras gere um modelo com vocabulário limitado, dependendo da aplicação desejada, esse tipo de detecção pode ser útil.

Além da detecção de palavras, também é possível aplicar a mesma técnica para reconhecimento de elementos sonoros de outras naturezas. Para isso, basta utilizar bases de dados com anotações temporais da ocorrência de efeitos sonoros para possibilitar o treinamento do algoritmo.

Outra modificação possível seria testar novos métodos de representação de sinais, como o MFCC, FBanks ou compactação dos espectrogramas utilizando centro de massa, também pode-se testar a combinação entre estas técnicas. Além disso, também é possível

modificar o *backbone* da rede e utilizar outras arquiteturas em conjunto com o sistema SSD ou mesmo utilizar outro sistema de detecção de objetos como o YOLO ou o Faster R-CNN.

REFERÊNCIAS

ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANé, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIéGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>.

ALGABRI, M.; MATHKOUR, H.; BENCHERIF, M. A.; ALSULAIMAN, M.; MEKHTICHE, M. A. Towards deep object detection techniques for phoneme recognition. *IEEE Access*, IEEE, v. 8, p. 54663–54680, 2020.

BRESOLIN, A. d. A. Reconhecimento de voz atraves de unidades menores do que a palavra, utilizando wavelet packet e svm, em uma nova estrutura hierarquica de decisao. Universidade Federal do Rio Grande do Norte, 2008.

CONIAM, D. Voice recognition software accuracy with second language speakers of english. *System*, Elsevier, v. 27, n. 1, p. 49–64, 1999.

Dai, W.; Dai, C.; Qu, S.; Li, J.; Das, S. Very deep convolutional neural networks for raw waveforms. p. 421–425, March 2017. ISSN 2379-190X.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255.

DING, C.; XIE, L.; ZHU, P. Head motion synthesis from speech using deep neural networks. *Multimedia Tools and Applications*, Springer, v. 74, n. 22, p. 9871–9888, 2015.

ERHAN, D.; SZEGEDY, C.; TOSHEV, A.; ANGUELOV, D. Scalable object detection using deep neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 2147–2154.

EVERINGHAM, M.; ESLAMI, S. A.; GOOL, L. V.; WILLIAMS, C. K.; WINN, J.; ZISSERMAN, A. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, Springer, v. 111, n. 1, p. 98–136, 2015.

EVERINGHAM, M.; GOOL, L. V.; WILLIAMS, C. K.; WINN, J.; ZISSERMAN, A. The pascal visual object classes (voc) challenge. *International journal of computer vision*, Springer, v. 88, n. 2, p. 303–338, 2010.

FAN, R.; LIU, G. Cnn-based audio front end processing on speech recognition. p. 349–354, July 2018.

FURUI, S. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, IEEE, v. 29, n. 2, p. 254–272, 1981.

GAMA, J.; FACELI, K.; LORENA, A.; CARVALHO, A. D. *Inteligência artificial: uma abordagem de aprendizado de máquina*. [S.l.]: Grupo Gen - LTC, 2011. ISBN 9788521618805.

GONZALEZ, R.; WOODS, R. *Processamento Digital De Imagens*. [S.l.]: ADDISON WESLEY BRA, 2009. ISBN 9788576054016.

GORDILLO, C. D. A. *Reconhecimento de Voz Contınua Combinando os Atributos MFCC e PNCC com Métodos de Robustez SS, WD, MAP e FRN*. Tese (Doutorado) — PUC–Rio, 2013.

GRAVES, A.; FERNÁNDEZ, S.; GOMEZ, F.; SCHMIDHUBER, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: ACM. *Proceedings of the 23rd international conference on Machine learning*. [S.l.], 2006. p. 369–376.

GRAVES, A.; MOHAMED, A.-r.; HINTON, G. Speech recognition with deep recurrent neural networks. In: IEEE. *2013 IEEE international conference on acoustics, speech and signal processing*. [S.l.], 2013. p. 6645–6649.

HAYKIN, S. *Redes Neurais: Princípios e Prática*. [S.l.]: Artmed, 2007. ISBN 9788577800865.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

HINTON, G.; DENG, L.; YU, D.; DAHL, G.; MOHAMED, A.-r.; JAITLY, N.; SENIOR, A.; VANHOUCKE, V.; NGUYEN, P.; KINGSBURY, B. *et al.* Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, v. 29, 2012.

HOWARD, A.; SANDLER, M.; CHU, G.; CHEN, L.-C.; CHEN, B.; TAN, M.; WANG, W.; ZHU, Y.; PANG, R.; VASUDEVAN, V. *et al.* Searching for mobilenetv3. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 1314–1324.

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

HUANG, X.; DENG, L. An overview of modern speech recognition. *Handbook of natural language processing*, v. 2, p. 339–366, 2010.

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

JIAO, L.; ZHANG, F.; LIU, F.; YANG, S.; LI, L.; FENG, Z.; QU, R. A survey of deep learning-based object detection. *IEEE Access*, IEEE, v. 7, p. 128837–128868, 2019.

JUANG, B.-H.; LEVINSON, S.; SONDHI, M. Maximum likelihood estimation for multivariate mixture observations of markov chains (corresp.). *IEEE Transactions on Information Theory*, IEEE, v. 32, n. 2, p. 307–309, 1986.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105.

KUZNETSOVA, A.; ROM, H.; ALLDRIN, N.; UIJLINGS, J.; KRASIN, I.; PONT-TUSET, J.; KAMALI, S.; POPOV, S.; MALLOCI, M.; DUERIG, T. *et al.* The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.

LATHI, B. P. *Sinais e Sistemas Lineares-2*. [S.l.]: Bookman, 2006.

LEE, K.-F.; HON, H.-W. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, IEEE, v. 37, n. 11, p. 1641–1648, 1989.

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

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. In: SPRINGER. *European conference on computer vision*. [S.l.], 2014. p. 740–755.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; BERG, A. C. Ssd: Single shot multibox detector. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 21–37.

LUGOSCH, L.; RAVANELLI, M.; IGNOTO, P.; TOMAR, V. S.; BENGIO, Y. Speech model pre-training for end-to-end spoken language understanding. *arXiv preprint arXiv:1904.03670*, 2019.

MCAULIFFE, M.; SOCOLOF, M.; MIHUC, S.; WAGNER, M.; SONDEREGGER, M. Montreal forced aligner: Trainable text-speech alignment using kaldi. In: *Interspeech*. [S.l.: s.n.], 2017. v. 2017, p. 498–502.

Meftah, A.; Alotaibi, Y. A.; Selouani, S. A comparative study of different speech features for arabic phonemes classification. p. 47–52, Nov 2016. ISSN 2473-3539.

MENG, J.; ZHANG, J.; ZHAO, H. Overview of the speech recognition technology. In: IEEE. *2012 fourth international conference on computational and information sciences*. [S.l.], 2012. p. 199–202.

Muckenhirn, H.; Magimai.-Doss, M.; Marcell, S. Towards directly modeling raw speech signal for speaker verification using cnns. p. 4884–4888, April 2018. ISSN 2379-190X.

Palaz, D.; Magimai.-Doss, M.; Collobert, R. Convolutional neural networks-based continuous speech recognition using raw speech signal. p. 4295–4299, April 2015. ISSN 1520-6149.

PANAYOTOV, V.; CHEN, G.; POVEY, D.; KHUDANPUR, S. Librispeech: an asr corpus based on public domain audio books. In: IEEE. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2015. p. 5206–5210.

QUINTANILHA, I. M.; BISCAINHO, L. W. P.; NETTO, S. L. Towards an end-to-end speech recognizer for portuguese using deep neural networks. *XXXV Simpósio Brasileiro de Telecomunicaç oes e Processamento de Sinais, Sao Pedro, Brazil*, 2017.

RABINER, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, Ieee, v. 77, n. 2, p. 257–286, 1989.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 779–788.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 91–99.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M. *et al.* Imagenet large scale visual recognition challenge. *International journal of computer vision*, Springer, v. 115, n. 3, p. 211–252, 2015.

SAK, H.; SENIOR, A.; BEAUFAYS, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: *Fifteenth annual conference of the international speech communication association*. [S.l.: s.n.], 2014.

SAK, H.; VINYALS, O.; HEIGOLD, G.; SENIOR, A.; MCDERMOTT, E.; MONGA, R.; MAO, M. Sequence discriminative distributed training of long short-term memory recurrent neural networks. In: *Fifteenth annual conference of the international speech communication association*. [S.l.: s.n.], 2014.

SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 4510–4520.

Santos, R. M.; Matos, L. N.; Macedo, H. T.; Montalvão, J. Speech recognition in noisy environments with convolutional neural networks. In: *2015 Brazilian Conference on Intelligent Systems (BRACIS)*. [S.l.: s.n.], 2015. p. 175–179.

SILVA, T. *Fonética e fonologia do português: roteiro de estudos e guia de exercícios*. [S.l.]: Contexto, 1999. ISBN 9788572441025.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

ZHANG, X.; ZHOU, X.; LIN, M.; SUN, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 6848–6856.