Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo sobre Redes Neurais no problema da Leitura Labial Automática

Lucas de Almeida Bandeira Macedo

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Bruno Luiggi Macchiavello Espinoza

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo sobre Redes Neurais no problema da Leitura Labial Automática

Lucas de Almeida Bandeira Macedo

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Bruno Luiggi Macchiavello Espinoza (Orientador)
CiC/UnB

Prof. Dr. Tiago Alves
FGA/UnB

Prof. Dr. Pedro Garcia Freitas
CiC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 19 de julho de 2023

Dedicatória

A todos que passaram pela minha vida, e me fizeram ser quem eu sou.

Agradecimentos

Agradeço muito aos meus pais, por terem sempre me incentivado a estudar e a aproveitar as oportunidades que a vida dá.

Agradeço a todos meus amigos que tornaram a jornada da graduação mais leve, em especial ao João Pedro, meu parceiro de monografia.

Agradeço ao meu orientador Bruno por ter me orientado com um espírito tão leve, e ao meu co-orientador Pedro, por ter ajudado tanto, mesmo entrando com o trabalho já em movimento.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Este trabalho tem como objetivo o estudo da aplicação de redes neurais no problema da leitura labial de frases completas. O trabalho é baseado nas redes LipNet e LCANet, arquiteturas baseadas em convoluções tridimensionais e redes recorrentes. O estudo se baseia em uma série de ablações, testando formas de pré-processamento dos vídeos, aplicação de modelos de linguagem no pós-processamento, e as duas arquiteturas citadas. O trabalho mostra que realizar o pré-processamento com um recorte dinâmico traz resultados levemente piores, entre 14% e 16% de piora no WER em alguns casos. A LCANet demonstra resultados superiores à Lipnet, entre 7% e 36%, e com uma convergência muito mais rápida, adquirindo o valor 4 de perda na época 23, 29 épocas antes da LipNet. Por fim, a presença de um modelo de linguagem no pós-processamento traz resultados estritamente melhores em todos os casos de teste.

Palavras-chave: Redes Neurais Artificiais, Leitura Labial Automática, Trabalho de Conclusão de Curso

Abstract

The main goal of this work is to study the use of neural networks on the problem of lipreading complete sentences. This work is based on LipNet and LCANet, two neural network architectures based on tridimensional convolutions and recurrent networks. This study is based on a series of ablations over the pre-processing, architecture choices and the use of a language model while decoding the model's output. THis work shows that using a dynamic crop, opposing to a fixed area crop, yields worse results, between 14% and 16% increase in WER. LCANet outperformed Lipnet, achieving up to 36% better accuracy and a faster convergence, obtaining 4 points of loss in the 23th epoch, 29 epochs faster than LipNet. Finally, the language model brought a strictly better accuracy over every test case.

Keywords: Artificial Neural Networks, Automatic Lipreading, Thesis

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AM Aprendizado de Máquina.

BI-GRU Bidirectional Gated Recurrent Unit.

BLEU Bilingual Evaluation Understudy.

CER Character Error Rate.

CNN Convolutional Neural Network.

CTC Connectionist Temporal Classification.

GPU Graphical Processing Unit.

GRU Gated Recurrent Unit.

IA Inteligência Artificial.

LM Language Model.

LSTM Long Short Term Memory.

ReLU Rectified Linear Unit.

RNN Recurrent Neural Network.

STCNN Spatio Temporal Convolutional Neural Network.

TCC Transformers Convolucionais Compactos.

ViT Vision Transformers.

VRAM Video Random Access Memory.

WER Word Error Rate.

Capítulo 1

Introdução

A leitura labial é a habilidade de inferir falas através do movimento labial de um orador. No dia a dia, usualmente é realizada de maneira inconsciente e auxilia na compreensão dos sons, principalmente em situações que a audição está prejudicada, como em lugares com muita poluição sonora. O Efeito McGurk [1] demonstra a importância da leitura labial na compreensão do áudio. Esse trabalho aponta que, frequentemente, a informação visual interfere na percepção da fala e, frequentemente, toma prioridade em relação a informação auditiva. Por exemplo: se escutamos a sílaba /ba/, mas enxergamos a pronúncia da sílaba /va/, o som percebido será /va/.

Quando lidamos com leitura labial exclusiva, ou seja, sem informações sonoras, o nível de compreensão humana diminui consideravelmente. Muitas dificuldades nascem quando retiramos o áudio do problema. Por exemplo: o movimento labial para as letras /b/ e /p/ é o mesmo, embora o som gerado seja diferente; nesses casos, a desambiguação só pode ser feita a partir do contexto em que a letra foi falada, levando em consideração as outras sílabas e palavras.

Por outro lado, a área de Redes Neurais apenas cresce. Com a relativa recente explosão de popularidade causada, principalmente, pelo uso de placas gráficas no treinamento [2], as redes neurais têm conseguido resolver os mais complexos dos desafios, estabelecendo-se como estado da arte em muitos áreas, como reconhecimento de fala e classificação de vídeos. O uso de redes neurais no problema de leitura labial nada mais é que uma mistura dessas duas áreas. O primeiro trabalho com redes neurais para o problema do reconhecimento de frases por leitura labial foi a LipNet [3]. Esse trabalho se tornou uma referência na área e sua estrutura geral é utilizada por muitos outros trabalhos [4, 5, 6, 7], resumindo-se em convoluções tridimensionais [8] e redes recorrentes, combinação de estruturas normalmente usada em classificação de vídeos, e CTC como função de perda, que foi criada para a área de reconhecimento de voz.

O propósito desse trabalho é realizar um estudo sobre redes neurais aplicadas à leitura

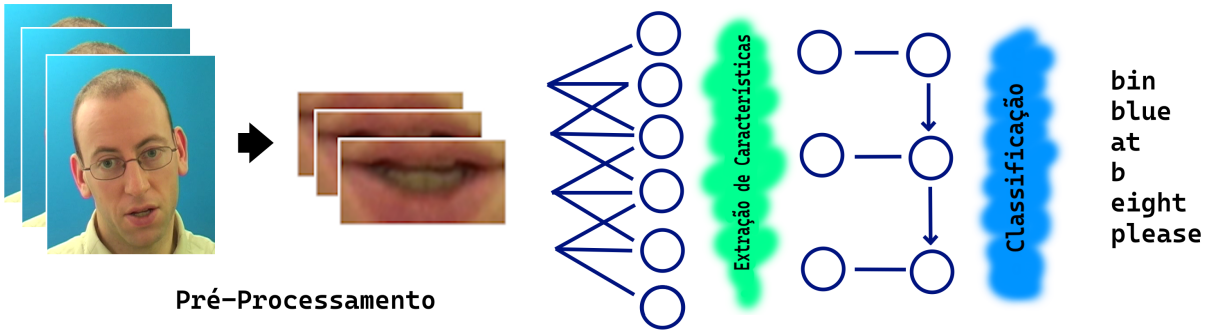

Figura 1.1: Fluxo de trabalho da LipNet.

labial automática e tem como base os mais recentes avanços na área, a partir da LipNet. É feita uma série de ablações para o estudo do problema:

- O pré-processamento é realizado com um recorte de área dinâmica e um recorte de área fixa, para estudar o impacto que diferentes tipos de recorte trazem para o desempenho da rede;
- Treinos com as diferentes redes implementadas para os testes, LipNet [3] e LCANet [4];
- Inserção e remoção do modelo de linguagem na decodificação do resultado.

Este trabalho é composto por 5 capítulos, incluindo a introdução. No Capítulo 2, é estabelecida uma base teórica para o trabalho, seguida de uma curta revisão literária sobre o histórico do tema. No Capítulo 3, o fluxo dos experimentos é detalhado, com explicações para todos os diferentes casos da ablação apresentada. No Capítulo 4, são mostrados os resultados dos experimentos descritos no capítulo anterior, junto a uma análise e discussão sobre cada um deles. Por fim, no Capítulo 5 há um resumo sobre tudo que foi discutido, seguido de uma conclusão e uma menção a trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Introdução a Redes Neurais Artificiais

2.1.1 Inteligência Artificial

Programar é ensinar para a máquina, através de instruções claras, como realizar determinada ação, como em receitas culinárias. Porém, existem coisas que são especialmente difíceis de serem ensinadas dessa maneira. Por exemplo: como julgar se determinada imagem é a imagem de um gato, ou de um cachorro? Para o ser humano é fácil julgar e explicar com palavras: "as orelhas são diferentes" ou "o formato da pupila", etc. Mas como transformar essas explicações em linhas de código? Afinal, uma imagem é apenas uma sequência de números, ou seja, de pixeis. "Se esse pixel for dessa cor", ou mesmo aumentando o nível: "se esse formato existir na foto"... é teoricamente possível montar um programa mas, para atingir uma precisão satisfatória, o programa teria que ser muito complexo para considerar todos os casos, as posições, a iluminação, etc.

A Inteligência Artificial (IA) entra na história com a função de resolver esse tipo de problema: os que conseguimos resolver em função da nossa inteligência humana. A IA nasce quando conseguimos criar programas que podem ser considerados inteligentes, na definição de "inteligência" que utilizamos para o ser humano [9]. Portanto, como fazer programas inteligentes? Existem muitas subáreas no tema de Inteligência Artificial, mas aprofundaremos no Aprendizado de Máquina (AM): máquinas que aprendem e melhoram com a experiência [10]. Essa área de pesquisa nos dá uma solução muito prática para o problema da diferenciação de cachorros e gatos: simplesmente mostre para a máquina vários exemplos de cachorros e vários exemplos de gatos. Com exemplos (experiência) o suficiente, a máquina irá aprender a diferenciar os dois, como um humano aprenderia.

Existem vários jeitos de montar um modelo de Aprendizado de Máquina. No âmbito de classificação de imagens, poderíamos utilizar algumas técnicas de processamento para

encontrar características únicas nas imagens [11, 12, 13] e usá-las para ensinar uma máquina a diferenciar diferentes classes. Alguns algoritmos de AM, como K-Vizinhos Mais Próximos, podem ser muito simples. Por exemplo, esse se baseia apenas em analisar se, de acordo com as características extraídas de todo o conjunto de dados (experiência) em comparação com o dado de entrada, os K mais dados mais parecidos (vizinhos mais próximos) são fotos de cachorros ou fotos de gato, e decidir com base na maioria. Porém, quando lidamos com imagem, o mais comum atualmente é encontrar uma abordagem com Redes Neurais.

2.1.2 Neurônios e Redes Neurais

Como o nome sugere, o neurônio é unidade mínima de qualquer rede neural [14]. Seu funcionamento é muito simples: recebe um conjunto de informações de entrada e calcula uma nova informação como saída:

$$Wx + \beta, \tag{2.1}$$

sendo W um vetor de pesos $1 \times N$, x o vetor de informações de entrada $N \times 1$, N tanto o número de entradas quanto o número de pesos no neurônio, e β um coeficiente linear, representando o viés [15]. O aprendizado de um neurônio está nos pesos e no coeficiente de viés. Cada elemento do vetor de pesos irá multiplicar um elemento do vetor de entrada, dando diferentes importâncias a diferentes elementos do vetor de entrada, e todos os valores resultantes serão somados, resultando em apenas um único número. Esse número será somado com o viés que, como o nome sugere, tem a função de alterar fixamente qualquer saída do neurônio, deslocando o resultado numérico para a esquerda ou para a direita no plano cartesiano. A Figura 2.1 apresenta um diagrama da estrutura de um neurônio artificial.

Em resumo, um neurônio puro é apenas uma função linear. Em um problema de classificação podemos usar um neurônio e especificar "tudo que estiver acima dessa linha é cachorro, tudo que está abaixo dessa linha é gato". Sempre que o modelo errar, ele irá tentar mudar um pouco essa linha de separação para que não erre mais o último exemplo, mas que idealmente também não passe a errar exemplos anteriores. Mas isso apenas funcionará se o problema for linearmente separável. Caso contrário, um neurônio terá dificuldades de fazer uma separação apropriada e nunca chegará em uma solução ótima.

Uma das saídas para este problema são as chamadas "funções de ativação", que nada mais são que funções aplicada à saída do neurônio. Qualquer função que receba um número e devolva outro número pode ser usada como função de ativação, inclusive funções não-lineares. Isso é uma saída para o problema da linearidade, pois assim é possível mapear os resultados lineares de um neurônio para funções mais complexas. Entre algumas funções

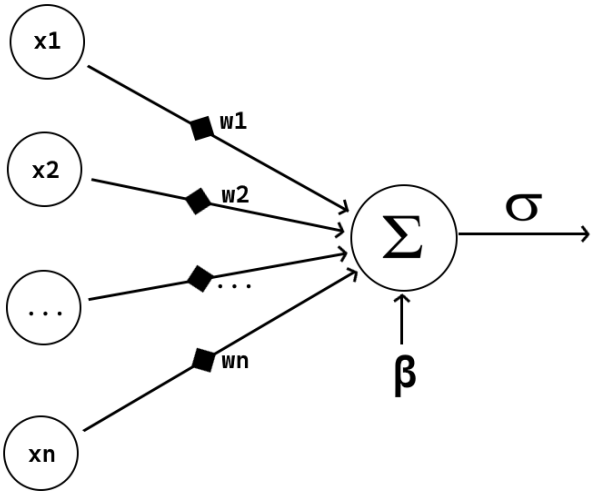

Figura 2.1: Exemplo de neurônio artificial. σ representa uma função de ativação qualquer.

de ativação notáveis, temos a função sigmóide,

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

que efetivamente mapeia os resultados de um neurônio entre 0 e 1, e a Tangente Hiperbólica,

$$\tanh(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}} \tag{2.3}$$

que tem um formato muito parecido com a sigmóide, com intervalo de saída entre -1 e 1 (ver Figura 2.2). Ambas funções são não-lineares e podem ser usadas para auxiliar no problema da linearidade de um neurônio.

Conhecendo um neurônio, a definição de uma rede neural é simples: são vários neurônios interligados, com ao menos uma camada escondida, ou seja, ao menos uma camada de neurônios entre a entrada e a saída. Usar mais neurônios é uma solução simples, mas resolve completamente o problema da não-linearidade. Com apenas uma camada escondida, uma rede neural consegue aproximar qualquer função matemática [16].

A forma mais básica de uma rede neural são as redes densamente conectadas. Essas se baseiam em camadas de neurônios que todo neurônio de uma camada está ligado a todo neurônio da camada seguinte, portanto o nome 'densamente conectada'. A Figura 2.3 mostra uma rede com 5 entradas, 2 camadas escondidas densamente conectadas, e uma camada de saída também densa. Note como todos os neurônios estão conectados entre si entre as camadas.

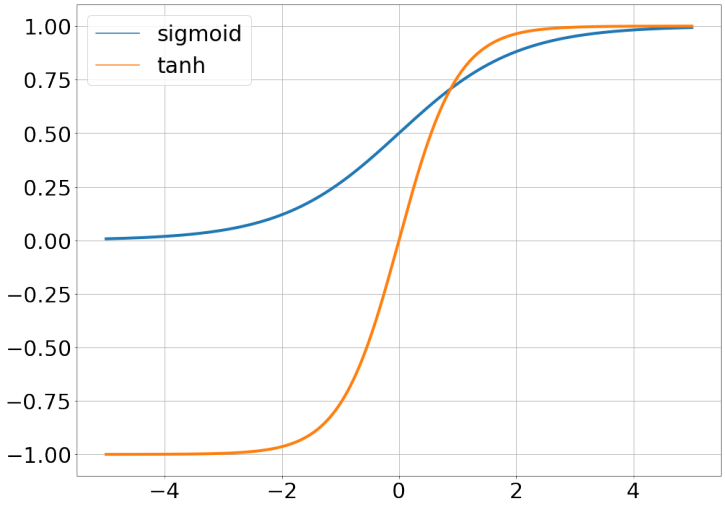

Figura 2.2: Comparação entre a função sigmóide e Tangente Hiperbólica.

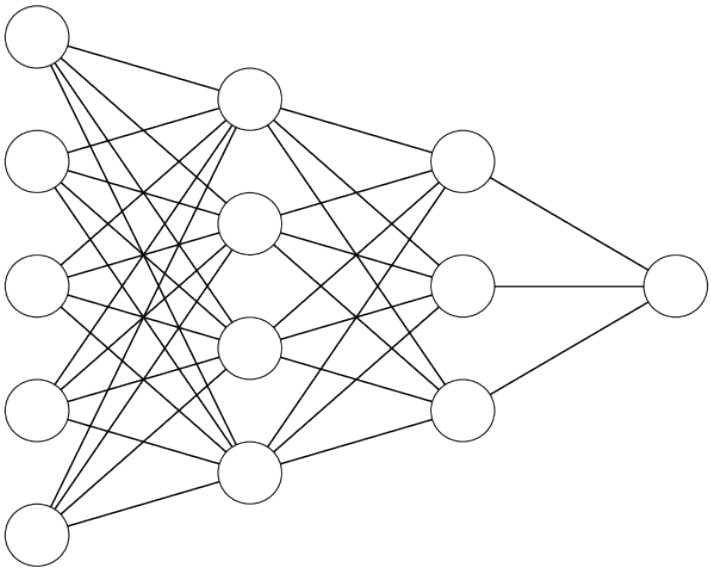

Figura 2.3: Exemplo de organização de neurônios em uma rede densa.

2.1.3 Sobre-ajuste e Sub-ajuste

No geral, quanto mais exemplos são mostrarmos para uma rede, mais ela irá aprender. Quanto mais neurônios colocarmos em uma rede, maior a capacidade de aprendizado. Porém, tudo em excesso faz mal. O fenômeno do sobre-ajuste é extremamente comum e acontece quando, ao invés de "aprender", a rede "decora" uma informação [17]. Pode acontecer em redes com muitos parâmetros, pois essas conseguem "armazenar mais conhecimento", ou quando a rede treina por mais tempo que deveria, memorizando as informações específicas dos dados de treino, que não serão úteis quando a rede encontrar dados novos. Dizemos que esse modelo perdeu a capacidade de generalização.

Por esse e por outros motivos, é normal que o conjunto de dados seja dividido em ao menos dois grupos distintos: treino e teste. É importante que o conjunto de teste nunca seja visto pela rede durante o treino, afinal, de nada vale uma prova que o aluno já viu a resposta. Quando um modelo está sofrendo sobre-ajuste, vemos um bom desempenho no conjunto de treino, e um desempenho inferior no conjunto de teste.

O sub-ajuste é justamente o contrário, quando a rede não consegue aprender o suficiente, e acontece pelos motivos inversos. O fenômeno pode acontecer quando a rede tem poucos parâmetros (pesos), e tem capacidade de aprendizado menor do que necessita para o problema. Também pode acontecer quando o modelo é simplesmente pouco treinado, quando não teve experiência suficiente para detectar os padrões do problema.

2.1.4 Gradientes e o Aprendizado

Mas como o aprendizado é realizado, matematicamente? Durante o treino, acontecem duas etapas. A primeira é a chamada 'alimentação', que se baseia em simplesmente colocar um dado na rede e adquirir o resultado, a previsão do modelo. Como ainda estamos treinando, precisamos também medir o quanto a rede errou nessa inferência, e para isso usamos uma Função de Perda. Essa função tem o objetivo de calcular a diferença entre o previsto e o real [18], e é usada como parâmetro para saber quanto a rede errou.

A segunda etapa do processo de aprendizado é a 'retro-alimentação'. Usamos essa informação de perda para fazer o caminho inverso na rede, estimando o quanto cada neurônio da rede errou, calculando os gradientes para cada neurônio. Um gradiente em respeito a perda é calculada pela derivada no ponto atual da rede. Como estamos calculando os gradientes de trás para frente, os últimos neurônios têm uma derivada simples:

$$f'(x), \tag{2.4}$$

mas todo neurônio anterior à última camada representa uma função composta por si mesmo e todos os próximos neurônios:

$$[f(g(x))]', \tag{2.5}$$

portanto, esse cálculo deve ser feito com a regra da cadeia:

$$[f(g(x))]' = f'(g(x))g'(x). \tag{2.6}$$

Cada gradiente indica o quanto e como mudanças nesses neurônios influenciariam na perda do modelo [17]. Já que queremos minimizar a perda, nosso objetivo também é decrescer esses gradientes.

Porém, nasce um problema quando estamos montando redes muito profundas. Pela forma que os gradientes são calculados, o cálculo em neurônios muito profundos se torna uma cadeia muito longa de derivadas. Com multiplicações atrás de multiplicações, é comum que gradientes divirjam (gradiente explosivo), com muitas multiplicações em sequência com números maiores em magnitude que 1, ou convirjam para 0 (gradiente desaparecido), multiplicando com números entre -1 e 1. Como os gradientes guiam as alterações no peso da rede, se eles explodirem ou desaparecerem os pesos da rede farão o mesmo, impedindo a rede de aprender qualquer padrão.

2.2 Funções de Perda

Funções de Perda têm um grande papel no treinamento de uma rede neural, uma vez que são elas que guiam o modelo para a convergência (Seção 2.1.4). Podemos abordar o problema de leitura labial de muitas formas diferentes. A forma mais simples é, com base em um curto vídeo, montamos uma rede neural que irá prever a palavra dita dentre todas as palavras possíveis, ou seja, a saída da rede será uma distribuição de probabilidades de N_p posições, sendo N_p a quantidade de palavras possíveis. Para medir o erro durante o treinamento, uma função comum é a entropia cruzada [19], que irá calcular a entropia entre a distribuição de probabilidade que a rede estimou, e a distribuição real (que na maioria das vezes indica 100% de chance em uma única palavra). Para o problema modelado dessa forma, a função de perda guiará o modelo para a convergência, mas a função escolhida também está limitando o modelo. Não podemos estender o problema para previsão de frases completas, por exemplo, pois teríamos que ter um estado para cada combinação de palavras possível.

Previsão de frases completas de tamanho arbitrário e formação de novas palavras com fonemas conhecidos são problemas que a entropia cruzada não consegue resolver por conta

própria. A ausência de uma função de perda apropriada foi um problema que a área de reconhecimento de fala sofreu por muito tempo, até a criação da Classificação Temporal Conexionista (em inglês *Connectionist Temporal Classification* - CTC) [20]. Essa função de perda revoluciona por não limitar a rede a ter um tamanho fixo na sua saída. Em vez de analisar diretamente a saída e comparar se os símbolos previstos (letras, fonemas ou palavras) são os mesmos que os símbolos verdadeiros, a CTC demanda a previsão de um símbolo para cada instante de tempo, de forma que a probabilidade de determinado símbolo acontecer em determinada posição é a soma de todas as probabilidades desse símbolo acontecer, considerando que vários símbolos iguais em sequência é considerado apenas uma ocorrência (mais detalhes na Seção 3.3).

2.3 Estruturas Neurais

Resta um dilema: redes profundas aumentam o potencial de aprendizado de uma rede, mas também podem causar a explosão e o desaparecimento dos gradientes. Podemos substituir redes profundas (muitas camadas) por redes largas (muitos neurônios por camada), mas isso trará outros problemas. Por exemplo, se temos uma imagem de 100×100 pixels conectada a uma camada densa de 128 neurônios, temos mais de um milhão de pesos apenas nessa pequena rede! Claro, é possível treinar assim, mas existem estruturas neurais (jeitos engenhosos de organizar neurônios) que não só reduzem o número de pesos necessários para o treinamento e contornam os problemas dos gradientes, mas também melhoram significativamente os resultados.

2.3.1 Redes Convolucionais

Muitas áreas de pesquisa com IA podem ser divididas entre antes e depois da popularização das redes convolucionais, pelo seu grande impacto. Em resumo, as Redes Neurais Convolucionais (em inglês *Convolutional Neural Network* - CNN) [21] são estruturas neurais voltadas para resumir informações locais em vetores ou matrizes, e funcionam exatamente como as convoluções normais, mas os filtros são pesos treináveis. Foram inicialmente introduzidas na área de classificação de imagens, e se diferem das camadas densas por, em vez de de colocar cada pixel da imagem em cada neurônio, um neurônio irá processar a imagem pedaço por pedaço, e cada resultado gerará um novo elemento de matriz, gerando uma nova imagem no final do processo. Aprofundando: dada uma matriz de pesos F (filtro) de dimensões $M \times N$, e uma imagem I de dimensões $H \times W$, sendo

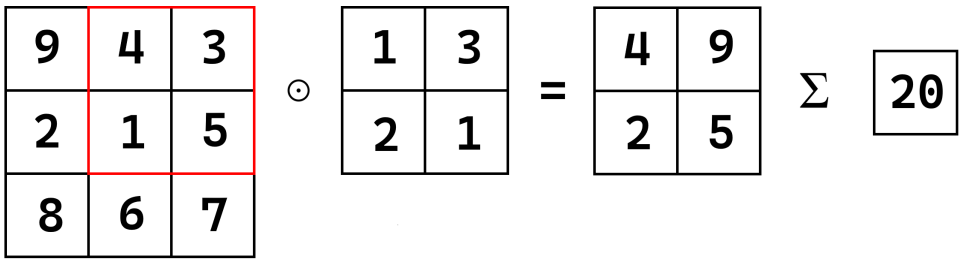

Figura 2.4: Exemplo de um passo convolucional.

$H \geq M$ e $W \geq N$, um passo convolucional é modelado por:

$$\sum_{i=1}^{M} \sum_{j=1}^{N} F_{i,j} \cdot I_{i+\Delta h, j+\Delta w}, \tag{2.7}$$

sendo Δh e Δw o deslocamento vertical e horizontal da extração de características na imagem. A estrutura convolucional fará esse passo de convolução em cada posição possível na imagem de entrada.

A Figura 2.4 ilustra um exemplo visual para melhor compreensão. Ela mostra apenas um passo da estrutura convolucional, com uma imagem 3×3, um filtro 2×2, e um deslocamento horizontal $\Delta w = 1$. A primeira matriz ilustrada é a imagem de entrada I, uma matriz com números quaisquer, e a segunda é o filtro F, uma matriz com números adquiridos através do processo de treinamento. A terceira matriz é o resultado da multiplicação elemento-a-elemento, onde cada elemento do filtro multiplicou o elemento correspondente na imagem, na área demarcada. A última matriz é o somatório dos elementos da terceira matriz, que é o resultado final desse passo. Um passo convolucional é equivalente ao processamento de um neurônio, sendo o filtro seus pesos, e a área demarcada o dado de entrada.

A Figura 2.5 mostra o resultado completo da convolução anterior. O passo convolucional é aplicado 4 vezes, uma para cada possível posição com o tamanho do filtro. Se não adicionarmos *padding*, o resultado final de uma convolução completa é sempre uma nova matriz, menor. Usando convoluções tradicionais, as novas matrizes podem ser novas imagens com diferentes características, como bordas realçadas, imagem borrada, etc. Porém, é difícil afirmar que uma convolução a partir de uma rede neural será uma imagem discernível, uma vez que os pesos treinados podem estar extraindo informações que não fazem sentido nenhum para o ser humano, mas matematicamente ajudará o modelo a fazer uma previsão.

Embora as convoluções tenham sido propostas para imagens, elas não se limitam mais a esse escopo. Existem variantes dessa estrutura mas que seguem exatamente a mesma

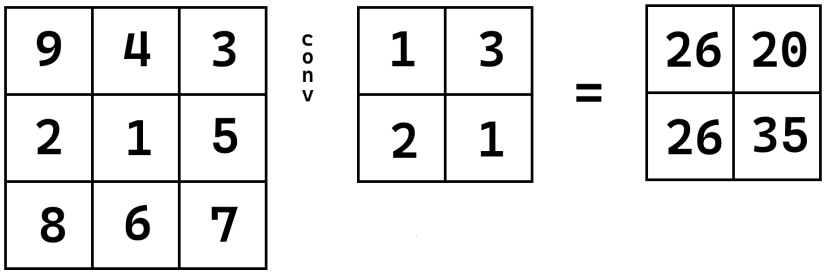

Figura 2.5: Resultado completo de uma convolução.

lógica, como as convoluções unidimensionais, que conseguem resumir a informação em vetores de uma dimensão só. Esse trabalho usa convoluções tridimensionais [8], também chamadas de Convoluções no Espaço-Tempo (em inglês *Spatio Temporal Convolutional Neural Network* - STCNN) uma vez que estamos processando vídeos, e não imagens, que seguem as seguintes dimensões: altura × largura × tempo.

2.3.2 Redes Recorrentes

Redes Neurais Recorrentes (em inglês *Recurrent Neural Network* - RNN) [22] são estruturas neurais voltadas para a análise temporal de determinado dado. Ou seja, se temos uma informação que varia ao longo do tempo (um vetor, em que cada elemento é um instante de tempo diferente), podemos usar essa estrutura. RNN's se baseiam em analisar iterativamente uma informação temporal, sempre analisando o instante de tempo atual junto com as informações adquiridas pela análise anterior:

$$o_i, h_i = RNN(x_i, h_{i-1}), \tag{2.8}$$

sendo o_i a saída da rede na iteração i, h_i o estado escondido resultante da iteração i, e x_i a entrada no instante de tempo i. Para cada instante de tempo, produzimos a saída da rede o, que pode ser usada pelas camadas subsequentes, e um estado escondido h, que será usado nos cálculos da próxima iteração. Essa é a estrutura geral de qualquer RNN, e variantes dessa estrutura mudarão apenas como são calculadas as saídas e os estados escondidos.

GRU

O problema de usar redes recorrentes é que elas têm uma inclinação natural a sofrer do problema do gradiente explosivo ou desaparecido [23], afinal, cada recorrência é uma função a mais na cadeia das derivadas. Porém, algumas estruturas recorrentes conseguem

contornar esse problema, e esse é o caso da Gated Recurrent Unit (GRU) [24]:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + C_z c_t), \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + C_r c_t), \\ \tilde{h}_t &= \tanh(W_h x_t + U_h(h_{t-1} \odot r_t) + C_h c_t), \\ h_t &= (1 - z_t) \odot h_{t-1} + (z_t \odot \tilde{h}_t). \end{aligned} \tag{2.9}$$

As entradas dessa estrutura são o dado temporal x_i, o estado escondido da iteração anterior h_{t-1}, e o contexto do instante de tempo atual c_t. O operador $\odot$ representa a multiplicação elemento-a-elemento. A introdução do vetor de contexto c é opcional, e é comumente descartado. W, U e C são pesos. A saída da rede é estado escondido h_t.

A GRU funciona em duas partes. Primeiramente, é calculado o portão de *reset* r. Esse portão servirá para medir a quantidade de informação do estado anterior que será introduzida no cálculo do $\tilde{h}$, dando oportunidade para o modelo remover informações anteriores que já não são mais úteis neste instante de tempo e destacar as informações atuais presentes na entrada x. Em seguida, é calculado o portão de atualização z, que será usado como parâmetro para fazer uma média ponderada entre a informação do estado escondido anterior e a nova informação $\tilde{h}$. Portanto, a saída no instante de tempo t nada mais é que uma média ponderada entre h_{t-1} e $\tilde{h}_t$, sendo que o portão de atualização representa os pesos dessa média.

2.3.3 Mecanismo de Atenção

O Mecanismo de Atenção [25] é uma estrutura neural que tem como objetivo, com base nas entradas da estrutura, destacar informações importantes em relação às outras. Foi originalmente projetada para tradução de máquina, um problema do tipo sequência-para-sequência, ou seja, problemas em que tanto a entrada quanto a saída são sequências temporais. Na forma que foi proposta originalmente, a atenção segue essa estrutura:

$$\begin{aligned} e_{ij} &= v^T \tanh(W_e s_{i-1} + U_e h_j), \\ \alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \\ c_i &= \sum_{j=1}^{T_x} \alpha_{ij} h_j, \end{aligned} \tag{2.10}$$

sendo s a sequencia temporal gerada como saída do mecanismo, h a sequencia temporal de entrada, e W, U e v são matrizes de pesos. No trabalho original, esse mecanismo deve ser usado juntamente com uma RNN para gerar a sequencia s. e é a chamada pontuação de

atenção, e é calculada entre a saída anterior s_{i-1} e cada entrada do mecanismo h. O vetor de pontuações é aplicado em uma *softmax*, que tem o objetivo de criar uma distribuição de probabilidade com base na magnitude dos elementos do vetor original. Por fim, essa distribuição de probabilidade é multiplicada pela entrada original h, com cada elemento da distribuição multiplicando um dos instantes de tempo presentes na entrada, e finalizando com um somatório de todos os elementos desse vetor resultante.

Na prática, cada pontuação e representará quanto determinado vetor deve ser considerado, quanta importância ele tem. Vetores com uma pontuação muito baixa serão multiplicados por números muito próximos de 0, reduzindo sua relevância no resto da rede, enquanto os com pontuação alta serão multiplicados por um número próximo de 1, se mantendo inalterado. O resultado disso é que, ao invés de dependermos inteiramente do estado escondido de uma RNN para obter a contextualização temporal, a rede consegue inferir quais instantes de tempo são relevantes durante a análise de algum instante de tempo, dando destaque a eles e suprimindo os pouco importantes. Outros tipos de atenção substituem apenas o cálculo da pontuação, preservando o resto da estrutura.

Transformer

A Atenção é tão poderosa que foi criada uma estrutura neural, baseada exclusivamente em camadas de atenção e camadas densas, que consegue substituir completamente as RNN's em algumas tarefas. Essa estrutura é a chamada Transformer [26], e foi originalmente proposta na área de Processamento de Linguagem Natural, no problema de tradução de máquina. Usando uma Atenção de Várias Cabeças (apenas várias unidades de atenção paralelas, recebendo a mesma entrada), a estrutura consegue extrair várias relações temporais diferentes para cada unidade, enriquecendo a análise temporal com análises de longa distância, já que essa estrutura não depende de estados escondidos para transmitir informações dos instantes anteriores. Por acabar com a recorrência, o Transformer acaba também com o problema do gradiente explosivo ou desaparecido que a recorrência trazia. Outra vantagem das Transformers é a redução no tempo de treinamento, uma vez que uma unidade recorrente realizará uma iteração para cada instante de tempo, enquanto a Transformer realiza todas simultaneamente.

2.4 Leitura Labial

A leitura labial automática é um tópico de pesquisa que consegue misturar trabalhos de diferentes áreas em suas arquiteturas. Os modelos propostos dessa área podem ser divididas em duas partes: a extração de características, onde um vídeo é resumido em números ricos em informações; e a classificação, quando a rede toma as decisões finais baseado nas

características adquiridas anteriormente. O classificador, por ser essencialmente um gerador de texto, pega emprestado muitos conceitos de áreas de Processamento de Linguagem Natural, como tradução automática ou reconhecimento de voz.

2.4.1 LipNet

A LipNet [3] foi a primeira rede neural a prever frases inteiras, oposto a apenas letras ou números. Esse trabalho estabeleceu novos padrões para a área e se tornou um ponto de referência para desempenho muito comum. A rede proposta é baseada em STCNN's como extratores de características, GRU's como classificadores e CTC como função de perda.

Extrator de Características

A primeira parte da rede terá a função de extrair informações do vídeo e, para isso, usará 3 camadas de STCNN, pois assim conseguirá condensar as informações tanto espacialmente quanto temporalmente. Entre cada uma dessas camadas serão aplicadas a normalização de batch [27], ReLU [28], dropout [29] e max-pooling [30], nessa ordem.

A **normalização de batch** é uma camada treinável de redes neurais que aprende a fazer a padronização (média 0 e desvio padrão 1) dos dados. Ou seja, a função dessa camada é aprender qual a média e o desvio padrão dos dados que chegam na camada e aplicar

$$\tilde{x} = y^{(k)} \frac{x^{(k)} - E[x]^{(k)}}{STD[x]^{(k)}} + \beta^{(k)}, \tag{2.11}$$

sendo k um índice associado a cada dimensão do dado, x o dado neste instante da rede, $E[x]$ e $STD[x]$ a esperança e o desvio padrão do conjunto de dados nesse instante da rede, e y e β sendo os parâmetros treináveis da rede. Ou seja, cada elemento será estandardizado usando como referência o conjunto de dados completo e, em seguida, sendo utilizado em uma função linear $y * \tilde{x} + \beta$. Essa normalização se tornou essencial no treino de redes convolucionais, permitindo treinos mais rápidos e resultados melhores.

A **Unidade de Reficação Linear (em inglês *Rectified Linear Unit* - ReLU)** é uma função de ativação simples:

$$ReLU(x) = MAX(0, x); \tag{2.12}$$

e será utilizada após cada normalização de batch. O uso de ReLU's como função de ativação, no lugar de sigmóides ou Tangentes Hiperbólica, é conhecido por trazer resultados melhores em problemas de imagens, mitigando o problema do desaparecimento do gradi-

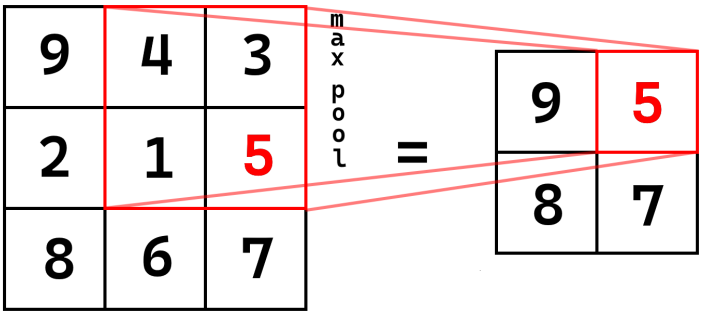

Figura 2.6: Imagem ilustrativa de max-pooling.

ente, e trazendo a presença de verdadeiros zeros na rede, ou seja, resultados de neurônios que não trarão impacto nenhum ao longo da predição [31].

A técnica conhecida como **dropout** tem como objetivo evitar o sobre-ajuste. Ela se baseia em, durante o treino, aleatoriamente ignorar alguns neurônios completamente (ou seja, colocar suas saídas como 0). Desse modo, para garantir melhor desempenho da rede durante o treino, a rede será obrigada a generalizar o significado dos neurônios, dando a eles funções menos específicas para que a rede não seja dependente de poucos neurônios para fazer muito trabalho.

Por fim, o **max-pooling** se baseia na ideia de "reduzir" a quantidade de informação em determinado espaço. Diferentemente das redes convolucionais, dentro de determinada vizinhança, o max-pooling simplesmente escolhe o maior elemento disponível, como ilustrado na Figura 2.6. Isso permite que a rede continue levando apenas as informações da maior magnitude, e se provou ser um método de redução de informação muito superior a outros métodos, como o sub-sampling, em CNN's [32].

Classificador

Com a extração de características visuais resolvida, agora é necessário resolver a extração de características temporais, ou seja, a relação temporal entre os quadros do vídeo, que compõe o movimento labial. Essa função será do decodificador da rede. São usadas duas **BI-GRU**'s para extrair toda a informação temporal a cerca do vídeo. Uma BI-GRU é apenas uma combinação de duas GRU's normais, porém, uma recebe o vetor na ordem temporal correta, enquanto a outra receberá o vetor na ordem inversa. O propósito desse mecanismo é aumentar a riqueza do contexto que as RNN's conseguem detectar, uma vez que cada uma conseguirá inferir informações diferentes dependendo de qual ordem receber as informações.

Por fim, com as informações temporais bem definidas, precisamos apenas produzir a saída final da rede. Para isso, aplicaremos uma camada densamente conectada com uma

saída do tamanho do vocabulário (neste caso será 28, sendo 26 letras do alfabeto, espaço em branco e um caractere vazio, que será usado pela função de perda) e com uma função de ativação softmax. Essa função será a responsável por produzir o vetor de distribuição de probabilidade a partir da saída da camada densa:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad para\ i = 1, 2, \ldots, K. \tag{2.13}$$

2.4.2 LCANet

A LCANet [4] é uma rede que construída com base na LipNet, e muitas coisas serão iguais entre as duas. É possível, na verdade, encontrar a LipNet inteiramente dentro da LCANet, porém com algumas coisas a mais. A LCANet vem com o objetivo de resolver o problema da dependência de longo prazo que redes recorrentes inerentemente apresentam, trazendo melhorias na parte de classificação da rede.

Primeiramente, após as camadas convolucionais, a LCANet implementa o que é chamado de camada **Highway** [33]. O nome sugere uma analogia com "estrada", pois o propósito dessa camada é que as informações relevantes passem livremente, enquanto as informações pouco relevantes serão retidas. Ela funciona de forma parecida com uma LSTM [34] ou GRU, só que sem a recorrência:

$$\begin{aligned} t &= \sigma(W_T x + b_T), \\ h &= \sigma(W_H x + b_H), \\ g &= t \odot h + (1 - t) \odot x, \end{aligned} \tag{2.14}$$

sendo x a entrada, g a saída, σ a função sigmóide, W os pesos e b os coeficientes de viés (bias). Essa camada funciona fazendo a média ponderada entre duas informações: a entrada da rede e o h, que nada mais é que a informação original transformada após passar por uma camada densa. A ponderação é decidida pela transform gate, definida por t, e pela carry gate, definida por $(1 - t)$. Essencialmente, a carry gate decidirá quanta da informação original continuará fluindo, e a transform gate decidirá quanto da informação transformada será introduzida na saída. O propósito dessa camada é aumentar a qualidade das informações que chegam no classificador.

A última e principal modificação proposta pela LCANet é a chamada **Atenção em Cascata**, introduzida logo após às duas BI-GRU's. Esse mecanismo é o mesmo apresentado na Seção 2.3.3, e se baseia em uma rede recorrente que, antes de cada recorrência, é feita uma atenção entre cada instante de tempo do dado de entrada e o estado anterior

da rede. Formulando matematicamente:

$$\begin{aligned} c_t &= a(s_{t-1}, h), \\ s_t &= GRU(Ey_{t-1}, s_{t-1}, c_t), \\ y_t &= SOFTMAX(D(s_t)), \end{aligned} \tag{2.15}$$

sendo t o instante de tempo da recorrência, a o mecanismo de atenção, h a saída das BI-GRU's, E uma matriz Embedding a nível de caracteres, D uma camada densamente conectada, e y_t a saída da rede no instante de tempo t. Uma matriz Embedding é apenas uma matriz de pesos treináveis que servem como uma tabela de pesquisa: cada chave da tabela é um símbolo, e os elementos são vetores com informações (características, pesos treinados) a respeito desse símbolo. Por receber toda a informação de entrada em cada recorrência, essa estrutura mitiga o problema da dependência de longo prazo presente nas RNN's, embora não tão bem quanto as Transformers.

2.4.3 LCSNet

A LCSNet [6] também é uma rede construída com base na LipNet. Essa rede nasce com a ideia oposta da LCANet: melhorias na extração de características, uma vez que apenas STCNN não entregam informações muito ricas ao classificador. As alterações propostas se resumem em filtrar os resultados das convoluções para que apenas o que é importante seja utilizado no classificador, excluindo informações pouco relevantes e redundantes. Dois módulos novos foram adicionados na extração de características da rede: a Atenção por Canal [35] e a Fusão Seletiva de Características.

As Atenção por Canal foram introduzidas após cada STCNN, e multiplicam cada um dos canais presentes na saída das convoluções por um número entre 0 e 1, dando diferentes importância a cada canal, e suprimindo os considerados pouco importantes. Essa estrutura funciona com dois processamentos em paralelo: em um caminho, a entrada sofre um max-pooling global, e no outro, a entrada sofre um average-pooling global. Ambos caminhos terminam em redes profundas densas. A diferença entre um pooling regular e um pooling global é apenas seu escopo: o regular resume a informação em pequenas parcelas, em pequenas áreas; o global resume a informação de um canal inteiro em apenas um número. Ambas redes densas compartilham os mesmos pesos. A saída de ambas redes é somada e escalada entre 0 e 1 com a função sigmóide, representando a

importância de cada informação. Essa estrutura é formulada por:

$$\begin{aligned} O_1 &= RD(APG(x)), \\ O_2 &= RD(MPG(x)), \\ saida &= \sigma(O_1 + O_2), \end{aligned} \tag{2.16}$$

em que APG e MPG são os average e max pooling globais, RD é a rede densa, e x é o valor de entrada. A saída da rede é um número, de 0 a 1, para cada canal da entrada, representando a importância que foi associada a cada um. Essa saída multiplica esses canais de volta, reduzindo a magnitude dos canais com baixa importância, e mantendo os canais de alta importância inalterados.

A Fusão Seletiva de Características é uma estrutura proposta para substituir a camada Highway usada pela LCANet. Seu propósito é o mesmo, filtrar as características adquiridas pelas convoluções, enriquecendo os dados para o classificador. Porém, seu funcionamento é diferente: a informação é dividida em dois vetores distintos, cada um produzido por uma camada densa com pesos distintos. Cada um desses vetores terá características diferentes, e disputam um espaço na saída final da estrutura, que será balanceado pelo resultado de uma rede densa profunda.

2.4.4 LipFormer

A LipFormer [7] é o atual estado da arte. Ela também é uma arquitetura baseada na LipNet, mas usa ideias das LCANet e LCSNet, além de suas ideias próprias. A principal contribuição dessa rede é propor uma solução multi-modal, ou seja, que recebe vários tipos de dados para resolver o problema. Além do vídeo, a rede recebe como entrada as características faciais do orador (ver Seção 3.1.1).

A rede é inicialmente dividida em duas vertentes, uma para a extração de características dos vídeos, e outra para as características faciais. A vertente do vídeo é baseada em uma sequência de três STCNN, como na LipNet, seguida por uma Atenção por Canal, como na LCSNet, e finalizando com duas BI-GRU's. A vertente das características faciais é baseada apenas em duas BI-GRU's. Em seguida há a fusão das características: ambas informações são combinadas em apenas uma através de uma atenção cruzada. Por fim, essa informação é processada por um classificador em cascata, semelhante ao da LCANet.

Capítulo 3

Metodologia

Treinar uma rede neural não é uma tarefa simples a ponto de apenas conectar os dados à rede, é uma tarefa de várias etapas:

- Pré-processamento dos dados, para otimizar o aprendizado da rede;
- Construção ou escolha da arquitetura a ser utilizada;
- Pós-processamento dos resultados.

3.1 GRID Dataset

O GRID [36] foi o conjunto de dados utilizado para o treino e avaliação dos modelos treinados. Cada elemento desse conjunto é um vídeo em ambiente controlado que um voluntário é gravado falando uma frase em inglês gerada aleatoriamente, a partir de palavras pré-determinadas. Cada frase tem sempre 6 palavras, na ordem: comando, cor, preposição, letra, número, advérbio. Seguindo essa regra, é possível montar 64 mil frases com as palavras disponíveis, mas o conjunto de dados tem apenas 34 mil vídeos gravados. Os vídeos foram gravados com a participação de 34 voluntários, cada um gravando mil vídeos. Cada vídeo foi gravado com o rosto centralizado e um pano de fundo monocromático, de forma que apenas o rosto do participante esteja em destaque, e nada mais atrapalhe. Este conjunto de dados é muito utilizado no problema de leitura labial, de forma que se tornou uma referência comum para avaliações de modelos.

3.1.1 Preparo dos dados

Raros são os problemas de aprendizado de máquina que não precisam de nenhum tipo de tratamento de dados. Dependendo da área, apenas entregar à rede dados crus pode ser uma má ideia, uma vez que esses podem vir com muito ruído. Isso se aplica aqui.

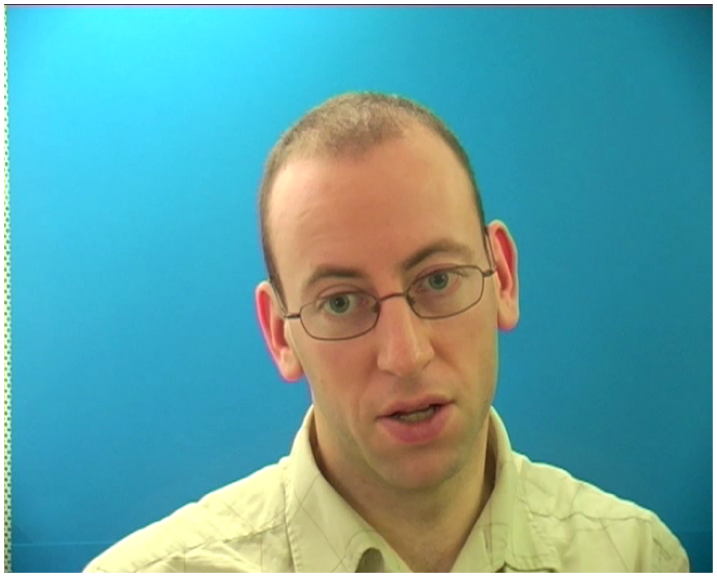

Figura 3.1: Exemplos de quadros de vídeo do conjunto de dados GRID.

Como os dados disponíveis são vídeos com o rosto inteiro sendo filmado, apenas inserir esses vídeos em um modelo de aprendizagem de máquina não trará bons resultados, pois existem muitas informações disponíveis em um vídeo desses que pouco importam para a leitura labial, como o formato do nariz, a cor do cabelo, a posição dos olhos, o plano de fundo, etc. A informação mais valiosa é, com certeza, o movimento dos lábios e da língua. Ou seja, para atingir melhores resultados, precisamos remover todas essas informações que não são importantes para não "confundir" a rede, de forma que ela se concentre apenas no que é importante. A melhor forma de atingir esse objetivo é realizar um processamento nos vídeos, excluindo toda a informação, ficando apenas com o recorte da boca.

Esse processamento foi realizado em três etapas: reconhecimento do rosto, o alinhamento e o recorte. O reconhecimento foi realizado utilizando um modelo pré-treinado de marcos faciais (*facial landmarks*) [37, 38]. Esse modelo é comumente utilizado na área de biometria facial, e foi treinado com o objetivo de encontrar 68 pontos de interesse no rosto, sendo 6 pontos de referência para cada olho e 20 pontos para a boca, como demonstrado na Figura 3.2. Usamos esse modelo para, para cada quadro de vídeo do conjunto de dados, extrair os pontos relativos aos centros de cada um dos olhos e da boca, calculando os centroides dos pontos:

$$C = \left(\frac{\sum_{i=1}^{N} X_i}{N}, \frac{\sum_{i=1}^{N} Y_i}{N} \right), \tag{3.1}$$

sendo X e Y as coordenadas de um dos olhos ou da boca, e N o número de pontos associados à parte do rosto que estamos considerando.

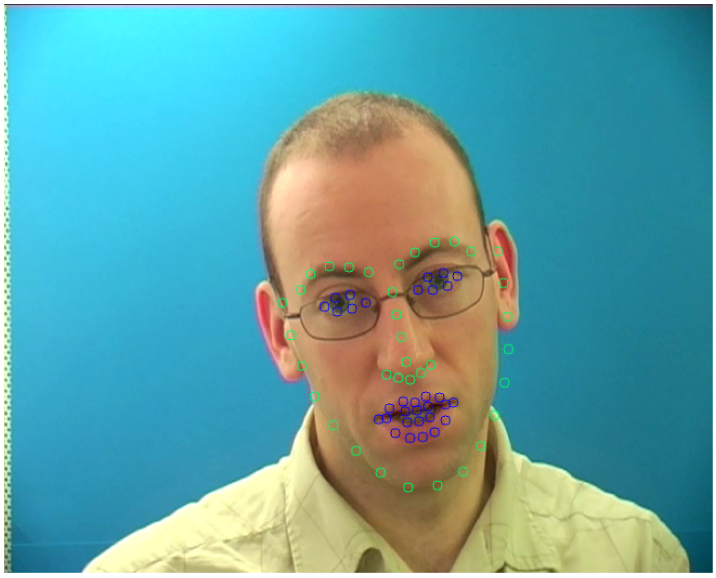

Figura 3.2: Exemplo de marcos faciais. Os pontos em azul são relativos à boca e aos olhos, enquanto os verde são relativos ao resto do rosto.

Em seguida, faremos o alinhamento: alinhar o rosto de forma que ele não fique inclinado nem para a esquerda, e nem para a direita, fique o quão simétrico for possível. Para isso, usaremos os centroides que calculamos: vamos rotacionar a imagem de forma que a distância e o ângulo entre esses três pontos, os dois olhos e a boca, permaneça sempre o mesmo, mas que os olhos estejam horizontalmente alinhados. Portanto, dado os centroides dos olhos esquerdo e direito, CE e CD, e o centroide da boca, CB, encontramos a distância entre os olhos DO,

$$DO = sqrt\left((CE_x - CD_x)^2 + (CE_y - CD_y)^2\right), \tag{3.2}$$

as coordenadas do ponto médio dos olhos MO,

$$MO = \left(\frac{CE_x + CD_x}{2}, \frac{CD_y + CD_y}{2}\right), \tag{3.3}$$

e a distância do ponto médio dos olhos e o centroide da boca DOB,

$$DOB = sqrt\left((DO_x - CB_x)^2 + (DO_y - CB_y)^2)\right). \tag{3.4}$$

Em seguida, calculamos os novos pontos para cada um dos centroides, que são utilizados como referência para a rotação da imagem. O centroide da boca servirá como âncora do alinhamento, ou seja, a imagem rotacionará em volta de um eixo imaginário nesse ponto. Portanto, o objetivo se torna encontrar apenas as novas coordenadas dos olhos. Primeiro,

Figura 3.3: Exemplo da rotação do vídeo..

calculamos o novo ponto médio entre os olhos NMO, que será horizontalmente alinhado com o centro da boca e manterá a distância original em relação à boca,

$$NMO = (CB_x, CB_y - DOB)\,. \tag{3.5}$$

Por fim, sabendo que os olhos são igualmente espaçados em relação ao seu ponto médio, podemos calcular o novo centroide de cada olho, NCE, NCD, de forma que eles estejam verticalmente alinhados entre si e também em relação ao ponto médio:

$$\begin{aligned} NCE &= \left(NMO_x - \frac{DO}{2}, NMO_y\right), \\ NCD &= \left(NMO_x + \frac{DO}{2}, NMO_y\right). \end{aligned} \tag{3.6}$$

Com as novas coordenadas disponíveis, basta realizar a rotação da imagem. A Figura 3.3 demonstra a preservação de distâncias e ângulos do método proposto.

Por último, precisamos extrair a imagem da boca. Para isso, faremos dois tipos de recorte para fins de comparação: um recorte fixo, recortando uma área de tamanho pré-definido ao redor do centro da boca; e um recorte dinâmico, recortando a menor área que inclua toda a boca. O primeiro caso é o mais simples: dado o centroide da boca, selecionamos uma área de tamanho fixo e pré-determinado, de forma que o centro dessa área seja também o centroide da boca. Dado o centroide da boca CB, o canto superior esquerdo SE e o canto inferior direito ID do recorte são calculados da seguinte forma:

$$\begin{aligned} SE &= (CB_x - L/2, CB_y - A/2)\,, \\ ID &= (CB_x + L/2, CB_y + A/2)\,; \end{aligned} \tag{3.7}$$

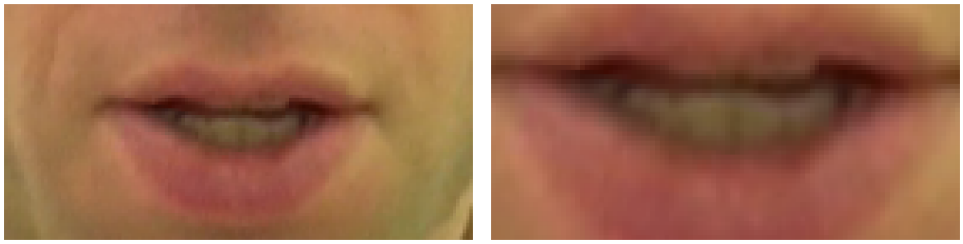

Figura 3.4: A figura à esquerda mostra o resultado do recorte fixo, enquanto a figura da direita mostra o resultado do recorte dinâmico..

sendo L e A a largura e a largura desejadas do recorte da boca. Essa área deve abranger uma parte maior que a área da boca, uma vez que é o mesmo recorte para todos os quadros do vídeo, e o os lábios não podem sair do enquadramento durante o ato de abrir e fechar a boca.

O segundo tipo de recorte, o recorte dinâmico, tem como objetivo excluir completamente qualquer informação se não a área da boca: através dos pontos de referência do rosto, escolhemos uma área retangular que abranja todos os pontos da boca, e que cada lado do retângulo toque sempre em um ponto. Dado que B_x e B_y são vetores com todas as coordenadas x e y dos pontos de referência da boca:

$$\begin{aligned} SE &= (MIN(B_x), MIN(B_y)), \\ ID &= (MAX(B_x), MAX(B_y)). \end{aligned} \tag{3.8}$$

Ou seja, a área de recorte é a menor área possível em que a boca continue bem enquadrada. Todos os vídeos devem ser redimensionados para tamanhos padrões, uma vez que a rede exige uma entrada de tamanho fixo.

3.2 Arquiteturas utilizadas

A escolha de uma arquitetura de rede neural é de extrema importância, pois ela deve atacar todas as complexidades da informação apresentada. Em um vídeo existem 3 dimensões de informação: altura e largura, compondo a imagem ou quadro de um vídeo; e o tempo, sendo várias imagens sendo apresentadas em sequência compondo um movimento. Qualquer arquitetura que for escolhida para o problema de leitura labial deve levar em consideração essas 3 dimensões durante o aprendizado, ou seja, deve aprender tanto a relação entre diferentes áreas de uma imagem, quanto o movimento apresentado por quadros adjacentes. Portanto, os experimentos foram conduzidos com as redes LipNet (Seção 2.4.1) e LCANet (Seção 2.4.2).

3.2.1 Outros experimentos

Entretanto, outros experimentos também foram conduzidos, na tentativa de expandir o repertório de arquiteturas para comparação. Essa seção é dedicada a arquiteturas que foram estudadas e implementadas, mas apresentaram resultados abaixo do esperado e foram descontinuadas da pesquisa.

Transformer de Visão

Os transformers foram originalmente introduzidos na área de Processamento de Linguagem Natural com o propósito de tradução de máquina (Seção 2.3.3), mas também é possível adaptar a estrutura para atacar outros tipos de problema. Esse é o caso dos Vision Transformers (ViT) [39]. A estrutura de um transformer foi montada esperando símbolos como entrada, para que a relação entre esses símbolos seja detectada pelas estruturas de atenção. Uma vez que símbolos de um texto são informações de caráter temporal, a atenção traçará a relação temporal entre as informações, mas isso não é uma limitação da estrutura. O Transformer apenas traçará a relação entre os diferentes elementos de um vetor. Portanto, a proposta do ViT é separar a imagem em vários pequenos pedaços, como se fossem símbolos textuais, e traçar a relação espacial entre eles.

Porém, o próprio trabalho original da ViT aponta que essa estrutura requer uma quantidade muito grande de dados para superar as estruturas convolucionais, na ordem dos milhões, algo que o conjunto de dados GRID não atinge. Os Transformers Compactos [40] são uma variante desse trabalho e, como o nome sugere, são redes com poucos parâmetros, que conseguem explorar o potencial dos Transformers sem necessitar de tantos dados. Das arquiteturas propostas por esse trabalho, foi utilizada o Transformers Convolucionais Compactos (TCC). Essa arquitetura se diferem em: ao invés de dividir as imagens em pequenos símbolos, a proposta é criar os símbolos diretamente com redes convolucionais, tornando os símbolos mais ricos com informações locais, mas ainda explorando o potencial dos Transformers de estabelecer relações de longa distância.

No trabalho de leitura labial, a proposta de implementar os TCC's foi enriquecer a informação entregue aos decodificadores das redes LipNet ou LCANet, melhorando a extração de características espacial para melhorar a classificação final, mas foi descartada por falta de tempo.

LipFormer

A LipFormer [7] é uma arquitetura inspirada nas Transformers, embora use poucas das suas ideias. O objetivo do trabalho é resolver o problema do sobre-ajuste inerente aos modelos de leitura labial, uma vez que eles costumam a criar muito viés em favor aos

oradores que foram vistos durante o treino, aprendendo informações como a cor da pele, ou o formato dos lábios. Portanto, sua principal contribuição é propor uma fusão multimodal para o problema de leitura labial, ou seja, misturar vários tipos de dados para uma única predição, para reduzir a dependência visual do modelo. Assim, além do vídeo, a arquitetura também faz uso dos marcos faciais extraídos do vídeo, conforme explicado na Seção 3.1.1.

A Lipformer também requer adições no pré-processamento, pois agora há um segundo dado entrando na rede. Em vez de inserir as coordenadas dos pontos de referência do rosto na rede, o trabalho propõe calcular o ângulo entre cada ponto de referência do lábio (20 pontos) e cada ponto do contorno do rosto (17 pontos), gerando 340 números por quadro de vídeo. Por fim, cada vetor de 340 posições será subtraído pelo vetor adjacente, de forma que o dado de entrada seja a diferença dos ângulos entre os quadros do vídeo.

Essa arquitetura também foi descartada por falta de tempo, uma vez que a publicação do artigo ocorreu no final do prazo de entrega do trabalho.

3.3 Pós-processamento

A saída de qualquer uma das redes apresentadas é uma matriz de dimensão 75×28, ou seja, 75 predições (uma para cada quadro do vídeo), com cada predição sendo uma distribuição de probabilidade de 28 possíveis estados, equivalente às 26 letras do alfabeto, o espaço em branco e a marcação vazia (␣) do CTC. Porém, pelo modo que o CTC funciona, transformar essa saída em palavras não é apenas escolher, para cada quadro, a letra ou marcação mais provável que aconteça, pois se a rede considerar que o orador está falando a letra $'A'$ por mais de um quadro durante o vídeo, então serão previstas várias letras $'A'$ consecutivamente, mas no resultado final deve ser apresentada apenas uma. Assim, durante a decodificação, letras repetidas consecutivas são consideradas apenas uma. Quando duas letras devem ser repetidas, como na palavra $'GREEN'$, a rede irá prever uma marcação vazia entre as duas letras $'E'$, de forma que não serão duas letras iguais consecutivas. Ou seja, a chance da letra $'A'$ ser prevista, se considerarmos apenas dois quadros do vídeo, é a soma das probabilidades $p(\text{␣}'A') + p('A'\text{␣}) + p('A''A')$. Se considerarmos todos os quadros do vídeo, e todas as possíveis combinações de letras possíveis, encontramos um problema de alta complexidade computacional.

Esse problema pode ser modelado como uma busca em grafo, uma vez que dada uma matriz de previsões, buscamos o caminho de maior probabilidade escolhendo uma letra por instante de tempo. Modelando o problema desta maneira, o caminho de maior probabilidade será a sentença a ser decodificada. Usamos o algoritmo conhecido como *Beam Search* que se baseia em, a partir do nó mais promissor, expandir a busca em até

N outros nós. Essa heurística serve para acelerar a execução do algoritmo e, ao mesmo tempo, limitar os recursos computacionais que serão usados para decodificar a frase.

Para melhorar a precisão dessa busca em grafo, um modelo de linguagem pentagrama será utilizado como segunda heurística. Os modelos de linguagem n-grama são modelos baseados nas cadeias de Markov, ou seja, a probabilidade de uma letra acontecer em determinada posição depende inteiramente das letras previstas anteriormente. Colocando isso em um exemplo matemático, se temos um modelo trigrama, a probabilidade da letra $'D'$ acontecer na palavra $'RED'$ é expressa pela probabilidade condicional:

$$P(D|R, E). \tag{3.9}$$

Com essa lógica, conseguimos escalar o cálculo de probabilidades para palavras inteiras. Por exemplo, a probabilidade da palavra $'PLACE'$ acontecer nesse mesmo modelo trigrama será

$$P(P|\langle s\rangle, \langle s\rangle) * P(L|\langle s\rangle, P) * P(A|P, L) * P(C|L, A) * P(E|A, C), \tag{3.10}$$

sendo $\langle s\rangle$ a indicação de começo de texto. Esse modelo será muito relevante para pequenas correções onde a rede pode ter errado, como trocado uma letra no meio de uma palavra, o que é muito comum em sistemas de reconhecimento de fala, uma vez que muitos sons e, nesse caso, muitos movimentos labiais se tornam ambíguos na hora de traduzir para texto, sendo impossível fazer uma decisão com boa precisão sem levar em consideração o contexto.

Capítulo 4

Resultados

4.1 Os dados

Os dados serão processados seguindo o protocolo apresentado na Seção 3.1.1. Cada elemento do conjunto de dados será um vídeo do recorte da boca, terá uma resolução de 100×50 pixels e 75 quadros. A extração de marcos faciais foi feita com a ferramenta Dlib [37]. Todos os vídeos correspondentes a um orador em específico (1000 vídeos) estão indisponíveis. Dos que estão disponíveis, nem todos os estão em boas condições: alguns estão corrompidos, outros o rosto do orador ficou oculto, resultando em problemas no pré-processamento. 330 vídeos se apresentam nessas condições, sobrando 32670 vídeos usáveis. O conjunto de dados também possui a legenda correspondente de cada vídeo, indicando quais palavras foram faladas e em quais momentos do vídeo. Esse dado será usado como "gabarito" para o modelo.

Fazemos, também, dois tipos de divisões para avaliação dos modelos: oradores sobrepostos e oradores ocultos [3]. Na primeira divisão, apenas dividimos aleatoriamente todo o conjunto de dados entre treino e teste, de forma que 80% dos vídeos sejam usados para treino e 20% dos vídeos sejam usados para teste. Esse cenário leva em consideração que oradores que participaram do treinamento são novamente usuários do modelo. A segunda divisão, dos oradores ocultos, se baseia em esconder alguns dos oradores do treino, de forma que os oradores que sejam apresentados no conjunto de testes sejam completamente novos à rede.

4.2 Aumento de dados

Aumento de dados é a técnica de produzir novos dados de maneira artificial e utilizá-los no treinamento. Essa técnica é de extrema importância quando o conjunto de dados é pequeno ou, no nosso caso, queremos evitar o sobreajuste (ver Seção 2.1.3). Três técnicas

são utilizadas nesse projeto: o espelhamento horizontal dos vídeos, a introdução de "travamentos" nos vídeos e o treinamento com palavras avulsas. Os espelhamentos servem para aumentar a capacidade do modelo de generalizar o problema, uma vez que não faz diferença para a leitura labial se uma boca está ou não refletida. Cada vídeo tem 50% de chance de ser espelhado durante o treino. O segundo aumento de dados, os travamentos são apenas repetições e deleções aleatórias de quadros em um vídeo, e serve para aumentar a "resiliência" do modelo a qualquer falha de gravação, e cada quadro dos vídeos têm 5% de chance de ser duplicado ou deletado.

Por fim, o treinamento com palavras soltas é apenas a introdução de novos vídeos com apenas uma palavra. Esses novos vídeos são gerados utilizando a legenda do vídeo, que indica qual palavra está sendo falada e suas correspondentes marcações temporais. Como o banco de dados é organizado de forma a sempre seguir uma ordem de classe de palavras ao construir uma frase (ver Seção 3.1), é possível que o modelo entenda que determinada palavra apenas pode acontecer no começo do vídeo, ou sempre no final, de forma a aprender a organização do próprio conjunto de dados, ao invés de aprender sobre leitura labial de fato. O treinamento com palavras soltas mitiga o problema, obrigando o modelo a entender palavras fora de qualquer contexto.

4.3 Métodos Avaliativos

Avaliar o quão bem um modelo conseguiu fazer a leitura labial é o mesmo que comparar o texto do que foi dito com o que foi previsto. Para isso, três métricas são utilizadas: a Taxa de Erro de Caracteres (em inglês *Character Error Rate* - CER), a Taxa de Erro de Palavras (em inglês *Word Error Rate* - WER) e o Avaliação em Estudo Bilíngue (em inglês *Bilingual Evaluation Understudy* - BLEU) [41]. CER e WER são apenas variações da mesma lógica, e são calculadas usando a Distancia de Levenshtein:

$$\frac{(S + D + I)}{N} \tag{4.1}$$

, onde S é o número de substituições de símbolos necessárias para tornar o texto gerado igual ao texto verdadeiro, D é o número de deleções, I é o número de inserções, e N é a quantidade de símbolos na frase. A métrica BLEU não só conta as diferenças, mas também leva em conta a extensão das similaridades, a ordem que acontecem, e a precisão combinada de diferentes tamanhos de n-gramas. É uma métrica primariamente utilizada em tradução, uma vez que consegue levar em consideração um número arbitrário de referências para uma única hipótese. Consideraremos apenas os unigramas para o cálculo do BLEU.

Tabela 4.1: Comparação de resultados das redes e os diferentes pré-processamentos. Asterisco (*) indica os resultados reportados pelo trabalho original.

	Sobrepostos			**Oculto**		
	CER	**WER**	**BLEU**	**CER**	**WER**	**BLEU**
LipNet*	1.9%	4.8%	-	6.4%	11.4%	-
LipNet (fixo)	1.32%	3.07%	96.92	15.63%	27.55%	72.44
LipNet (dinâmico)	1.50%	3.55%	96.44	12.88%	21.94%	78.06
LCANet*	1.3%	2.9%	97.4%	-	-	-
LCANet (fixo)	1.22%	2.70%	97.29%	9.85%	17.99%	82.00%
LCANet (dinâmico)	1.43%	3.22%	96.77%	11.76%	19.45%	80.54%
LCSNet*	1.0%	2.3%	-	-	-	-
Lipformer*	-	1.45%	-	-	9.64%	-

4.4 Detalhes de implementação

Ambas as arquiteturas foram implementadas utilizando Tensorflow[1]. A implementação do Torch do *Beam Search* foi utilizada, com o tamanho do raio (*beam width*) em 200. O modelo de linguagem utilizado foi implementado pela biblioteca NLTK, e treinado com as sentenças do conjunto de dados de treino. Todas as camadas dropout foram utilizadas com a taxa em 50%. Foi utilizada uma *Learning Rate* de 0.0001 e otimizador ADAM [42]. Durante o treinamento, foi utilizado o processador Intel Core i7-8700 com 32GB de RAM com duas GPU's Nvidia RTX 2080 de 12GB de VRAM. Todas as redes foram treinadas com tamanho de *batch* de 64 vídeos. Nessas circunstâncias cada rede demorou entre 1.5 e 3 dias para o treinamento completo.

4.5 Comparação de Resultados

Todos os experimentos foram conduzidos nas mesmas circunstâncias, utilizando as técnicas de processamento de dados descritas anteriormente. Nenhum dado externo ao conjunto de dados GRID [36] foi utilizado. Não foi realizado nenhum tipo de transferência de aprendizado, ou seja, todos os modelos foram treinados do zero. Os experimentos foram feitos com a intenção de reproduzir os resultados atingidos pelos artigos de referência.

Como mostrado na Tabela 4.1, comparando os resultados reportados pelo artigo original com os resultados da nossa implementação, a LipNet atingiu resultados diferentes nas duas modalidades propostas, melhores resultados nos oradores sobrepostos e piores resultados nos oradores ocultos. Isso demonstra que os nossos modelos treinados tiveram uma capacidade muito maior de aprender os detalhes de como algumas pessoas em especí-

[1]O repositório com as implementações está disponível em `https://github.com/ABMHub/TCC/tree/entrega_tcc`

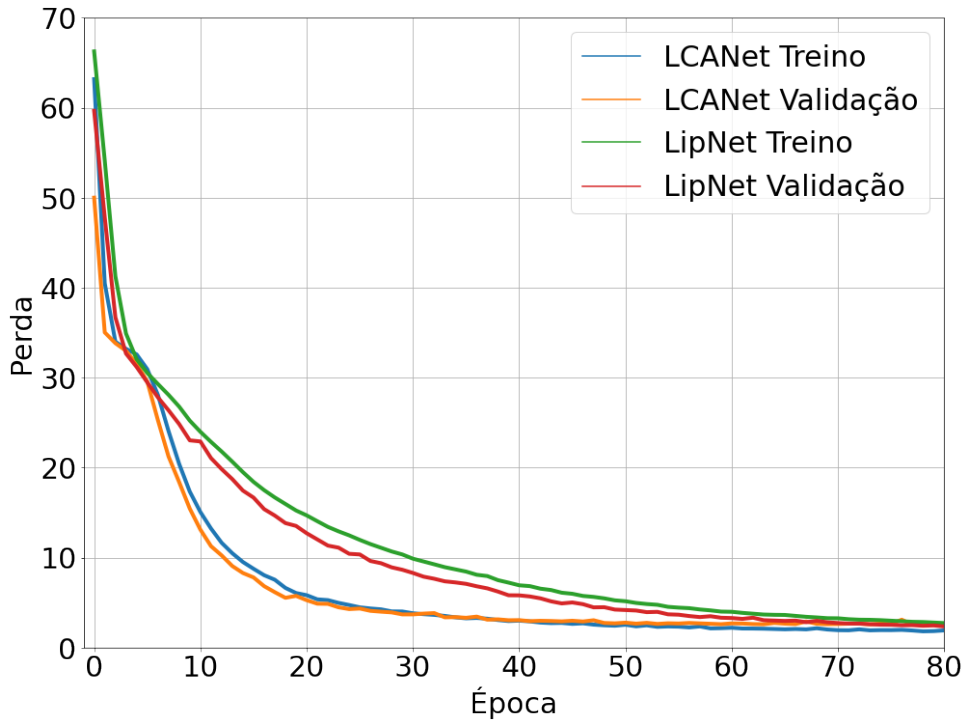

Figura 4.1: Gráfico comparativo entre arquiteturas da perda (loss) ao longo das épocas.

fico falam, como seus vícios musculares, mas tiveram um desempenho pior que o esperado na questão da generalização do aprendizado, uma vez que não conseguiram prever muito bem para pessoas que nunca viram no treino. Especula-se que a diferença está no pré-processamento ou no aumento de dados. A LCANet, por outro lado, atingiu resultados muito próximos dos reportados.

A LCANet obteve resultados melhores que a LipNet. Isso é esperado, pois a LCANet vem com a proposta de melhorar a arquitetura original da LipNet. A margem de melhoria, entretanto, é menor que as reportadas originalmente, uma vez que os resultados adquiridos com a arquitetura LipNet se diferenciaram tanto na modalidade sobrepostos. Observando a Figura 4.1 e comparando as duas arquiteturas, vemos que a LCANet tem uma velocidade também de convergência muito maior. Para efeitos de comparação, nesse mesmo gráfico, a loss de validação da LCANet atinge o valor 4 na época 23, enquanto a LipNet atinge o valor 4 na época 52. Essa diferença acontece em função do módulo de atenção presente na LCANet, que consegue manter uma correlação temporal de longo alcance, diferente o uso exclusivo de RNN's, que sofrem do problema de "esquecimento" do contexto em sequencias temporais muito grandes.

A Tabela 4.2 mostra que a presença de um modelo de linguagem para auxiliar na decodificação do resultado traz resultados estritamente melhores. Isso ocorre pois muitos fonemas são ambíguos para a rede, por exemplo, o movimento para a emitir as consoantes

Tabela 4.2: Comparação de resultados do impacto do modelo de linguagem na decodificação do resultado. Utilizando recorte fixo.

	Sobrepostos			**Oculto**		
	CER	**WER**	**BLEU**	**CER**	**WER**	**BLEU**
LipNet (sem LM)	1.46%	3.76%	96.23%	13.89%	27.33%	72.67%
LipNet (com LM)	1.42%	3.35%	96.64%	13.55%	22.91%	77.08%
LCANet (sem LM)	4.26%	15.51%	84.81%	13.66%	31.48%	68.51%
LCANet (com LM)	1.22%	2.70%	97.29%	10.59%	17.77%	82.22%

"b" e "p" são o mesmo, e depender apenas do movimento dos lábios para tomar a decisão é o mesmo que jogar uma moeda. O modelo de linguagem desambigua essa decisão, trazendo uma análise estatística sobre a posição de cada letra dentro de determinada frase ou palavra, essencialmente fazendo pequenas correções ortográficas.

4.6 Mapa de Saliência

Entender os padrões aprendidos por uma rede não é simples, uma vez que o produto de uma rede neural é apenas um conjunto de pesos. Porém, é possível visualizar os lugares que a rede considerou maior importância em determinado dado. Ou seja, conseguimos fazer um mapa de saliência para medir a importância de determinados pontos do vídeo [43]. A extração de um mapa de saliência para determinado dado é feita através dos gradientes calculados pela rede após a previsão desse dado - onde os gradientes forem grandes, o modelo declarou importante; onde os gradientes forem pequenos, o modelo considerou pouco relevante para o resultado final.

A Figura 4.2 demonstra os mapas de saliência de dois modelos diferentes, mas com previsões em cima do mesmo vídeo. O exemplo superior mostra o mapa de saliência no modelo treinado com recorte fixo. Podemos observar que, por ter acesso visual a uma área muito maior do rosto do orador, o modelo explora muito os limites da boca, como os cantos dos lábios e algumas marcas de expressão. O recorte dinâmico se limita exclusivamente a área dos lábios, sem nenhuma outra informação, e a rede se concentra no posicionamento dos dentes, da língua e o formato dos lábios. Um ponto negativo desse processamento é o redimensionamento utilizado, pois causa distorções no formato da boca e a criação de novos dados, através de interpolação. Vemos na Tabela 4.1 que o recorde dinâmico ganha apenas com a LipNet, na modalidade ocultos, enquanto o recorte fixo ganha em todos os outros casos.

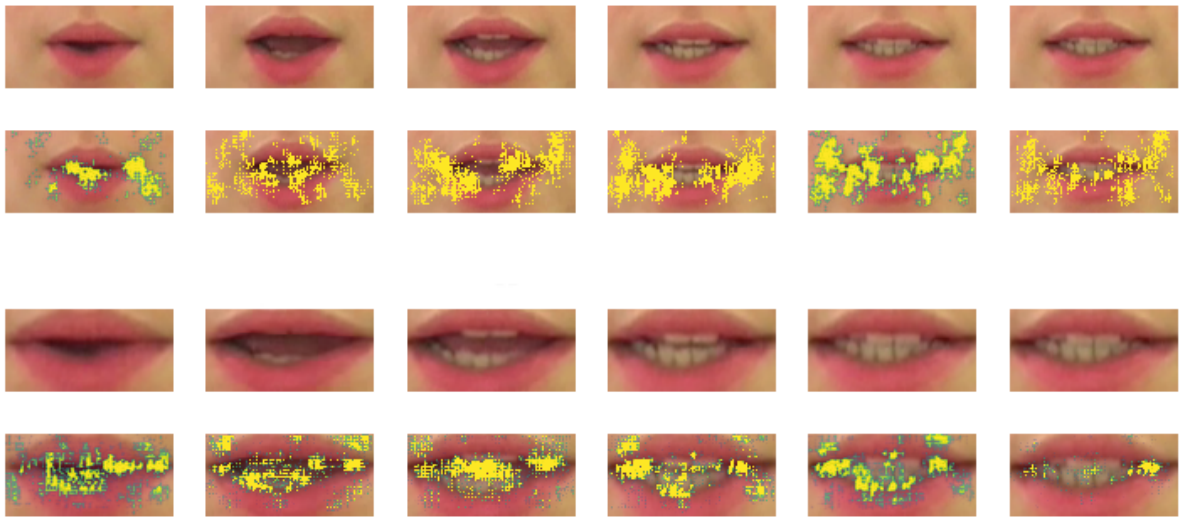

Figura 4.2: Mapa de saliência do modelo na previsão da palavra “White” com recorte fixo e recorte dinâmico, respectivamente.

Capítulo 5

Conclusão

Comparada com o estado atual do reconhecimento de voz, a leitura labial automática ainda tem muito o que evoluir. Mas analisar dessa forma é uma comparação injusta, uma vez que a leitura labial vem para cobrir os casos em que o reconhecimento por áudio falha, como em dados com muito ruído sonoro ou que o áudio se perdeu por completo. Nesses casos, a leitura labial automática se torna uma alternativa para atingir a acessibilidade.

Este trabalho começou com uma contextualização teórica sobre Aprendizado de Máquina, na Seção 2.1. Vimos que, embora existam muitas alternativas, usar Redes Neurais em problemas com imagens atinge os melhores resultados. Porém, essas redes não podem ser usadas sem pensamento crítico, uma vez que redes muito profundas podem trazer problemas com o aprendizado, e treinos mal conduzidos podem trazer problemas com a convergência do modelo. Também foram apresentadas estruturas neurais, que são formas inteligentes de organizar os neurônios de uma rede, de modo a otimizar o aprendizado para algum tipo específico de problema, como as CNN's são usadas em imagens.

Em seguida, na Seção 2.4 são apresentados os recentes trabalhos na área, cada um considerado o estado-da-arte no momento de sua publicação. A LipNet [3] é um dos trabalhos mais influentes da área, por conseguir trazer redes neurais para resolver o problema da leitura labial de frases completas. Esse trabalho propôs o uso de convoluções tridimensionais como extratores de características, e redes recorrentes como classificadores. Os trabalhos seguintes trouxeram melhorias com diferentes objetivos, mas todos seguiram as mesmas estruturas gerais e os mesmos métodos avaliativos.

Na Seção 3 foi descrita a forma que os experimentos foram conduzidos. Seu fluxo de trabalho se baseia em 3 etapas: o pré-processamento, o treino e o pós-processamento. O pré-processamento se baseia em um recorte do vídeo original, de forma que apenas a área dos lábios seja inserida na rede para treino. Os treinos foram conduzidos apenas com as redes LipNet [3] e LCANet [4], embora alguns outros testes tenham sido realizados, sem sucesso. Por fim, o pós processamento se baseia em escolher a sequência de letras de

maior probabilidade de ocorrência, baseado em regras pré-definidas da função de perda CTC [20].

Por fim, na Seção 4.5 fazemos análises sobre os resultados dos experimentos. Vemos que a atenção em cascata proposta pela LCANet traz uma convergência muito mais rápida que a estrutura da LipNet, embora seus resultados estejam relativamente próximos. Vemos também, na Tabela 4.2, que a introdução do modelo de linguagem no pós-processamento da rede, para corrigir a ortografia das frases geradas, traz resultados estritamente melhores. Também observamos o impacto dos diferentes pré-processamentos no aprendizado e como a informação é tratada pela rede, analisando os mapas de saliência gerados para cada tipo de recorte.

Este trabalho criou uma ótima fundamentação teórica, não só na área de leitura labial, mas em redes neurais como um todo. A LCANet foi um verdadeiro desafio pois, diferentemente da LipNet, o código de suas estruturas neurais (Highway, Atenção em Cascata) não estavam disponíveis, então foi necessária a compreensão profunda da estrutura para implementá-la a nível de neurônios. Embora os experimentos não tenham sido aproveitados, os estudos na LipFormer e nos Transformers de Visão também abriram portas para trazer estruturas recentes de outros trabalhos para a área de leitura labial.

Como trabalhos futuros, inicialmente eu gostaria de voltar nas arquiteturas que não pude implementar por falta de tempo, como a LipFormer [7] e a LCSNet [6], e explorar o uso de Transformers de Visão [39] no problema de maneira mais profunda. Mas, visando o mestrado acadêmico, meu maior objetivo se baseia em estudar o reconhecimento de fala utilizando tanto o áudio quanto o vídeo em apenas um modelo de aprendizado de máquina, estudando abordagens multimodais. Creio que usar exclusivamente vídeo ou áudio é um desperdício de informação, afinal, nós humanos utilizamos as duas informações ao mesmo tempo, durante a comunicação.

Referências

[1] MCGURK, HARRY e JOHN MACDONALD: *Hearing lips and seeing voices*. Nature, 264(5588):746–748, Dec 1976, ISSN 1476-4687. https://doi.org/10.1038/264746a0. 1

[2] Krizhevsky, Alex, Ilya Sutskever e Geoffrey E Hinton: *Imagenet classification with deep convolutional neural networks*. Em Pereira, F., C.J. Burges, L. Bottou e K.Q. Weinberger (editores): *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. 1

[3] Assael, Yannis M., Brendan Shillingford, Shimon Whiteson e Nando de Freitas: *Lipnet: Sentence-level lipreading*. CoRR, abs/1611.01599, 2016. http://arxiv.org/abs/1611.01599. 1, 2, 14, 27, 33

[4] Xu, Kai, Dawei Li, Nick Cassimatis e Xiaolong Wang: *Lcanet: End-to-end lipreading with cascaded attention-ctc*. CoRR, abs/1803.04988, 2018. http://arxiv.org/abs/1803.04988. 1, 2, 16, 33

[5] Margam, Dilip Kumar, Rohith Aralikatti, Tanay Sharma, Abhinav Thanda, Pujitha A. K, Sharad Roy e Shankar M. Venkatesan: *Lipreading with 3d-2d-cnn BLSTM-HMM and word-ctc models*. CoRR, abs/1906.12170, 2019. http://arxiv.org/abs/1906.12170. 1

[6] Xue, Feng, Tian Yang, Kang Liu, Zikun Hong, Mingwei Cao, Dan Guo e Richang Hong: *Lcsnet: End-to-end lipreading with channel-aware feature selection*. ACM Trans. Multimedia Comput. Commun. Appl., 19(1s), jan 2023, ISSN 1551-6857. https://doi.org/10.1145/3524620. 1, 17, 34

[7] Xue, Feng, Yu Li, Deyin Liu, Yincen Xie, Lin Wu e Richang Hong: *Lipformer: Learning to lipread unseen speakers based on visual-landmark transformers*. IEEE Transactions on Circuits and Systems for Video Technology, páginas 1–1, 2023. 1, 18, 24, 34

[8] Tran, Du, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani e Manohar Paluri: *Learning spatiotemporal features with 3d convolutional networks*. CoRR, abs/1412.0767, 2014. http://arxiv.org/abs/1412.0767. 1, 11

[9] McCarthy, John: *What is artificial intelligence?* 2004. 3

[10] Mitchell, Tom M: *Machine learning*, volume 1. McGraw-hill New York, 1997. 3

[11] Rosten, Edward e Tom Drummond: *Machine learning for high-speed corner detection*. Em Leonardis, Aleš, Horst Bischof e Axel Pinz (editores): *Computer Vision – ECCV 2006*, páginas 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg, ISBN 978-3-540-33833-8. 4

[12] Bay, Herbert, Andreas Ess, Tinne Tuytelaars e Luc Van Gool: *Speeded-up robust features (surf)*. Computer Vision and Image Understanding, 110(3):346–359, 2008, ISSN 1077-3142. https://www.sciencedirect.com/science/article/pii/S1077314207001555, Similarity Matching in Computer Vision and Multimedia. 4

[13] Lowe, D.G.: *Object recognition from local scale-invariant features*. Em *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, páginas 1150–1157 vol.2, 1999. 4

[14] McCulloch, Warren S e Walter Pitts: *A logical calculus of the ideas immanent in nervous activity*. The bulletin of mathematical biophysics, 5(4):115–133, dezembro 1943. 4

[15] Laboratory, Stanford University. Stanford Electronics Laboratories. Solid State Electronics, B. Widrow, United States. Office of Naval Research, United States. Army. Signal Corps, United States. Air Force e United States. Navy: *Adaptive "adaline" Neuron Using Chemical "memistors."*. 1960. https://books.google.com.br/books?id=Yc4EAAAAIAAJ. 4

[16] Hornik, Kurt, Maxwell Stinchcombe e Halbert White: *Multilayer feedforward networks are universal approximators*. Neural Networks, 2(5):359–366, 1989, ISSN 0893-6080. https://www.sciencedirect.com/science/article/pii/0893608089900208. 5

[17] Howard, J. e S. Gugger: *Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD*. O'Reilly Media, Incorporated, 2020, ISBN 9781492045526. https://books.google.no/books?id=xd6LxgEACAAJ. 7, 8

[18] Hastie, Trevor, Robert Tibshirani e Jerome Friedman: *The elements of statistical learning: data mining, inference and prediction*. Springer, 2ª edição, 2009. http://www-stat.stanford.edu/~tibs/ElemStatLearn/. 7

[19] Cybenko, George V., Dianne P. O'Leary, Jorma Rissanen e Ima Program on Mathematics in High-Performance Computing: *The mathematics of information coding, extraction, and distribution*. 1999. 8

[20] Graves, Alex, Santiago Fernández, Faustino Gomez e Jürgen Schmidhuber: *Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks*. Em *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, página 369–376, New York, NY, USA, 2006. Association for Computing Machinery, ISBN 1595933832. https://doi.org/10.1145/1143844.1143891. 9, 34

[21] Cun, Y. Le, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel e D. Henderson: *Handwritten Digit Recognition with a Back-Propagation Network*, página 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990, ISBN 1558601007. 9

[22] Amari, S. I.: *Learning patterns and pattern sequences by self-organizing nets of threshold elements*. IEEE Transactions on Computers, C-21(11):1197–1206, 1972. 11

[23] Hochreiter, Sepp: *Untersuchungen zu dynamischen neuronalen netzen*. abril 1991. 11

[24] Cho, Kyunghyun, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk e Yoshua Bengio: *Learning phrase representations using RNN encoder-decoder for statistical machine translation*. CoRR, abs/1406.1078, 2014. `http://arxiv.org/abs/1406.1078`. 12

[25] Bahdanau, Dzmitry, Kyunghyun Cho e Yoshua Bengio: *Neural machine translation by jointly learning to align and translate*. Em Bengio, Yoshua e Yann LeCun (editores): *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. `http://arxiv.org/abs/1409.0473`. 12

[26] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser e Illia Polosukhin: *Attention is all you need*. CoRR, abs/1706.03762, 2017. `http://arxiv.org/abs/1706.03762`. 13

[27] Ioffe, Sergey e Christian Szegedy: *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. CoRR, abs/1502.03167, 2015. `http://arxiv.org/abs/1502.03167`. 14

[28] Fukushima, Kunihiko: *Visual feature extraction by a multilayered network of analog threshold elements*. IEEE Transactions on Systems Science and Cybernetics, 5(4):322–333, 1969. 14

[29] Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever e Ruslan Salakhutdinov: *Dropout: A simple way to prevent neural networks from overfitting*. J. Mach. Learn. Res., 15(1):1929–1958, jan 2014, ISSN 1532-4435. 14

[30] Ranzato, Marc'Aurelio, Fu Jie Huang, Y Lan Boureau e Yann LeCun: *Unsupervised learning of invariant feature hierarchies with applications to object recognition*. Em *2007 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 1–8, 2007. 14

[31] Glorot, Xavier, Antoine Bordes e Yoshua Bengio: *Deep sparse rectifier neural networks*. Em Gordon, Geoffrey, David Dunson e Miroslav Dudík (editores): *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 de *Proceedings of Machine Learning Research*, páginas 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. `https://proceedings.mlr.press/v15/glorot11a.html`. 15

[32] Scherer, Dominik, Andreas Müller e Sven Behnke: *Evaluation of pooling operations in convolutional architectures for object recognition.* Em Diamantaras, Konstantinos, Wlodek Duch e Lazaros S. Iliadis (editores): *Artificial Neural Networks – ICANN 2010*, páginas 92–101, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg, ISBN 978-3-642-15825-4. 15

[33] Srivastava, Rupesh Kumar, Klaus Greff e Jürgen Schmidhuber: *Training very deep networks.* CoRR, abs/1507.06228, 2015. `http://arxiv.org/abs/1507.06228`. 16

[34] Hochreiter, Sepp e Jürgen Schmidhuber: *Long short-term memory.* Neural Comput., 9(8):1735–1780, nov 1997, ISSN 0899-7667. `https://doi.org/10.1162/neco.1997.9.8.1735`. 16

[35] Woo, Sanghyun, Jongchan Park, Joon-Young Lee e In So Kweon: *CBAM: convolutional block attention module.* CoRR, abs/1807.06521, 2018. `http://arxiv.org/abs/1807.06521`. 17

[36] Cooke, Martin, Jon Barker, Stuart P. Cunningham e Xu Shao: *An audio-visual corpus for speech perception and automatic speech recognition.* The Journal of the Acoustical Society of America, 120 5 Pt 1:2421–4, 2006. 19, 29

[37] King, Davis E.: *Dlib-ml: A machine learning toolkit.* J. Mach. Learn. Res., 10:1755–1758, dec 2009, ISSN 1532-4435. 20, 27

[38] Sagonas, Christos, Georgios Tzimiropoulos, Stefanos Zafeiriou e Maja Pantic: *300 faces in-the-wild challenge: The first facial landmark localization challenge.* Em *2013 IEEE International Conference on Computer Vision Workshops*, páginas 397–403, 2013. 20

[39] Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit e Neil Houlsby: *An image is worth 16x16 words: Transformers for image recognition at scale.* CoRR, abs/2010.11929, 2020. `https://arxiv.org/abs/2010.11929`. 24, 34

[40] Hassani, Ali, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li e Humphrey Shi: *Escaping the big data paradigm with compact transformers.* CoRR, abs/2104.05704, 2021. `https://arxiv.org/abs/2104.05704`. 24

[41] Papineni, Kishore, Salim Roukos, Todd Ward e Wei Jing Zhu: *Bleu: A method for automatic evaluation of machine translation.* Em *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, página 311–318, USA, 2002. Association for Computational Linguistics. `https://doi.org/10.3115/1073083.1073135`. 28

[42] Kingma, Diederik P. e Jimmy Ba: *Adam: A method for stochastic optimization.* Em Bengio, Yoshua e Yann LeCun (editores): *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. `http://arxiv.org/abs/1412.6980`. 29

[43] Simonyan, Karen, Andrea Vedaldi e Andrew Zisserman: *Deep inside convolutional networks: Visualising image classification models and saliency maps*. Em *Workshop at International Conference on Learning Representations*, 2014. 31