



Programa de Pós-Graduação em Ciência da Computação

Álvaro Leandro Cavalcante Carneiro

Reconhecimento de palavras em língua de sinais baseado em
aprendizado profundo e descritores *handcrafted* de baixo custo

Rio Claro - SP

2023

UNIVERSIDADE ESTADUAL PAULISTA

“Júlio de Mesquita Filho”

Instituto de Geociências e Ciências Exatas

Câmpus de Rio Claro

ÁLVARO LEANDRO CAVALCANTE CARNEIRO

**Reconhecimento de palavras em língua de sinais baseado em
aprendizado profundo e descritores *handcrafted* de baixo
custo**

Dissertação de mestrado apresentada ao Instituto de Geociências e Ciências Exatas do Câmpus de Rio Claro, da Universidade Estadual Paulista “Júlio de Mesquita Filho”, como parte dos requisitos para a obtenção do título de mestre no Programa de Pós-Graduação em Ciência da Computação.

Orientador: Dr. Denis Henrique Pinheiro Salvadeo

Rio Claro - SP

2023

C289r Carneiro, Álvaro Leandro Cavalcante
Reconhecimento de palavras em língua de sinais baseado em
aprendizado profundo e descritores handcrafted de baixo custo /
Álvaro Leandro Cavalcante Carneiro. -- Rio Claro, 2023
159 p. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp),
Instituto de Geociências e Ciências Exatas, Rio Claro
Orientador: Denis Henrique Pinheiro Salvadeo

1. Língua de sinais. 2. Inteligencia artificial. 3. Processamento de
imagens. 4. Reconhecimento de padrões. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de
Geociências e Ciências Exatas, Rio Claro. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

“Júlio de Mesquita Filho”

Instituto de Geociências e Ciências Exatas

Câmpus de Rio Claro

ÁLVARO LEANDRO CAVALCANTE CARNEIRO

Reconhecimento de palavras em língua de sinais baseado em aprendizado profundo e descritores *handcrafted* de baixo custo

Dissertação de mestrado apresentada ao Instituto de Geociências e Ciências Exatas do Câmpus de Rio Claro, da Universidade Estadual Paulista “Júlio de Mesquita Filho”, como parte dos requisitos para a obtenção do título de mestre no Programa de Pós-Graduação em Ciência da Computação.

Comissão Examinadora

Prof. Dr. DENIS HENRIQUE PINHEIRO SALVADEO

IGCE/UNESP/Rio Claro (SP)

Prof. Dr. APARECIDO NILCEU MARANHA

FC/Unesp/Bauru (SP)

Prf. Dr. CLEBER ZANCHETTIN

CIn/UFPE/Recife (PE)

Conceito: Aprovado

Rio Claro (SP), 02 de Março de 2023.

Agradecimentos

Gostaria de agradecer primeiramente a Deus pela força e capacitação necessárias dadas a mim para possibilitar a conclusão desse programa de pós graduação. Também, não poderia deixar de agradecer imensamente a toda a minha família, por sempre acreditarem em mim e me apoiarem constantemente nos estudos, em especial ao meu avô Edivaci, por ser a base de toda a minha criação, e à minha esposa Esther Sato, que sempre esteve ao meu lado e foi minha ajudadora nos momentos mais difíceis.

Agradeço ao meu orientador Denis Salvadeo por ter acreditado em mim e no projeto proposto, se mostrando sempre disponível a ajudar e sendo fundamental para o desenvolvimento do trabalho através de suas ótimas sugestões e ensinamentos.

Por fim, quero agradecer a todos os meus colegas do grupo de pesquisa PRISMA (*Pattern Recognition, Image and Signal Processing, and Multimedia Applications*) pela parceria e colaboração constante através de ideias e discussões diversas, em especial ao Darlan Nakamura, Davi Duarte e ao Lucas de Brito.

Resumo

Diversas técnicas de aprendizado profundo e visão computacional têm sido utilizadas nos últimos anos para a criação de sistemas de reconhecimento e tradução de língua de sinais para a língua nativa a partir de vídeos, servindo como uma ferramenta de comunicação para os milhões de deficientes auditivos ao redor do mundo. Ainda assim, inúmeros fatores devem ser considerados para a criação de um sistema como esse, aumentando a complexidade da tarefa. Primeiramente, o treinamento de um modelo de classificação exige uma grande quantidade de dados, o que representa uma dificuldade visto que esta área sofre com a carência de bases de dados em larga escala disponíveis publicamente. Além disso, a fim de evitar a ambiguidade entre as palavras, é preciso considerar o maior número de parâmetros linguísticos possíveis na execução dos gestos que formam os sinais. Na prática, acrescenta-se ainda que o conjunto tecnológico adotado seja condizente com a realidade, evitando sensores custosos, intrusivos ou com baixa mobilidade, bem como arquiteturas muito complexas de aprendizado profundo, reduzindo os requisitos computacionais. Isso é importante para possibilitar a adoção em larga escala e em ambiente real da solução criada, promovendo a acessibilidade. Baseado nisso, este trabalho tem como objetivo propor um sistema eficiente de reconhecimento de palavras em língua de sinais, adotando sensores e técnicas de baixo custo. Para isso, diversas arquiteturas de detecção de objetos foram simplificadas e treinadas especificamente para a tarefa de detecção da face e das mãos do intérprete, garantindo o foco nas regiões mais relevantes da imagem e gerando entradas com maior valor semântico para o classificador. Além disso, uma nova abordagem foi proposta para obtenção de atributos utilizados para representação do ponto de contato e do movimento das mãos, aumentando a discriminação entre os sinais. Os resultados obtidos demonstram a eficiência dos detectores, atingindo um mAP de mais de 95% com a velocidade de inferência até 77% menor em relação à versão original do modelo. Ademais, os atributos *handcrafted* (criados sem a ajuda de modelos) geraram um acréscimo de 7,96% de acurácia em um dos conjuntos de dados testados, adicionando menos de 700 mil parâmetros, demonstrando o potencial da técnica em aumentar a compensação entre custo computacional e acurácia.

Palavras-chave: Língua de sinais; Inteligência artificial; Processamento de imagens; Reconhecimento de padrões.

Abstract

Several deep learning and computer vision techniques have been used in recent years to create systems for recognizing and translating sign language into native language from videos, serving as a communication tool for the millions of hearing impaired people around the world. Nevertheless, a number of factors must be considered to create such a system, increasing the complexity of the task. First of all, training a classification model requires a large amount of data, which represents a difficulty, since this area suffers from the lack of large-scale publicly available datasets. Moreover, in order to avoid ambiguity between words, it is necessary to consider as many linguistic parameters as possible in the execution of the gestures that form the signs. In practice, it is also added that the technological set adopted is consistent with reality, avoiding expensive, intrusive, or low-mobility sensors, as well as very complex deep learning architectures, reducing the computational requirements. This is important to allow a large-scale and real-world adoption of the created solution, promoting accessibility. Based on this, this work aims to propose an efficient system for sign language recognition using low-cost sensors and techniques. To this end, several object detection architectures were simplified and trained specifically for the task of detecting the interpreter's face and hands, ensuring the focus on the most relevant regions of the image and generating inputs with higher semantic value for the classifier. In addition, a new approach was proposed for obtaining attributes used to represent the point of contact and hand movement, increasing the discrimination between the signs. The obtained results highlight the efficiency of the detectors, reaching an mAP of more than 95% and an inference speed up to 77% higher compared to the original version of the model. Furthermore, the handcrafted attributes (created without the help of models) increased the accuracy by 7.96% in one of the tested datasets with the addition of fewer than 700 thousand parameters, demonstrating the potential of the technique to improve the tradeoff between computational cost and accuracy.

Keywords: Sign language; Artificial intelligence; Image processing; Pattern recognition.

Lista de figuras

1.1	Localização de cada um dos parâmetros linguísticos em uma imagem 2D contendo um sinal executado em LIBRAS.	16
1.2	Diferença no sinal de “D” (à esquerda) e “DEUS” (à direita) em LIBRAS. . . .	16
1.3	Diferença semântica entre sinais baseado na (a) orientação da palma da mão, (b) no número de mãos e (c) nas marcações não-manuais.	17
1.4	Sinais de HOJE/AGORA/PRESENTE.	17
1.5	Luva virtual de baixo custo para captura de gestos da mão através de sensores de curvatura.	18
2.1	Esquema simplificado de um neurônio biológico.	23
2.2	Modelo de um neurônio artificial do tipo Perceptron.	24
2.3	Modelo do tipo <i>Multi Layer Perceptron</i> (MLP)	25
2.4	Representação da função sigmoide.	27
2.5	Representação da função tangente hiperbólica.	27
2.6	Representação da função ReLu.	28
2.7	Representação da função <i>Leaky</i> ReLU.	29
2.8	Otimização de uma função de custo não convexa.	31
2.9	Convergência da função de custo com (em vermelho) e sem o uso do Momentum no algoritmo de otimização SGD.	34
2.10	Ilustração do efeito na convergência causado pelo algoritmo AdaGrad.	37
2.11	Exemplo da superfície de decisão de um modelo que está altamente ajustado aos dados (<i>overfitting</i>).	40
2.12	Comparativo entre uma rede neural sem nenhum tipo de regularização, à esquerda, e a aplicação da regularização baseada em <i>dropout</i> , à direita.	41
2.13	Representação visual do ponto de parada desejado para o modelo, evitando que o sobreajuste na base de treinamento degrade a capacidade de generalização. .	42
2.14	Neurônio recorrente e seu desenrolar através do tempo.	45
2.15	Arquitetura de RNN do tipo sequência-para-vetor.	46

2.16	Célula LSTM.	48
2.17	Célula GRU.	50
2.18	Representação da estrutura das BRNNs.	52
2.19	Representação do uso de vetores de contexto em redes recorrentes bidirecionais.	54
2.20	Exemplo de uma operação de convolução.	56
2.21	Exemplo de uma arquitetura de CNN com o empilhamento das camadas de extração de característica (convolução e <i>max pooling</i>) e o classificador (<i>Multilayer Perceptron</i>)	59
2.22	Aumento de dados aplicado no sinal ÔNIBUS em LIBRAS através de transformações geométricas e de intensidade.	61
2.23	Exemplo de uma arquitetura de transferência de aprendizado.	62
2.24	Operação básica de convolução 2+1D.	65
2.25	Exemplo de uma arquitetura de CNN multicanal no contexto do reconhecimento de sinais.	66
2.26	Processo de fusão tardia para classificação.	67
2.27	Exemplo do uso da detecção de objetos em diferentes cenas.	69
2.28	Exemplo de intersecção sobre união.	70
2.29	Arquitetura de detecção de objetos YOLO.	74
2.30	Arquitetura de detecção de objetos EfficientDet.	76
2.31	Resultados da inferência de poses corporais em diferentes cenários e situações.	77
2.32	Múltiplas poses estimadas por problemas na detecção de objetos.	78
2.33	Arquitetura do OpenPose.	79
3.1	Arquitetura multimodal utilizada por Belissen et al. (2019).	89
3.2	Arquitetura multimodal utilizada por Xiao et al. (2021).	92
4.1	Exemplos de imagens da base de dados AUTSL Sincan e Keles (2020)	102
4.2	Exemplos de imagens da base de dados CSL Pu et al. (2016)	103
4.3	Exemplos de imagens com as mãos e a face anotadas das bases de dados públicas compiladas por Mohaimenianpour e Vaughan (2018)	104
4.4	Comparação entre diferentes proporções de caixas âncoras, considerando o padrão do modelo SSD (em vermelho) e as otimizadas para o contexto de língua de sinais (em verde).	107

4.5	Exemplo mostrando a imagem original (canto superior esquerdo) seguido de operações de aumento de dados, sendo elas: rotação horizontal, escala de cinza, adição de <i>patches</i> pretos, distorção nos canais de cor e ajustes no contraste . . .	109
4.6	Figura triangular gerada pela ligação entre os centroides das caixas delimitadoras.	111
4.7	Arquitetura de reconhecimento e criação de figura geométrica proposta por Xiao et al. (2020)	112
4.8	Comparação entre o formato e os valores do triângulo considerando o mesmo sinal capturado de formas diferentes. Os ângulos internos são representados em azul, enquanto as distâncias são mostradas em laranja.	114
4.9	Figura do triângulo gerada a partir da execução dos sinais.	116
4.10	Amostra de imagens da face e das mãos em um vídeo com aumento de dados aleatório.	117
4.11	Uso do valor de espaçamento para amostragem dos quadros do vídeo.	118
4.12	Quadros de diferentes intervalos de tempo (indicado pelos números em vermelho) de um dos vídeos da base AUTSL.	120
4.13	Comparação entre as caixas delimitadoras geradas pelo modelo (em vermelho) em relação ao aumento manual de 10% em seu tamanho (em verde).	121
4.14	Comparação entre um quadro original (à esquerda) e um quadro cortado (à direita) do conjunto CSL.	121
4.15	Processo utilizado para diferenciação das duas mãos	122
4.16	Unificação das imagens das mãos em um único canal.	124
4.17	Arquitetura proposta para o reconhecimento de sinais.	125
5.1	Intersecção visual entre os canais da face e das mãos.	134
5.2	Efeito do borrão de movimento no canal das mãos do conjunto AUTSL.	137
5.3	Sinal de “MASCULINO” (em cima) e “JOALHERIA” (em baixo) na base de dados CSL.	137
5.4	Sinal de “BOLINHO FRITO” (em cima) e “SORVETE” (em baixo) na base de dados CSL.	138

Lista de tabelas

1	Comparação dos conjuntos de dados em larga escala (mais do que 50 sinais) criados manualmente e com pelo menos 5 intérpretes.	85
2	Comparação de conjuntos de sinais em larga escala criados através da mineração de conteúdo.	86
3	Comparação entre trabalhos que utilizam informações de pose.	93
4	Comparação entre os trabalhos que utilizam detectores de objetos.	95
5	Comparação entre os trabalhos que utilizam outras ou nenhuma técnica de pré-processamento.	97
6	Comparação de trabalhos que utilizam modelos do tipo GCN.	98
7	Variação nos intervalos de hiperparâmetros para cada canal de RNN.	125
8	Variação nos intervalos de hiperparâmetros para cada canal de rede neural recorrente.	126
9	Resultado do treinamento dos detectores de objetos.	130
10	Desempenho obtido pelos detectores de objetos não simplificados.	131
11	Arquiteturas geradas para cada uma das redes recorrentes propostas.	132
12	Comparação dos resultados dos modelos no conjunto de dados AUTSL.	133
13	Comparação dos resultados dos modelos no conjunto de dados CSL.	136

Sumário

1	Introdução	14
1.1	Língua de sinais	14
1.2	Sensores para a captura de sinais	18
1.3	Técnicas para o reconhecimento de sinais	19
1.4	Objetivos	21
1.5	Organização da dissertação	21
2	Fundamentação Teórica	23
2.1	Redes neurais artificiais	23
2.1.1	Funções de ativação	26
2.1.1.1	Função sigmoide	26
2.1.1.2	Função tangente hiperbólica	27
2.1.1.3	Função ReLU	28
2.1.1.4	Função <i>Leaky ReLU</i>	28
2.1.1.5	Função Softmax	29
2.1.2	Função de custo	30
2.1.3	Funções de otimização	31
2.1.3.1	Problemas comuns na otimização de redes neurais	31
2.1.3.2	Gradiente descendente	32
2.1.3.3	Gradiente descendente estocástico	32
2.1.3.4	Momentum	33
2.1.3.5	Programação da taxa de aprendizado	34
2.1.3.6	AdaGrad	36
2.1.3.7	RMSProp	37
2.1.3.8	Adam	37
2.1.4	Retropropagação do erro	38
2.1.5	<i>Overfitting e underfitting</i>	39
2.1.6	Regularização	40

2.1.6.1	<i>Dropout</i>	41
2.1.6.2	<i>Early Stopping</i>	42
2.1.7	Métricas de avaliação	42
2.1.8	Considerações finais da Seção	44
2.2	Redes neurais recorrentes	45
2.2.1	Células LSTM	47
2.2.2	Células GRU	49
2.2.3	Redes recorrentes bidirecionais	51
2.2.4	Camadas de atenção	52
2.2.5	Considerações finais da seção	54
2.3	Redes neurais convolucionais	55
2.3.1	Operação de convolução	55
2.3.2	<i>Pooling</i>	58
2.3.3	Classificadores	58
2.3.4	<i>Global Average Pooling</i>	59
2.3.5	<i>Batch normalization</i>	59
2.3.6	Aumento de dados	60
2.3.7	Transferência de aprendizado	61
2.3.8	Redes neurais convolucionais 3D	63
2.3.9	Redes Neurais Convolucionais 2+1D	65
2.3.10	Redes neurais convolucionais multicanais	66
2.3.11	Considerações finais da Seção	68
2.4	Detecção de objetos	68
2.4.1	Avaliação dos detectores de objetos	69
2.4.2	Principais arquiteturas de detecção de objetos	72
2.4.3	Considerações finais da seção	76
2.5	Estimativa de pose corporal	76
2.5.1	Abordagem <i>top-down</i>	77
2.5.2	Abordagem <i>bottom-up</i>	79
2.5.3	Considerações finais da seção	80
2.6	Considerações finais do capítulo	80

3	Reconhecimento de língua de sinais com aprendizado profundo	81
3.1	Conjuntos de dados para o reconhecimento de sinais isolados	82
3.1.1	Coleta manual de dados	82
3.1.2	Criação de bases de dados através da mineração de conteúdo	84
3.1.3	Considerações finais da Seção	86
3.2	Trabalhos na área do reconhecimento de língua de sinais	87
3.2.1	Redes Neurais Convolucionais	87
3.2.1.1	Extração de pose corporal	88
3.2.1.2	Deteção de objetos	94
3.2.1.3	Demais técnicas de pré-processamento	95
3.2.1.4	Considerações finais da Seção	96
3.2.2	Redes Convolucionais em Grafos	97
3.3	Considerações finais do Capítulo	98
4	Desenvolvimento do Projeto	100
4.1	Materiais e métodos	100
4.1.1	Tecnologias e <i>hardware</i>	101
4.1.2	Bases de dados para classificação	101
4.1.3	Bases de dados para detecção de objetos	103
4.1.4	Pré-processamento do sistema	105
4.1.4.1	Deteção de dicas visuais para língua de sinais	105
4.1.4.2	Representação do movimento e do ponto de contato	110
4.1.4.3	Figura do triângulo	115
4.1.4.4	Aumento de dados	116
4.1.4.5	Pré-processamento das imagens para classificação	117
4.1.5	Sistema proposto para classificação dos sinais	123
4.1.6	Classificação dos sinais com a figura do triângulo	126
4.2	Metodologia experimental	127
4.3	Considerações finais do Capítulo	129
5	Resultados e Discussões	130
5.1	Resultados da detecção de objetos	130
5.2	Resultados dos atributos <i>handcrafted</i>	132

6	Conclusões	140
	Referências Bibliográficas	143

1 Introdução

A comunicação é fundamental para a vida em sociedade, onde ela é utilizada diariamente pelas pessoas para se expressarem e também ter acesso a serviços essenciais como educação, transporte e saúde. Neste sentido, a língua de sinais é a forma de comunicação adotada em diversos países por pessoas com perda auditiva profunda, uma condição que atinge, só no Brasil, mais de 2,1 milhões de indivíduos (IBGE., 2010). Ainda segundo a [Organização Mundial da Saúde \(OMS\)](#). (2021), existem 430 milhões de pessoas ao redor do mundo com perda auditiva incapacitante, sendo estimado mais de 700 milhões até 2050.

Ainda que ferramentas tecnológicas como *smartphones* e aplicativos de troca de mensagens tenham facilitado a comunicação escrita entre as pessoas, a ([Federação Mundial dos Surdos.](#), 2003) estima que 80% dos deficientes auditivos têm baixo ou nenhum nível de escolaridade, sendo analfabetos ou semianalfabetos em suas línguas nativas, inviabilizando este meio de comunicação. Além disso, é notório que a maior parte dos ouvintes não sabem como falar a língua de sinais, criando uma barreira que dificulta as interações sociais dessa comunidade.

Por outro lado, os avanços da inteligência artificial e visão computacional dos últimos anos têm chamado a atenção de pesquisadores na área da interação humano-máquina, que pode ser, entre outras coisas, uma poderosa ferramenta de assistência à comunicação dos deficientes auditivos por meio do reconhecimento e tradução da língua de sinais para a língua nativa através da geração de legendas, contribuindo com a inclusão social dessas pessoas e mitigando os impactos causados pela deficiência.

Deste modo, este capítulo tem por objetivo introduzir os principais conceitos e desafios da área, apresentando na Seção 1.1 os aspectos que compõe os sinais que representam as palavras na língua de sinais. Em seguida, as Seções 1.2 e 1.3 exploram os principais sensores e técnicas utilizados no estado da arte da tarefa de reconhecimento de sinais, destacando as suas vantagens e desvantagens. Por fim, a Seção 1.4 apresenta os principais objetivos do trabalho, enquanto a Seção 1.5 descreve a forma como o conteúdo deste documento de dissertação foi organizado.

1.1 Língua de sinais

A língua de sinais é uma das formas de comunicação utilizada pelos deficientes auditivos que ocorre através da representação visual da língua nativa, e é adotada por inúmeros países ao redor do mundo. A palavra “sinal”, neste contexto, diz respeito à execução de um gesto

semântico em uma determinada língua de sinais, sendo que cada sinal é geralmente relacionado a uma palavra.

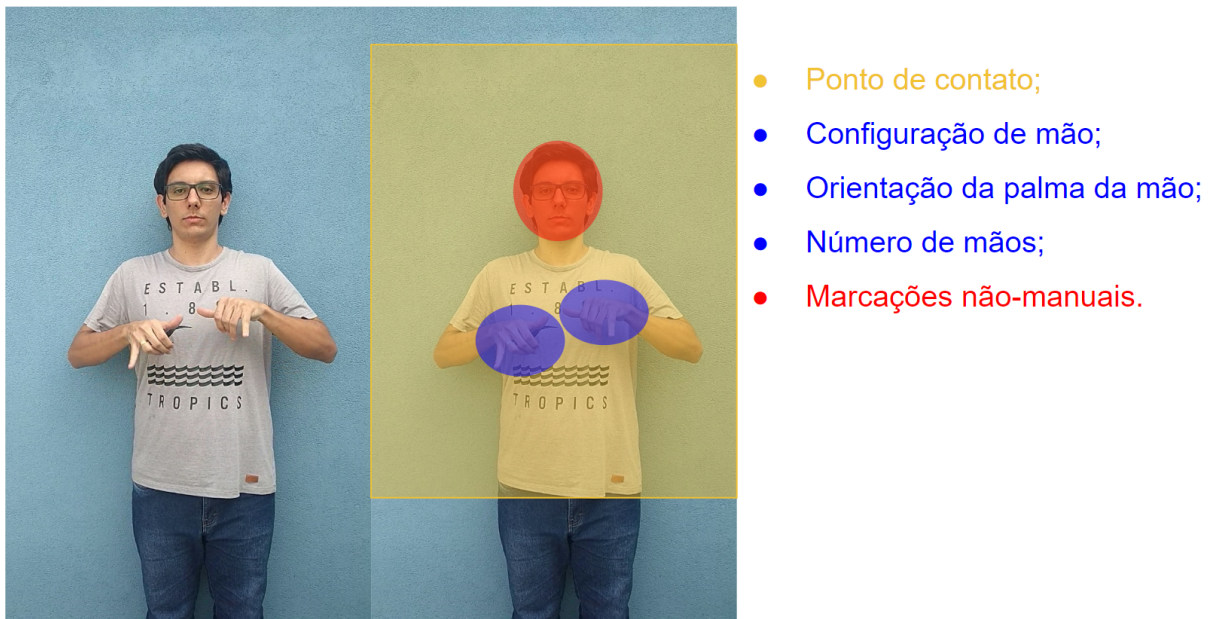
Assim como qualquer língua, existem diferenças nos sinais não só entre idiomas distintos (língua de sinais Brasileira, Americana, Indiana e entre outras), mas também em um mesmo idioma, devido a fatores regionais e culturais. Ainda assim, a estrutura base que compõe um sinal é a mesma, devendo ser compreendida para possibilitar a criação de um sistema de reconhecimento de sinais eficiente, independente de qual língua nativa, ou variação da língua, se está considerando.

Xavier e Barbosa (2014) apontam que os principais parâmetros linguísticos que constituem os sinais da Língua Brasileira de Sinais (LIBRAS) são: configuração de mão, localização, movimento, orientação, número de mãos e marcações não-manuais. A configuração de mão consiste na disposição dos dedos de ambas as mãos durante a produção do sinal. A localização, ou ponto de contato, como também é conhecido, refere-se à posição espacial ou parte do corpo em que o sinal é executado. O movimento diz respeito à forma e velocidade em que as mãos se movem durante a execução do sinal. As marcações não-manuais são caracterizadas pelo uso de outras partes do corpo para transmitir o significado do sinal, sendo principalmente composto pelas expressões faciais. Por fim, além do número de mãos utilizadas, a orientação da palma da mão também é levada em consideração, podendo ou não ser virada para o intérprete. Embora os autores utilizem a LIBRAS como base para o estudo, os parâmetros supracitados também se aplicam para as demais línguas de sinais ao redor do mundo.

Dito isso, a Figura 1.1 destaca o local da imagem onde cada um dos parâmetros se encontram (com exceção do movimento, por se tratar de uma única imagem 2D), sendo possível observar a relação entre a proporção espacial ocupada e a relevância semântica do segmento (através do número de parâmetros ali representados). Deste modo, fica evidente como a maior parte da informação utilizada para discernir os sinais é contida em uma fração pequena da imagem, constituída pelas mãos e face do intérprete. O ponto de contato, em contrapartida, depende de praticamente toda a informação espacial disponível na imagem para sua representação, visto que os sinais são realizados em qualquer local acima da cintura do indivíduo. A Figura 1.2, por exemplo, ilustra o efeito deste parâmetro na formação dos sinais da letra “D” e da palavra “DEUS” em LIBRAS, sendo diferenciados apenas pela localização das mãos.

A Figura 1.3, por sua vez, ilustra a influência dos demais parâmetros na execução e no significado dos sinais em LIBRAS, onde o símbolo “@” representa que o sinal é o mesmo

Figura 1.1: Localização de cada um dos parâmetros linguísticos em uma imagem 2D contendo um sinal executado em LIBRAS.



Fonte: Autor.

Figura 1.2: Diferença no sinal de “D” (à esquerda) e “DEUS” (à direita) em LIBRAS.

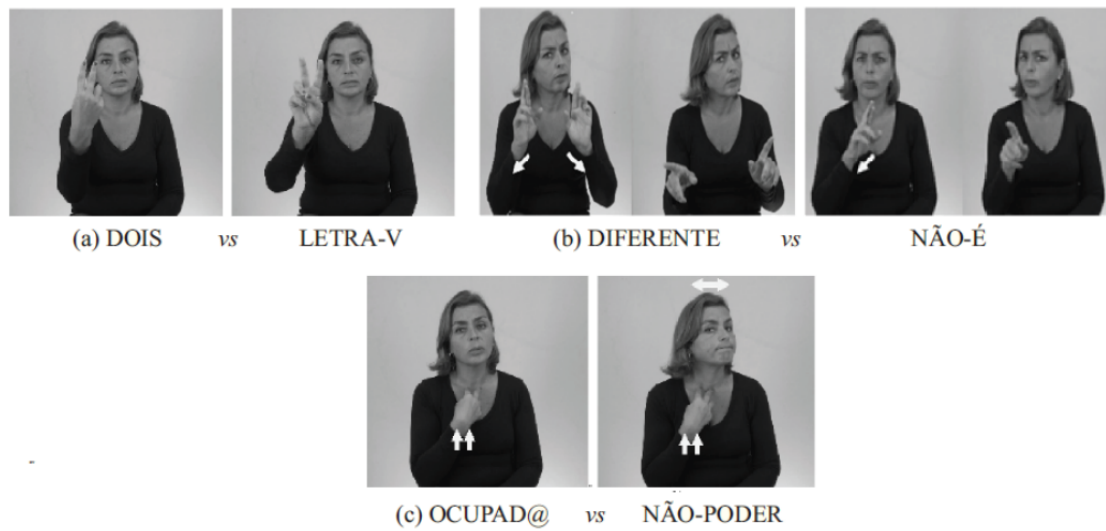


Fonte: Autor.

independente do gênero, como “ocupado” e “ocupada”, por exemplo. Conforme observado na figura, alguns sinais possuem a maioria dos parâmetros similares, sendo diferenciados por um ou dois deles. Os sinais de “OCUPAD@” e “NÃO-PODER”, por exemplo, se distinguem apenas pelo movimento da cabeça e expressão facial. Outra propriedade importante a ser considerada a respeito dos sinais é a polissemia, ou seja, sua capacidade de apresentar múltiplos

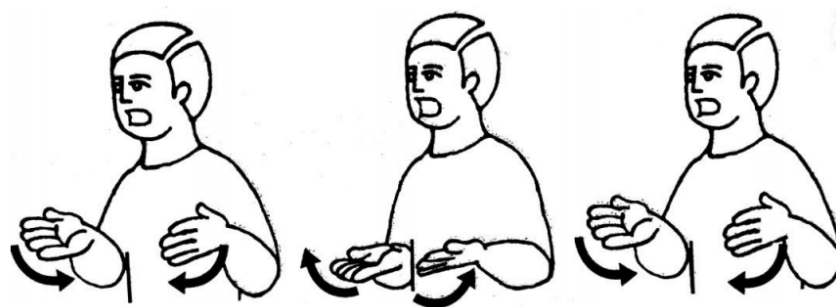
significados dependendo do contexto. Sinais polissêmicos são idênticos em relação aos parâmetros linguísticos, gerando uma ambiguidade lexical (Mendonça e Militão, 2018), fazendo com que só seja possível identificar o significado do sinal através do contexto em que a fala se insere. A Figura 1.4 ilustra um exemplo de polissemia.

Figura 1.3: Diferença semântica entre sinais baseado na (a) orientação da palma da mão, (b) no número de mãos e (c) nas marcações não-manuais.



Fonte: Xavier e Barbosa (2014).

Figura 1.4: Sinais de HOJE/AGORA/PRESENTE.



Fonte: Capovilla et al. (2009).

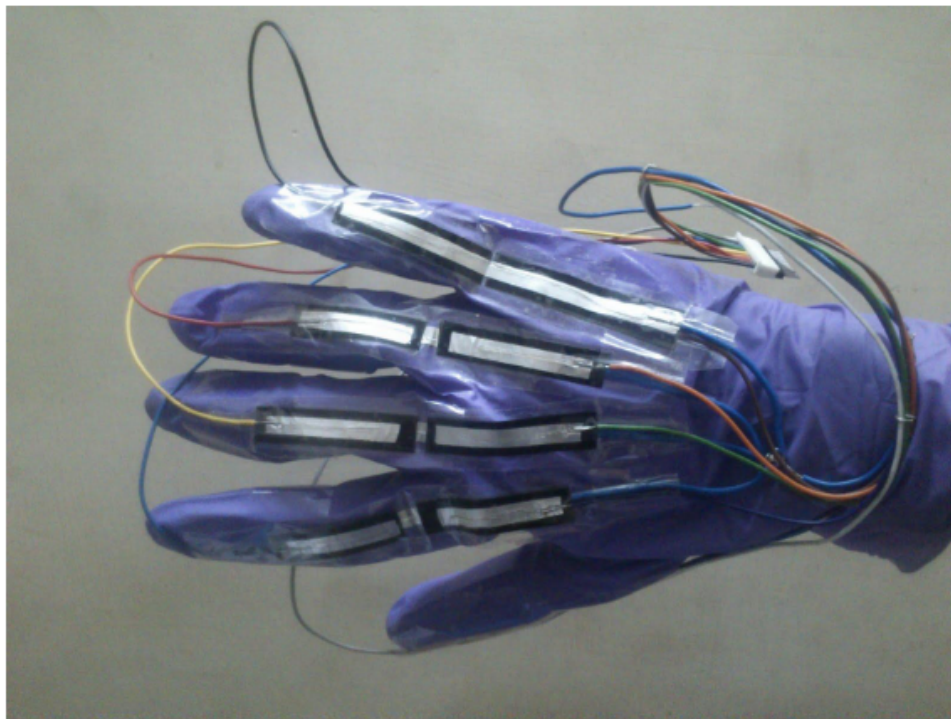
Por fim, embora existam diversos sinais estáticos, a maior parte dos que compõe a língua de sinais são dinâmicos, ou seja, que dependem de movimento (Cruz, 2020). Portanto, a captura desses sinais deve ser feita em formato de vídeo ou através de sequências de imagens que representem a trajetória do gesto. Além disso, mesmo sinais estáticos podem possuir dependência temporal, devido à necessidade de contextualização gerada por alguns sinais, como é o caso dos polissêmicos.

1.2 Sensores para a captura de sinais

Uma forma de categorizar trabalhos anteriores da área de reconhecimento de sinais é através do tipo de sensor utilizado na captura do conjunto de dados, uma vez que essa decisão impacta em toda a execução posterior da solução a ser desenvolvida. Dito isso, os três principais sensores encontrados na literatura são as luvas virtuais, os sensores de profundidade e as câmeras fotográficas no espaço de cor RGB (*Red, Green, Blue* em inglês).

As luvas virtuais, também chamadas de *data gloves*, são dispositivos vestíveis acoplados às mãos do usuário e que possuem diversos sensores de curvatura responsáveis por capturar o movimento das falanges dos dedos, conforme ilustrado pela Figura 1.5. Embora esforços tenham sido realizados para se criar luvas de baixo custo (Preetham et al., 2013), esse ainda é um dos desafios de se adotar uma solução como essa em larga escala, visto que esse equipamento é projetado para uso individual, dados os diferentes aspectos morfológicos das mãos de cada indivíduo. Além disso, a falta de praticidade no uso de uma luva como essa no cotidiano acaba inibindo ainda mais a adoção da mesma em ambiente real.

Figura 1.5: Luva virtual de baixo custo para captura de gestos da mão através de sensores de curvatura.



Fonte: Preetham et al. (2013).

Deste modo, ainda que sensores físicos sejam altamente precisos na captura do sinal, as luvas ficam limitadas apenas às informações das mãos, sendo insuficientes para o reconhecimento

de sinais dependentes de expressão facial, além de ser necessária a inclusão de sensores de movimento (como um giroscópio, por exemplo) para a captura do ponto de contato em que o sinal é executado e dos movimentos realizados, elevando ainda mais o custo e falta de portabilidade do dispositivo.

Sensores de profundidade como o *Leap Motion* (Potter et al., 2013) e o *Kinect* (Han et al., 2013), por outro lado, são capazes de capturar o sinal de maneira menos intrusiva, através de imagens em escala de cinza ou RGB e com uma dimensão de profundidade, gerando um mapeamento mais preciso dos movimentos do usuário. Ainda assim, estes sensores sofrem com problemas similares às luvas de dados para adaptação ao cotidiano e adoção em larga escala, visto o custo elevado dos mesmos e dependência de um dispositivo externo para processamento dos dados capturados.

Por fim, os sensores RGB são dispositivos caracterizados por sua mobilidade e baixo custo, como câmeras fotográficas digitais e aparelhos celulares, sendo o foco deste trabalho. Embora a aquisição de conjuntos de dados com esses dispositivos seja mais complexa, devido a fatores como iluminação, plano de fundo, número de câmeras utilizadas, e posicionamento do intérprete frente à câmera (Mekala et al., 2011), esses equipamentos possuem como vantagem a praticidade aos usuários, pois muitos deles já fazem parte do cotidiano da maioria das pessoas, como o caso dos *smartphones*. Baseado nisso, uma solução desenvolvida com o uso desses sensores possui maior chance de adoção em larga escala e, conseqüentemente, de atenuar o problema de acessibilidade que atinge a comunidade de deficientes auditivos.

Além da definição do sensor, é importante considerar a forma como os sinais são capturados, sendo necessário incluir todos os parâmetros linguísticos que os compõem, conforme discutido na Seção 1.1, possibilitando o reconhecimento de sinais com características similares e evitando ambigüidades. Muitas pesquisas anteriores da área focam exclusivamente nas mãos do intérprete, limitando a abordagem a um número reduzido de sinais.

1.3 Técnicas para o reconhecimento de sinais

Considerando os trabalhos que utilizam de sensores RGB, as redes neurais convolucionais, também chamadas de CNNs (*Convolutional Neural Networks* em inglês), têm apresentado resultados estado da arte no reconhecimento de sinais, visto que estes modelos são capazes de generalizar a extração de características para inúmeras classes distintas, sendo um importante aspecto para este tipo de tarefa, onde existem milhares de palavras que formam o vocabulário

a ser reconhecido (de Castro et al., 2019). Diferentes arquiteturas têm sido empregadas com o objetivo de extrair informação de todos os parâmetros linguísticos de maneira eficiente, considerando a dimensão temporal e espacial, como é o caso das redes neurais recorrentes (Seção 2.2), CNNs 3D (Seção 2.3.8), CNNs 2+1D (Seção 2.3.9), dentre outras.

Porém, assim como outras abordagens baseadas em aprendizado profundo, as CNNs dependem de uma grande quantidade de dados para serem treinadas, o que gera uma barreira de entrada para a área de reconhecimento de sinais, visto que as bases de dados dependem de muito recurso humano para sua criação. Além disso, o custo computacional deve ser o menor possível, permitindo que o sistema auxilie na comunicação em tempo real entre indivíduos ou seja empregado em dispositivos de borda como os *smartphones*, viabilizando a sua escalabilidade.

Como forma de facilitar a convergência e melhorar os resultados e a eficiência dos modelos, diversas técnicas de pré-processamento têm sido aplicadas para o reconhecimento de sinais (Seção 3.2). Dentre elas, destaca-se o uso de detectores de objetos (Seção 2.4), pela sua capacidade em localizar partes específicas da imagem de maneira satisfatória e com baixo custo computacional, podendo ser empregados na detecção das dicas visuais (termo utilizado na literatura para se referir às mãos e ao rosto do intérprete). Apesar disso, utilizar apenas esses segmentos de imagem na classificação do sinal negligencia os parâmetros linguísticos de movimento e de localização das mãos, prejudicando os resultados.

Assim sendo, a principal motivação do trabalho é propor uma nova técnica de pré-processamento baseada em detecção de objetos que permita que as CNNs foquem nas partes mais importantes da imagem ao passo que evita a perda da informação de localização e movimento das mãos. Para isso, ao invés de utilizar características pouco explicáveis obtidas a partir de modelos de aprendizado de máquina, uma série de atributos referenciais foram calculados para compor a solução proposta. Dessa forma, o conhecimento a respeito dos parâmetros linguísticos foi fundamental para auxiliar no processo de criação de novos atributos, também conhecido como *feature engineering* (Géron, 2017), em que as características resultantes são geralmente chamadas de *handcrafted* (feitas à mão) por não serem criadas automaticamente por modelos.

Por fim, este trabalho foca no reconhecimento isolado de sinais, também conhecido como reconhecimento a nível de palavras, onde o conjunto de dados é composto por vídeos de pequena duração e cada vídeo possui apenas um sinal. As diferenças entre as áreas de reconhecimento isolado e contínuo de sinais são destacadas no Capítulo 3.

1.4 Objetivos

O objetivo geral deste trabalho é explorar os parâmetros linguísticos que formam os sinais a fim de ponderar a sua importância e criar métodos simplificados que possam, com o menor custo computacional, garantir o reconhecimento dos sinais executados. Dentre as investigações, foram consideradas diferentes arquiteturas de CNNs para extração de características e detectores de objetos para o pré-processamento dos vídeos. Além disso, um novo método foi proposto, baseando-se em atributos referenciais extraídos dos centroides das caixas delimitadoras dos detectores de objetos a fim de representar a informação dos parâmetros de movimento e localização das mãos. As principais questões levantadas pela pesquisa são:

1. É possível obter êxito na extração de dicas visuais com baixo custo computacional?
2. É possível representar o ponto de contato e o movimento do sinal através de um vetor de atributos *handcrafted*?

O primeiro questionamento é inspirado pelas novas arquiteturas de detecção de objetos, capazes de atingir bons resultados enquanto demandam de cada vez menos recursos computacionais, podendo ser empregadas para reduzir a complexidade da etapa de pré-processamento dos sistemas que necessitam das dicas visuais como informação de entrada.

O segundo questionamento, por sua vez, gira em torno da hipótese de que é possível substituir o plano de fundo e o próprio corpo do intérprete (utilizados para representar a localização e movimento das mãos) por um vetor de características referenciais, conforme discutido em detalhes na Seção 4.1.4.2, reduzindo a quantidade de informação a ser processada pelo sistema. Dito isso, as contribuições pretendidas por essa pesquisa foram:

- I Adaptar, treinar e comparar diferentes arquiteturas de detecção de objetos para extração de dicas visuais (mãos e rosto);
- II Analisar o uso de um vetor de atributos referenciais para representação do ponto de contato e movimento do sinal.

1.5 Organização da dissertação

De modo geral, o restante deste trabalho está organizado da seguinte forma: o Capítulo 2 discorre sobre a fundamentação teórica das principais técnicas de aprendizado de máquina que

são utilizadas em pesquisas recentes na área de reconhecimento de sinais, destacando os métodos mais comuns e desafios existentes no treinamento dos modelos. O Capítulo 3 explora os principais trabalhos da literatura, considerando diferentes arquiteturas de modelos, técnicas de pré-processamento e bases de dados. O Capítulo 4 detalha o proposto neste trabalho, o desenvolvimento do projeto e a metodologia utilizada para os experimentos realizados. O Capítulo 5 exhibe os resultados obtidos pela pesquisa através dos experimentos. Por fim, o Capítulo 6 apresenta algumas conclusões a respeito da dissertação, bem como apresenta sugestões de melhorias a serem realizadas por trabalhos futuros.

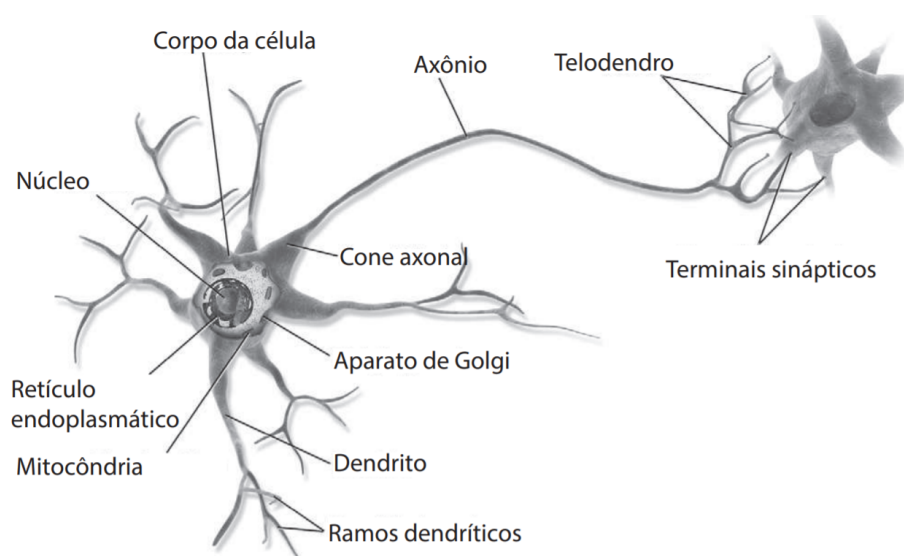
2 Fundamentação Teórica

As técnicas de aprendizado de máquina têm sido amplamente adotadas em diversas áreas do conhecimento, devido à sua capacidade de adaptação a diferentes domínios e níveis de complexidade. Dentre elas, destacam-se as redes neurais artificiais e suas derivações, integrando diferentes arquiteturas. Baseado nisso, este capítulo tem por objetivo introduzir os principais aspectos teóricos relativos às redes neurais artificiais (Seção 2.1), como a sua arquitetura base, funções de ativação, processo de treinamento e avaliação, entre outros detalhes. Além disso, demais modelos e arquiteturas serão abordadas, por serem relevantes para a área de reconhecimento de sinais, como as redes neurais recorrentes (Seção 2.2), as CNNs (Seção 2.3), os detectores de objetos (Seção 2.4) e os modelos de inferência de pose corporal (Seção 2.5).

2.1 Redes neurais artificiais

As redes neurais artificiais são modelos matemáticos bioinspirados no sistema nervoso central dos animais. O neurônio biológico, sendo a menor unidade que compõe as redes neurais, possui uma série de conexões para entrada (dendritos) e uma conexão para saída (axônio) de impulsos eletroquímicos que representam a transmissão de informação, conforme ilustrado na Figura 2.1. A troca de informação ocorre por meio dos neurotransmissores através das conexões sinápticas, e é condicionada pelas excitações e inibições recebidas por células vizinhas (Gurney, 1997), gerando um efeito em cadeia.

Figura 2.1: Esquema simplificado de um neurônio biológico.



Fonte: Geron (2017).

O neurônio artificial, inicialmente proposto por [McCulloch e Pitts \(1943\)](#) e aprimorado por [Rosenblatt \(1958\)](#) em um modelo chamado Perceptron, segue uma lógica estrutural bastante similar ao biológico, representando a saída z do neurônio através de uma soma ponderada entre as informações de entrada $(x_1, x_2, \dots, x_n)$ e os pesos $(w_1, w_2, \dots, w_n)$ (também chamados de parâmetros) que refletem a importância das entradas, conforme representado na Equação 2.1 ([Géron, 2017](#)):

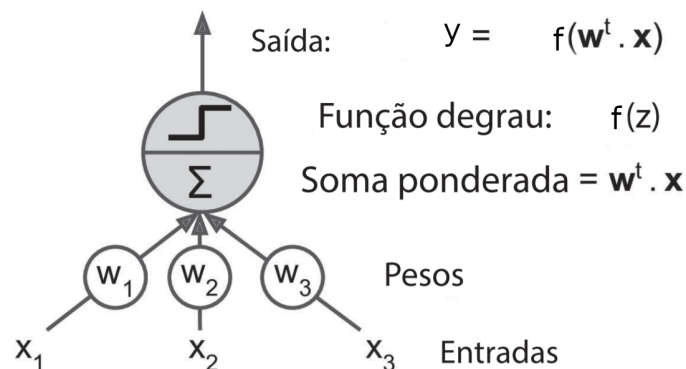
$$z = w_1x_1 + w_2x_2 + \dots + w_nx_n = \mathbf{w}^T \cdot \mathbf{x}. \quad (2.1)$$

O resultado dessa soma é ainda submetido a uma função de ativação binária $f(z)$ (conhecida como função degrau ou *step function*, em inglês) que determina, baseado em um limiar, se o neurônio transmitirá uma informação excitatória ($\hat{y} = 1$) ou inibitória ($\hat{y} = 0$), conforme descrito na Equação 2.2 ([Rosenblatt, 1958](#)):

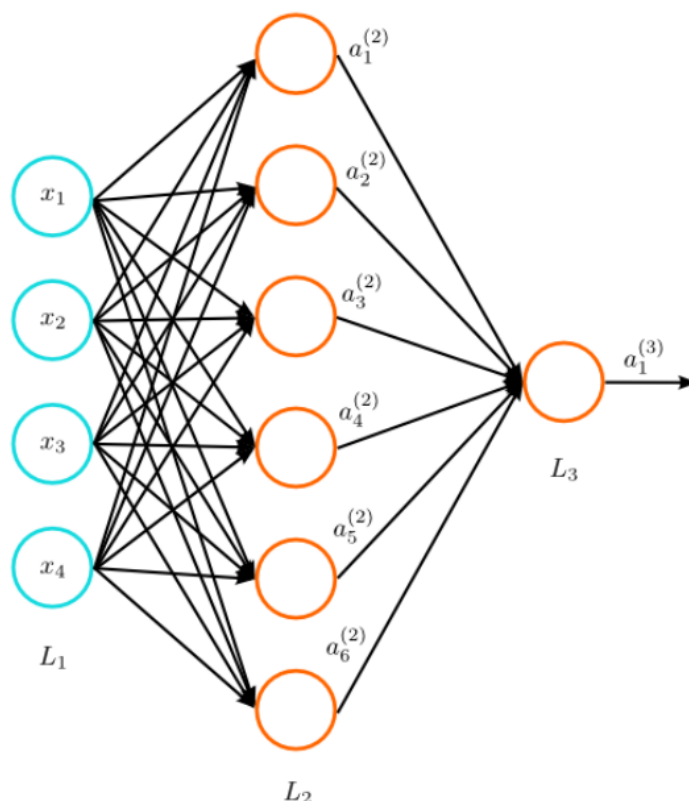
$$f(z) = \begin{cases} 1, & \text{se } z \geq 0 \\ 0, & \text{caso contrário.} \end{cases} \quad (2.2)$$

O fluxo completo, por sua vez, é ilustrado na Figura 2.2. Embora o Perceptron seja capaz de resolver problemas linearmente separáveis, existe uma limitação do modelo quando as classes não podem ser distinguidas através de um hiperplano. Portanto, foi proposto o modelo de rede neural denominado *Multi Layer Perceptron* (MLP) ([Werbos, 1974](#)), constituído por três camadas de neurônios do tipo Perceptron e funções de ativação não lineares, ilustrado pela Figura 2.3.

Figura 2.2: Modelo de um neurônio artificial do tipo Perceptron.



Fonte: Adaptado de [Géron \(2017\)](#).

Figura 2.3: Modelo do tipo *Multi Layer Perceptron* (MLP)

Fonte: Magalhães (2018).

No MLP, observa-se que além da camada de entrada (L_1) e a de saída (L_3) existe uma camada intermediária (L_2), chamada de camada oculta. Esse tipo de rede neural também é conhecida como *feedforward*, pelo fato da informação fluir da camada de entrada até a camada de saída sem conexões de retroalimentação (Goodfellow et al., 2016). Com isso, criou-se uma alternativa mais robusta do que o uso de um simples Perceptron e que consegue lidar com problemas não linearmente separáveis. Não obstante, também é notável que a mesma arquitetura pode ser adotada para a criação de modelos com mais camadas ocultas, sendo conhecidos como modelos de aprendizado profundo ou *deep learning* (Glorot e Bengio, 2010), capazes de resolver problemas mais complexos como o reconhecimento de padrões em imagens e o processamento de linguagem natural.

O treinamento deste tipo de modelo, por sua vez, é caracterizado por um processo iterativo de atualização de parâmetros baseado no erro calculado por uma função de custo, repetido até que o algoritmo convirja para o ponto ótimo global, onde o menor custo possível é obtido. Como essa convergência dificilmente ocorre, dado que a complexidade do problema geralmente é alta, se estabelece um critério de parada baseado no número de épocas, onde uma época é definida

pela passagem de todas as amostras de dados do conjunto pela rede neural.

Dito isso, as seções subsequentes se aprofundam nos detalhes que permeiam a composição e o processo de treinamento das redes neurais profundas, como as principais funções de ativação (discutidas na Seção 2.1.1), a definição da função de custo (Seção 2.1.2), as funções de otimização (Seção 2.1.3), o algoritmo de retropropagação dos erros (Seção 2.1.4), o efeito de sobreajuste e subajuste do modelo (Seção 2.1.5), as técnicas de regularização (Seção 2.1.6), e as métricas comuns de avaliação de resultados (Seção 2.1.7).

2.1.1 Funções de ativação

Um dos componentes de maior importância no modelo de redes neurais artificiais é a função utilizada para ativação dos neurônios, determinando como a informação será passada adiante pela rede e podendo impactar de forma significativa na performance para resolução de problemas. Essa seção explora os detalhes das funções de ativação mais comumente utilizadas, sendo elas a função sigmoide (2.1.1.1), tangente hiperbólica (2.1.1.2), *ReLU* (2.1.1.3), *leaky ReLU* (2.1.1.4) e *softmax* (2.1.1.5).

2.1.1.1 Função sigmoide

A função sigmoide, também chamada de logística, é amplamente utilizada devido à sua simplicidade e também não linearidade. Sua definição matemática pode ser expressa na Equação 2.3 (Géron, 2017):

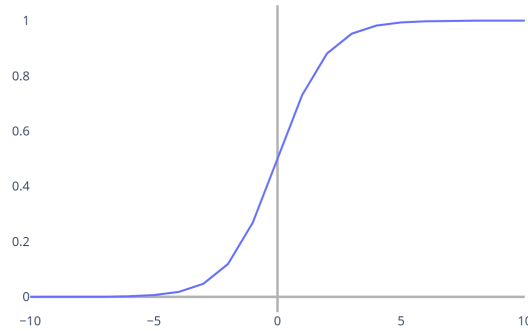
$$\sigma(z) = \frac{1}{1 + e^{-z}}, \quad (2.3)$$

em que e representa uma função exponencial e z é o valor resultante da soma ponderada exibida na Equação 2.1. Trata-se de uma função monótona estritamente crescente, cujo o domínio se limita entre 0 e 1, seguindo um formato de “S” que representa o intervalo de valores que a mesma pode assumir, conforme ilustrado na Figura 2.4.

A função degrau, exibida na Seção 2.1, contém apenas segmentos planos (Géron, 2017), impossibilitando o uso de algoritmos de otimização baseados em gradiente (discutidos na Seção 2.1.3) e, portanto, a função logística se torna uma opção mais viável por possuir derivadas diferentes de zero e, dessa forma, permitir a estimativa de gradientes.

Contudo, esta função também possui algumas limitações que atrapalham sua adoção em modelos mais complexos, como em redes neurais multicamadas, entre elas podemos citar: sa-

Figura 2.4: Representação da função sigmoide.



Fonte: Autor.

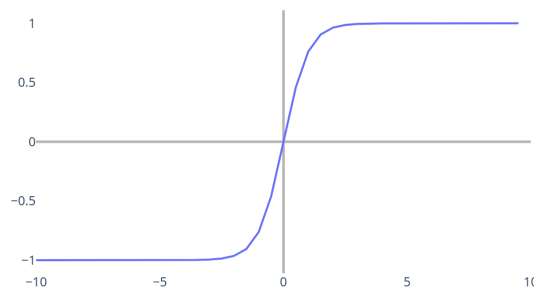
turação do gradiente, convergência lenta, e não centralização em zero (Nwankpa et al., 2021). A saturação dos gradientes, por exemplo, ocorre na maior parte do domínio da função, saturando para um valor alto quando as entradas são muito positivas e para um valor baixo quando as entradas são muito negativas, dificultando o aprendizado dos modelos (Goodfellow et al., 2016).

2.1.1.2 Função tangente hiperbólica

A função tangente hiperbólica é vista como uma alternativa à sigmoide, uma vez que ela é centrada em zero e possui melhor simetria, conforme ilustrado na Figura 2.5. Os valores dessa função estão no intervalo de -1 a 1, sendo gerados conforme exibido pelo Equação 2.4 (Zamanlooy e Mirhassani, 2013):

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \quad (2.4)$$

Figura 2.5: Representação da função tangente hiperbólica.



Fonte: Autor.

Embora essa função seja amplamente utilizada, inclusive nas redes neurais recorrentes (conforme discutido na Seção 2.2), sua formulação também permite que o problema de saturação

dos gradientes continue a existir, ainda que em menor escala.

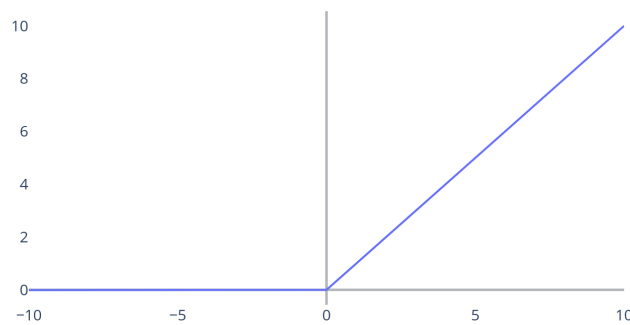
2.1.1.3 Função ReLU

A ReLU (*Rectified Linear Unit*) (Géron, 2017) é uma das funções mais utilizadas atualmente por não ser linear, resolver o problema de saturação e possuir menor custo computacional, dada a simplicidade em seu cálculo, conforme definido pela Equação 2.5 (Magalhães, 2018):

$$f(z) = \max(0, z) = \begin{cases} 0, & \text{para } z < 0 \\ z, & \text{para } z \geq 0. \end{cases} \quad (2.5)$$

Conforme ilustrado na Figura 2.6, a ReLU é composta por uma etapa não linear, zerando todos os valores negativos, e uma parte linear, mantendo os valores positivos. Com isso, apenas alguns neurônios são ativados por vez, gerando um ganho de eficiência computacional (Nwankpa et al., 2021).

Figura 2.6: Representação da função ReLu.



Fonte: Autor.

Ainda assim, a ReLU tem uma limitação significativa relacionada ao problema das derivadas nulas para todo conjunto $\mathbb{R}^-$ (Magalhães, 2018), fazendo com que os gradientes desapareçam e, com isso, anulando os neurônios, ou seja, impedindo que seus valores sejam atualizados, o que dificulta o treinamento (Nwankpa et al., 2021).

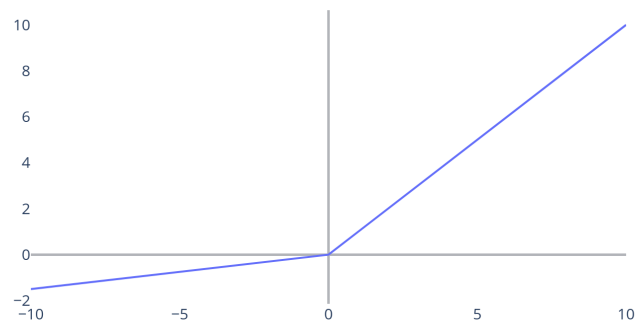
2.1.1.4 Função Leaky ReLU

A função *Leaky ReLU* (Xu et al., 2015) foi criada como uma alternativa ao problema da morte dos neurônios observado na função ReLU, fazendo com que o valor do gradiente não seja anulado facilmente, conforme representado por sua definição matemática na Equação 2.6 (Magalhães, 2018):

$$f(z) = \max(\alpha z, z) = \begin{cases} \alpha z, & \text{para } z < 0 \\ z, & \text{para } z \geq 0. \end{cases} \quad (2.6)$$

Assim, ao invés de zerar os valores negativos, a constante α gera um componente linear muito pequeno, impedindo que neurônios sejam inativados e parem de ser atualizados. A Figura 2.7 ilustra a diferença entre a função *Leaky ReLU* e a *ReLU*.

Figura 2.7: Representação da função *Leaky ReLU*.



Fonte: Autor.

2.1.1.5 Função Softmax

A função Softmax é muito utilizada como ativação para os neurônios da camada de saída das redes neurais em problemas com múltiplas classes, fazendo com que cada neurônio desta camada passe a representar uma classe distinta.

Essa função também se limita ao intervalo de 0 a 1, porém, a soma de todos os valores do vetor de saída é sempre igual a 1, fazendo com que cada neurônio seja interpretado como a probabilidade de ativação para determinada classe.

De maneira formal, a função Softmax pode ser representada por um vetor $\mathbf{p} = [p_1, p_2, \dots, p_k]$, sendo que k é o número de classes e o termo p_i é obtido através da Equação 2.7 (Magalhães, 2018):

$$p(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}, \quad (2.7)$$

em que $\mathbf{z}$ representa o vetor com os valores das somas ponderadas de cada um dos neurônios da camada de saída. Por fim, a classe predita pela rede neural é determinada pela maior probabilidade do vetor $\mathbf{p}$, conforme exibido na Equação 2.8 (Agarap, 2018):

$$\hat{y} = \arg \max_{i \in 1 \dots k} p_i. \quad (2.8)$$

2.1.2 Função de custo

A função de custo (também conhecida como *loss function* em inglês), tem o papel de calcular a magnitude dos erros gerados pelo modelo. O processo de iteração de treinamento gira em torno do valor produzido por essa função, buscando minimizá-la, convergindo para o ponto onde o menor erro possível é gerado, chamado de mínimo global. Existem inúmeras formas de se avaliar o erro, sendo que a mais comum delas para problemas de classificação é a entropia cruzada, também chamada de *cross entropy* (Murphy, 2012).

A entropia cruzada é um cálculo proveniente da teoria da informação que mensura a quantidade de informação necessária para representar um evento baseado em sua incerteza, em que eventos mais comuns precisam de menos informação (*bits*) do que eventos menos comuns (Géron, 2017). Dado dois conjuntos de distribuições probabilísticas q e p , a entropia cruzada é definida pela Equação 2.9 (Géron, 2017):

$$H(p, q) = - \sum_x p(x) \log(q(x)), \quad (2.9)$$

em que x representa os possíveis estados que o evento em questão pode assumir. No contexto do aprendizado de máquina, a fórmula da entropia utilizada como função de custo difere de modo a calcular a distribuição probabilística proveniente da camada de saída, como as probabilidades obtidas pela função *Softmax* (Seção 2.1.1.5), em relação à distribuição probabilística da classe alvo, conforme definido na Equação 2.10 (Géron, 2017):

$$j(\boldsymbol{\theta}) = - \frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(p_k^{(i)}), \quad (2.10)$$

em que $y_k^{(i)}$ é a probabilidade alvo da i -ésima instância pertencer à classe k , m é o número total de instâncias de treinamento e $\boldsymbol{\theta}$ representa o vetor de parâmetros do modelo. Desta forma, o modelo apresenta um cenário de convergência quando a distribuição probabilística calculada pela função de ativação é alinhada com os valores esperados pelas classes das instâncias, resultando em valores próximos a 0 de custo para o vetor de parâmetros $\boldsymbol{\theta}$.

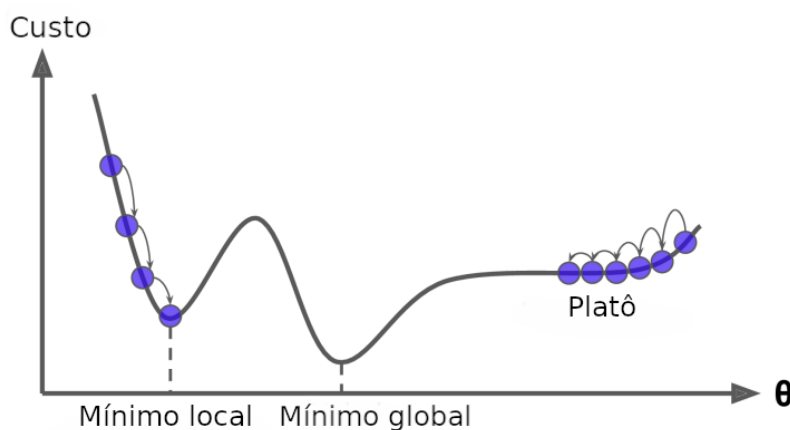
2.1.3 Funções de otimização

A atualização dos parâmetros (pesos, vieses) de uma rede neural visa reduzir os erros calculados pela função de custo, ajustando a superfície de decisão do modelo aos dados sendo observados. Essa tarefa pode ser entendida como um problema de otimização complexo, dado que modelos de aprendizado profundo podem chegar facilmente a milhões de parâmetros, gerando uma função de otimização de custo não convexa com inúmeros desafios a serem transpostos. Pensando nisso, diferentes estratégias foram desenvolvidas para fazer com que tais funções sejam otimizadas e converjam para o menor erro possível de maneira eficiente. A Seção 2.1.3.1 levanta os principais problemas na otimização de redes neurais profundas, enquanto nas seções subsequentes (Seções 2.1.3.2 a 2.1.3.8) são discutidos alguns dos algoritmos de otimização utilizados para essa tarefa.

2.1.3.1 Problemas comuns na otimização de redes neurais

Conforme discutido anteriormente, a função a ser otimizada pelas redes neurais é não convexa e, por isso, possui diferentes problemáticas que devem ser consideradas em seu treinamento. Primeiramente, existe o problema dos mínimos locais, sendo comumente referidos como uma armadilha para os algoritmos de otimização. Um mínimo local é caracterizado por um ponto onde os erros gerados são menores que os da vizinhança, tendenciando o algoritmo de otimização a estacionar. Geralmente, a função conta com inúmeros mínimos locais (Goodfellow et al., 2016), fazendo com que a descoberta do mínimo global seja dificultada. A Figura 2.8 ilustra o problema supracitado.

Figura 2.8: Otimização de uma função de custo não convexa.



Fonte: Adaptado de Géron (2017).

Além disso, a Figura 2.8 também exibe outro problema comum dessas funções, chamados de platôs ou *plateaus* em inglês. Assim como o nome sugere, essas regiões são caracterizadas por um custo muito similar, independente da mudança paramétrica, fazendo com que os gradientes sejam muito parecidos e dificultando a otimização.

2.1.3.2 Gradiente descendente

O gradiente descendente é uma técnica genérica de otimização capaz de encontrar soluções ótimas em diversos problemas distintos (Géron, 2017). Ele funciona de forma iterativa, calculando os gradientes locais gerados pela função de custo $j(\boldsymbol{\theta})$ e atualizando o valor de $\boldsymbol{\theta}$ baseado na magnitude do gradiente e na direção indicada pelas derivadas. A derivada da função $j(\boldsymbol{\theta})$, denotada por $j'(\boldsymbol{\theta})$, resulta na inclinação de $j(\boldsymbol{\theta})$ no ponto $\boldsymbol{\theta}$ (Goodfellow et al., 2016). Com isso, é possível verificar como uma mudança no valor de entrada da função (pesos) gera uma mudança correspondente no valor de saída, podendo ser usada para minimizar a função movendo os valores de $\boldsymbol{\theta}$ pequenos passos no sinal oposto à derivada.

Para funções com múltiplas entradas, como normalmente é o caso das redes neurais, as derivadas parciais $\frac{\partial}{\partial \theta_i} j(\boldsymbol{\theta})$ são levadas em consideração, mensurando o quanto j muda com apenas a alteração da entrada θ_i no ponto $\boldsymbol{\theta}$ (Goodfellow et al., 2016). O gradiente passa a ser escrito como um vetor que generaliza o valor de todas as derivadas parciais da seguinte forma: $\nabla_{\boldsymbol{\theta}} j(\boldsymbol{\theta})$. Finalmente, o processo de atualização dos parâmetros é dado pela Equação 2.11 (Goodfellow et al., 2016):

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \varepsilon \nabla_{\boldsymbol{\theta}} j(\boldsymbol{\theta}). \quad (2.11)$$

Com isso, o valor de entrada é modificado baseado na multiplicação das derivadas parciais por um parâmetro ε chamado de taxa de aprendizado. A Figura 2.8 ilustra esse processo, onde cada círculo representa um ponto com diferentes valores do vetor de parâmetros do modelo (considerando uma inicialização randômica, por exemplo) e as setas representam a evolução dos parâmetros em cada iteração, baseado nos gradientes da função de custo. No exemplo citado é possível verificar que a movimentação pelo espaço de parâmetros é mais lenta no platô do que no ponto de inclinação acentuada, conforme discutido na Seção 2.1.3.4.

2.1.3.3 Gradiente descendente estocástico

O principal problema do algoritmo de gradiente descendente é o fato de que os valores são

computados baseado em toda a base de treinamento, sendo extremamente lento e custoso para bases de dados grandes (Géron, 2017), o que costuma ser o caso em problemas de aprendizado profundo. Como forma de transpor esses limites, é proposto o uso do gradiente descendente estocástico ou SGD (*stochastic gradient descent* em inglês), que a cada iteração utiliza uma parcela aleatória (por isso o nome de estocástico) da base de dados para a atualização dos parâmetros, gerando uma aproximação do método original, conforme definido na Equação 2.12 (Goodfellow et al., 2016):

$$g = \frac{1}{m} \nabla_{\theta} \sum_{i=1}^m j(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)}), \quad (2.12)$$

onde $j(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)})$ é a função de custo considerando a entrada $\mathbf{x}^{(i)}$ e a saída esperada $\mathbf{y}^{(i)}$ e m representa a quantidade de instâncias amostradas do conjunto. Embora o SGD não tenha garantia de convergência e seja sensível aos valores iniciais do modelo, ele é o algoritmo de otimização mais utilizado para o contexto de *deep learning*, dada sua viabilidade computacional para conjuntos de dados de larga escala (Goodfellow et al., 2016).

2.1.3.4 Momentum

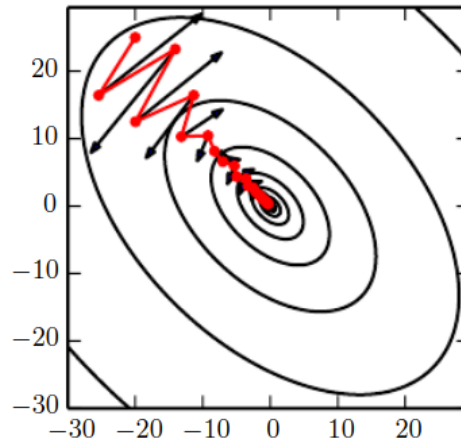
Ainda que o SGD seja mais viável computacionalmente, o treinamento de redes neurais profundas é um processo que pode levar dias ou até semanas, devido ao vasto espaço de busca gerado pelo elevado número de parâmetros. Por conta disso, o Momentum (Polyak, 1964) é um método muito empregado para acelerar o treinamento dos algoritmos de otimização, principalmente quando estes enfrentam situações de curvatura acentuada, gradientes pequenos (porém consistentes) ou gradientes ruidosos (Goodfellow et al., 2016).

O termo se origina da física e diz respeito à tendência do aumento da velocidade de uma partícula que está sob uma força. Da mesma forma, o algoritmo considera os valores das iterações passadas do gradiente para reforçar o movimento, conseguindo passar mais rápido por platôs e também ignorar com facilidade certos mínimos locais. A Figura 2.9 ilustra o efeito do uso do Momentum no SGD (linha vermelha).

Outra vantagem possível de se observar na Figura 2.9 é que o Momentum reduz a variação gerada pelo SGD, comum pelo fato de apenas uma porção (*batch*) da base de dados ser utilizada no treinamento a cada iteração. Com isso, ao considerar a média móvel dos gradientes passados, a convergência se dá de maneira mais linear.

Um novo parâmetro $\alpha \in [0, 1)$ também é introduzido, determinando a rapidez com que as

Figura 2.9: Convergência da função de custo com (em vermelho) e sem o uso do Momentum no algoritmo de otimização SGD.



Fonte: Goodfellow et al. (2016).

contribuições dos gradientes anteriores irão decair, atuando como um mecanismo de fricção para evitar que os valores se tornem muito grandes para a atualização dos pesos. A definição matemática do Momentum aplicado ao SGD é dada pelas Equações 2.13 e 2.14 (Goodfellow et al., 2016):

$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^m j(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)}) \right), \quad (2.13)$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathbf{v}. \quad (2.14)$$

A velocidade é representada pela variável $\mathbf{v}$. Quanto maior o tamanho de α em relação a ϵ , mais os gradientes anteriores irão afetar a direção do movimento.

2.1.3.5 Programação da taxa de aprendizado

A taxa de aprendizado (ϵ), introduzida na Equação 2.11, é um dos hiperparâmetros mais importantes para o algoritmo de otimização baseado em gradientes, determinando o tamanho dos passos que serão dados em cada iteração. Quanto maior a taxa de aprendizado, mais abrupta é a atualização dos parâmetros, o que garante uma exploração mais rápida pelo espaço, mas pode facilitar a perda do ponto ótimo. Por outro lado, utilizar uma taxa de aprendizado pequena pode assegurar que o mínimo global não escape, porém é possível que esse ponto nunca seja atingido, dado o número de iterações necessárias para convergência do modelo devido às alterações sutis nos parâmetros.

Por conta da importância desse hiperparâmetro, diversas técnicas têm surgido visando alcançar um balanceamento entre a velocidade de convergência e a estabilidade na atualização dos parâmetros, entre elas a programação da taxa de aprendizado (chamada de *learning rate schedule* em inglês) e os algoritmos de taxa de aprendizado adaptativa (Seções 2.1.3.6 a 2.1.3.8).

A programação da taxa de aprendizado é responsável por controlar a forma como o valor de ε irá se comportar ao longo das iterações. É comum que esse valor diminua de forma linear, fazendo com que a atualização dos pesos seja maior no início da otimização e menor no final, quando o ponto mínimo se encontra mais próximo. A queda linear da taxa de aprendizado pode ser definida pela Equação 2.15 (Goodfellow et al., 2016):

$$\varepsilon_i = (1 - \alpha)\varepsilon_0 + \alpha\varepsilon_\tau, \quad (2.15)$$

em que τ é o número de iterações em que ε irá ser reduzido até que seja mantido constante, i é a iteração atual, ε_0 é a taxa de aprendizado inicial e $\alpha = \frac{i}{\tau}$. Usualmente, o valor de τ é definido como o número de iterações necessárias para se completar algumas épocas pelo conjunto de treinamento, enquanto ε_τ é aproximadamente 1% do valor de ε_0 (Goodfellow et al., 2016). Ainda assim, não existe um consenso de qual o melhor valor para ε_0 , sendo que a escolha geralmente se baseia na observação das curvas de aprendizado do modelo através de treinamentos experimentais.

Além disso, o valor de ε_0 também pode ser guiado dependendo da arquitetura da rede neural e do objetivo pretendido, como é possível observar no caso do processo de ajuste fino dos parâmetros (discutido na Seção 2.3) em CNNs que foram treinadas em milhões de imagens, por exemplo, onde é comum utilizar uma taxa de aprendizado inicial muito pequena, entre $1e^{-5}$ e $1e^{-6}$.

Por fim, diversos outros métodos foram criados para reduzir a taxa de aprendizado ao longo das iterações, como o decaimento em etapas (Ge et al., 2019), triangular (Goyal et al., 2017) e cíclico (Smith, 2017). O decaimento em etapas corta a taxa de aprendizado por um fator constante a cada n número de épocas, reduzindo o valor geometricamente em um formato de degrau. O cronograma triangular, por sua vez, é baseado na ideia de preaquecer o treinamento, iniciando o processo com uma taxa de aprendizado pequena e aumentando seu valor gradativamente pelas épocas até atingir um limite l , voltando a reduzir gradualmente após o atingimento do limite. Com essa abordagem, os autores obtiveram sucesso no treinamento distribuído de um modelo complexo de classificação de imagens, evitando que os parâmetros se degradem no início do

treinamento, graças a taxa de aprendizado pequena, e garantindo que o espaço de busca seja devidamente explorado nas demais iterações. Finalmente, o decaimento cíclico faz com que os valores de ε aumentem e diminuam baseado em ciclos de iterações, sendo que cada ciclo possui um limiar mínimo e máximo para a taxa de aprendizado. Os autores demonstram que essa é uma técnica eficiente para escapar de mínimos locais ao longo da convergência do modelo.

2.1.3.6 AdaGrad

Outra abordagem comum para lidar com a taxa de aprendizado são os algoritmos adaptativos, que consideram um ε para cada parâmetro e adaptam seu valor ao longo do curso do treinamento (Goodfellow et al., 2016). Um dos primeiros algoritmos nesse quesito é o AdaGrad (Duchi et al., 2011), responsável por reduzir rapidamente a taxa de aprendizado para parâmetros com derivada parcial maior, enquanto reduz menos os de menor derivada parcial. Para isso, é calculado o valor quadrático acumulado dos gradientes, representado pela variável $\mathbf{r}$, conforme visto na Equação 2.16 (Goodfellow et al., 2016):

$$\mathbf{r} \leftarrow \mathbf{r} + \mathbf{g} \odot \mathbf{g}, \quad (2.16)$$

em que o valor de $\mathbf{g}$ são os gradientes da iteração, conforme visto na Equação 2.12.

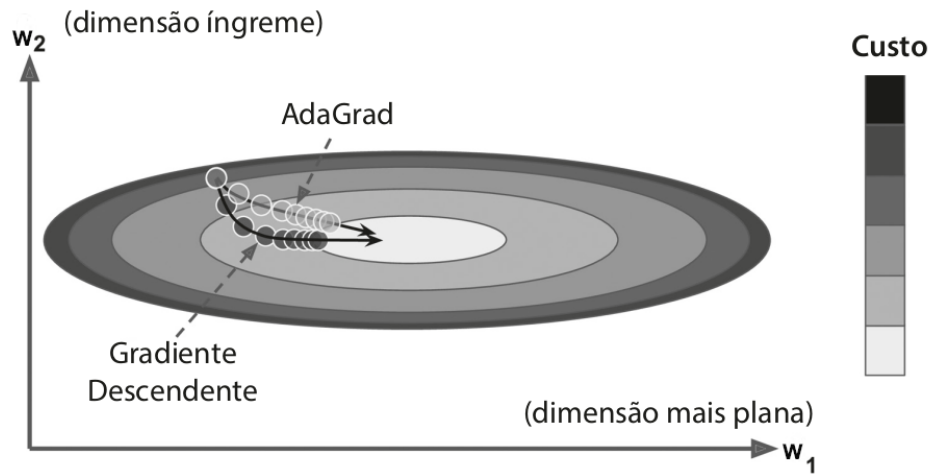
Por fim, para cada iteração da atualização dos parâmetros, os valores do gradiente são escalonados pela divisão da raiz quadrada (aplicada por elemento) da soma de $\mathbf{r}$, somados a uma pequena constante δ usada para estabilizar a divisão, conforme exibido pela Equação 2.17 (Goodfellow et al., 2016):

$$\Delta \theta \leftarrow -\frac{\varepsilon}{\delta + \sqrt{\mathbf{r}}} \odot \mathbf{g}. \quad (2.17)$$

Com isso, a convergência do modelo ocorre mais rapidamente, principalmente para pontos mais íngremes no gradiente, conforme ilustrado na Figura 2.10. Além disso, outra vantagem das técnicas adaptativas é a automatização na escolha e ajuste do valor de ε , reduzindo o esforço manual e a necessidade de definir cronogramas, conforme visto na Seção 2.1.3.5.

Embora o AdaGrad funcione bem para problemas convexos, ele não é recomendado para redes neurais, uma vez que o escalonamento acaba reduzindo muito o valor da taxa de aprendizado, fazendo com que o algoritmo pare de otimizar antes de encontrar o ponto ótimo (Géron, 2017).

Figura 2.10: Ilustração do efeito na convergência causado pelo algoritmo AdaGrad.



Fonte: [Géron \(2017\)](#).

2.1.3.7 RMSProp

O algoritmo RMSProp ([Hinton et al., 2012](#)) é uma modificação do AdaGrad mais adequada para uso em funções não convexas, como é o caso das redes neurais artificiais. Sua diferença se encontra na forma de acumular os gradientes, gerando uma média móvel exponencialmente ponderada para o decaimento da taxa de aprendizado, que desconsidera alguns valores passados do gradiente baseado em um novo parâmetro ρ , conforme definido pela Equação 2.18 ([Goodfellow et al., 2016](#)):

$$\mathbf{r} \leftarrow \rho \mathbf{r} + (1 - \rho) \mathbf{g} \odot \mathbf{g}. \quad (2.18)$$

O passo seguinte ao cálculo dos valores acumulados é o mesmo, definido na Equação 2.16.

2.1.3.8 Adam

Por fim, Adam (derivado de *adaptive moments* em inglês) ([Kingma e Ba, 2015](#)) é outro algoritmo muito utilizado para a adaptação da taxa de aprendizado, inclusive para problemas de *deep learning*, que une os conceitos de RMSProp com o Momentum, controlando a média móvel exponencialmente ponderada de cada um deles através de dois parâmetros ρ_1 e ρ_2 . Esses parâmetros são utilizados em cada iteração para o cálculo da estimativa de primeira ordem (representada pela variável $\mathbf{s}$) e segunda ordem (representada pela variável $\mathbf{r}$) do gradiente, ambas com um valor inicial de $\mathbf{0}$, sendo calculadas conforme definido nas Equações 2.19 e 2.20 ([Goodfellow et al., 2016](#)):

$$\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1) \mathbf{g} \quad (2.19)$$

e

$$\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1 - \rho_2) \mathbf{g} \odot \mathbf{g}. \quad (2.20)$$

Além disso, as estimativas passam por uma correção de viés, vista nas Equações 2.21 e 2.22 (Goodfellow et al., 2016):

$$\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1 - \rho_1} \quad (2.21)$$

e

$$\hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1 - \rho_2}. \quad (2.22)$$

Por fim, a atualização dos parâmetros no algoritmo Adam é realizada conforme a equação 2.23 (Goodfellow et al., 2016):

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \left(-\varepsilon \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}} + \delta}} \right). \quad (2.23)$$

2.1.4 Retropropagação do erro

Uma vez que o custo é calculado, conforme visto na Seção 2.1.2, é preciso atualizar os parâmetros da rede neural, visando convergir para o menor erro possível. Para isso, o algoritmo de retropropagação dos erros (Rumelhart et al., 1986), também chamado de *backpropagation* em inglês, é parte fundamental do processo. A ideia desse algoritmo, no contexto do aprendizado de máquina, é o de calcular os gradientes da função de custo em relação aos parâmetros utilizados de maneira eficiente através do uso da regra de cadeia. O nome do algoritmo se dá pelo fato de computar iterativamente o quanto cada neurônio da rede contribuiu para o erro resultante, da camada de saída até a camada de entrada (Géron, 2017), percorrendo o caminho inverso ao *feedforward*. Para isso, Rumelhart et al. (1986) propõe o cálculo das derivadas parciais do erro E produzido pela saída y em relação ao valor esperado d da rede para cada um de seus neurônios, conforme definido pela Equação 2.24 (Rumelhart et al., 1986), onde a regra de cadeia é aplicada para E em relação à entrada x_j :

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dx_j}. \quad (2.24)$$

Conforme apresentado por [Rumelhart et al. \(1986\)](#), é possível saber como uma mudança na entrada x para uma unidade de saída afetará o erro. Baseado no fato de que as entradas são apenas uma função linear dos pesos das conexões das unidades de nível inferior, é possível calcular também a mudança no erro E da função através da alteração em um peso w_{ji} do neurônio j para o neurônio i , conforme representado na Equação 2.25 ([Rumelhart et al., 1986](#)):

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ji}}. \quad (2.25)$$

Com isso, ao considerar todas as conexões emanando do neurônio i , é possível obter ([Rumelhart et al., 1986](#)):

$$\frac{\partial E}{\partial y_j} = \sum_j \frac{\partial E}{\partial x_j \cdot w_{ji}}. \quad (2.26)$$

Finalmente, uma vez que os gradientes são calculados para todas as camadas através do *backpropagation*, os algoritmos de otimização são empregados para atualização dos parâmetros, conforme detalhado na Seção 2.1.3.

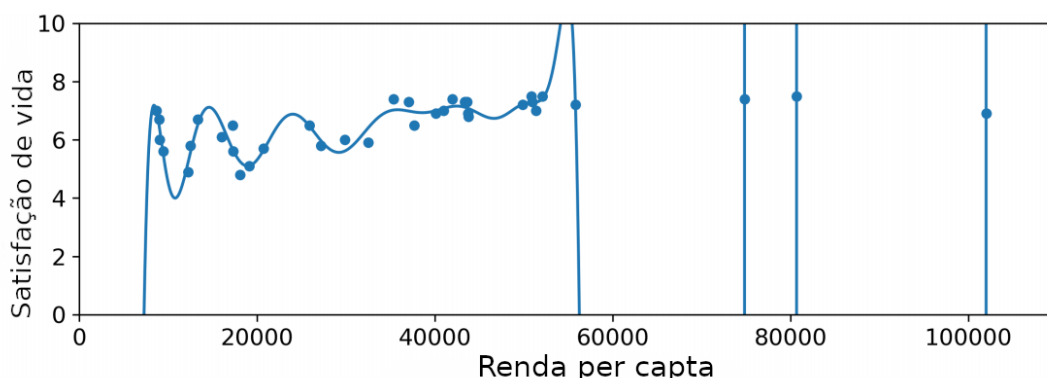
2.1.5 *Overfitting e underfitting*

Dentre os problemas mais comuns no treinamento de modelos de aprendizado de máquina destacam-se os efeitos do sobreajuste, comumente conhecido como *overfitting*, e o subajuste, chamado de *underfitting*, ambos relacionados ao ajuste dos parâmetros do modelo aos dados que estão sendo utilizados para treinamento.

Overfitting é o nome dado ao fenômeno de ajuste excessivo dos parâmetros aos dados, fazendo com que padrões sejam detectados até mesmo em pontos ruidosos e afetando a capacidade de generalização do modelo, visto que a sua superfície de decisão fica enviesada em relação às observações de treinamento. A Figura 2.11 ilustra este processo.

Conforme é possível notar, os parâmetros se ajustam de maneira tal ao dados que os pontos ruidosos presentes após o valor de 60000 são incluídos como parte do padrão, fazendo com que o modelo se torne específico demais aos dados utilizados no treinamento. Como principal consequência desse efeito, é possível notar a degradação dos resultados nas bases de validação e teste, indicando que o modelo pode não se comportar bem com amostras nunca antes vistas,

Figura 2.11: Exemplo da superfície de decisão de um modelo que está altamente ajustado aos dados (*overfitting*).



Fonte: Adaptado de [Géron \(2017\)](#).

dificultando sua adoção em ambiente real.

No contexto do aprendizado profundo, o efeito de *overfitting* é muito comum, dada a complexidade e o número de parâmetros que as redes neurais multicamadas possuem, aumentando a tendência do modelo a se ajustar em excesso. Dito isso, uma das possíveis soluções para reduzir o sobreajuste é a simplificação do modelo, podendo ser através de alterações em sua arquitetura (reduzindo o número de neurônios e camadas, por exemplo), pela escolha de um modelo com menos parâmetros, como uma regressão logística, ou por meio das técnicas de regularização, discutidas na Seção 2.1.6. Além disso, reduzir os ruídos ou aumentar a quantidade de dados da base podem auxiliar na atenuação desse fenômeno ([Géron, 2017](#)).

O *underfitting*, por sua vez, é o efeito contrário ao *overfitting*, observado quando o modelo não consegue capturar os padrões presentes nos dados devido à sua simplicidade em relação ao problema que se está tentando resolver. Esse efeito, entretanto, é menos comum em redes neurais profundas, pois a complexidade do modelo pode ser aumentada facilmente através da adição de mais camadas ocultas e neurônios, gerando uma tendência de sobreajuste maior do que a de subajuste.

2.1.6 Regularização

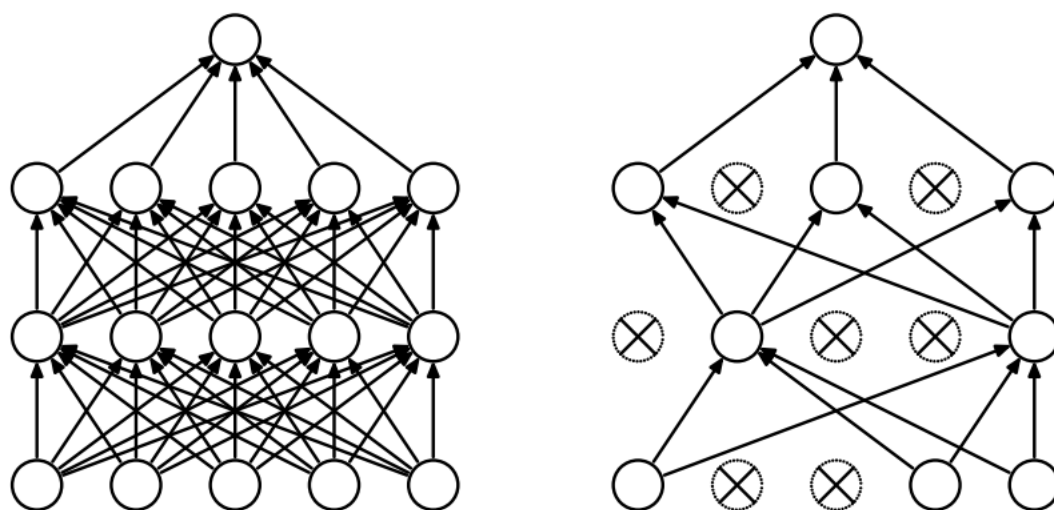
Conforme visto na Seção 2.1.5, uma das formas de atenuar o efeito do *overfitting* é a regularização, que visa principalmente reduzir os graus de liberdade do modelo ([Géron, 2017](#)). Estudos têm mostrado como algumas técnicas computacionalmente simples podem ser empregadas em modelos complexos para gerar ganhos expressivos nos resultados. Dentre elas, podemos citar o *dropout* (Seção 2.1.6.1) e o *early stopping* (Seção 2.1.6.2), comumente utilizados com

as redes neurais profundas.

2.1.6.1 Dropout

O *dropout* (Srivastava et al., 2014) é uma das técnicas mais populares para regularização de modelos de *deep learning*. A ideia dessa técnica é simples: a cada iteração de treinamento, todos os neurônios (com exceção apenas da camada de saída), possuem uma probabilidade (chamada de taxa de *dropout*) de terem seus valores momentaneamente ignorados (Géron, 2017), conforme ilustrado pela Figura 2.12.

Figura 2.12: Comparativo entre uma rede neural sem nenhum tipo de regularização, à esquerda, e a aplicação da regularização baseada em *dropout*, à direita.



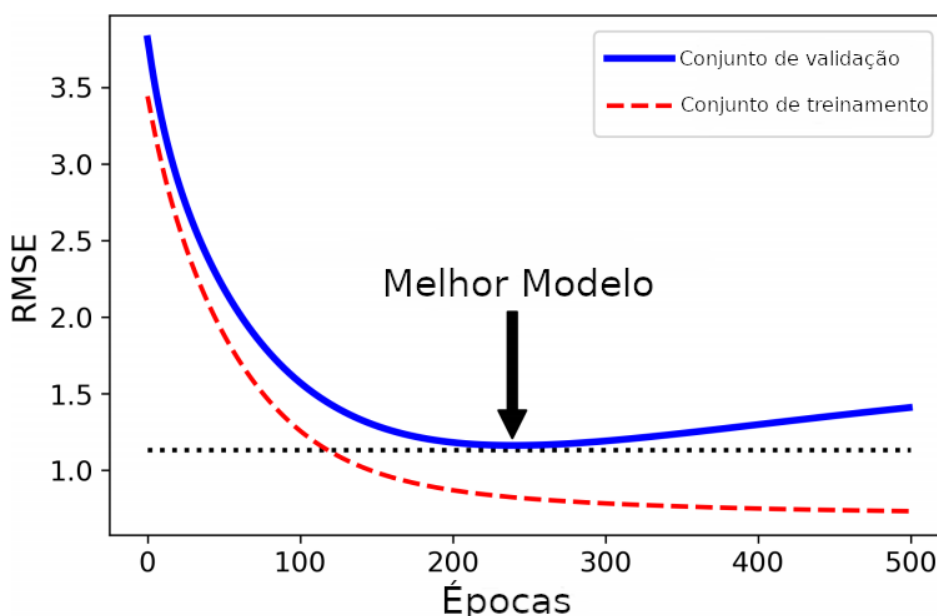
Fonte: Srivastava et al. (2014).

Aplicar o *dropout* equivale a amostrar uma rede neural menor para ser treinada a cada iteração (Srivastava et al., 2014), resultando em um modelo de arquitetura diferente, porém, com o compartilhamento dos mesmos parâmetros, fazendo com que os neurônios sejam menos coadaptados com a vizinhança e tenham maior importância individual (Géron, 2017). Dessa forma, essa técnica pode ser vista como uma aproximação de um *bagging* (ou *bootstrap aggregating* em inglês) (Maclin e Opitz, 1997) de um conjunto de redes neurais, porém de forma muito mais barata computacionalmente (Goodfellow et al., 2016). Por fim, o número de camadas e a taxa de *dropout* em cada camada costumam ser definidos a partir de treinamentos experimentais, visando principalmente aumentar a capacidade de generalização do modelo através de ganhos de acurácia nas bases de validação e teste.

2.1.6.2 *Early Stopping*

O *early stopping* (Géron, 2017) consiste em uma forma muito simples de regularização, porém de grande utilidade para algoritmos de otimização iterativa, como é o caso do gradiente descendente discutido na Seção 2.1.3.2. À medida em que o modelo é treinado, seu erro de validação diminui cada vez mais até chegar a um mínimo. Porém, conforme os parâmetros continuam sendo atualizados pelas iterações, é possível que os erros voltem a aumentar, indicando que o modelo está se sobreajustando à base de treinamento e perdendo a capacidade de generalizar os dados de validação, conforme ilustrado na Figura 2.13.

Figura 2.13: Representação visual do ponto de parada desejado para o modelo, evitando que o sobreajuste na base de treinamento degrade a capacidade de generalização.



Fonte: Adaptado de Géron (2017).

Dessa forma, a técnica de *early stopping* visa parar o treinamento do modelo no melhor momento possível, antes que haja sobreajuste. Além de poder ser feito de maneira visual, uma abordagem comum para essa parada é adicionar um critério, como um número i de iterações de treinamento sem a diminuição no custo, indicando que possivelmente o ponto mínimo foi atingido.

2.1.7 Métricas de avaliação

Assim como o nome sugere, as métricas de avaliação mensuram o desempenho do modelo treinado. Diferentes métricas são estabelecidas dependendo do tipo de problema tratado (classificação, regressão, treinamento não supervisionado, e entre outros) e também do objetivo que

se pretende atingir com a solução proposta. Em um contexto médico, onde se visa detectar a presença de uma determinada doença, por exemplo, é mais desejável que o modelo indique um paciente saudável como doente do que um doente como saudável. A consequência do primeiro tipo de erro, chamado de falso positivo, é apenas que o paciente deverá passar por uma avaliação médica mais cuidadosa e talvez realizar mais exames. Porém, no segundo tipo de erro, chamado falso negativo, o paciente pode ser liberado mesmo estando doente, aumentando as chances do agravamento da enfermidade. A métrica que mensura a quantidade de falsos positivos do modelo é a precisão, enquanto a revocação (ou *recall* em inglês) avalia a quantidade de falsos negativos.

Ainda assim, essa troca entre precisão e revocação não costuma ser considerada nos trabalhos relacionados a reconhecimento de sinais, visto que todas as palavras são tratadas de igual modo, visando acertar o maior número de sinais possível. Dito isso, a métrica mais comum para essa área é a acurácia (conforme evidenciado na Seção 3.2), definida na Equação 2.27:

$$\text{acurácia} = \frac{\text{Acertos do modelo}}{\text{Total de instâncias}}. \quad (2.27)$$

O cálculo da acurácia é bastante simples, gerando um valor entre 0 e 1, que aumenta junto aos acertos do modelo. Embora eficiente, essa métrica não funciona bem para classes muito desbalanceadas, visto que uma acurácia alta pode esconder um viés de acerto por parte do modelo, caso algumas classes estejam sendo muito mais representadas do que outras. Diferentemente de uma doença rara ou uma fraude bancária, os sinais são fáceis de serem reproduzidos e coletados para reduzir o efeito de desbalanceamento. Porém, métodos de criação de bases de dados baseado em mineração de conteúdo, conforme discutido na Seção 3.1.2, devem tomar um cuidado especial com esse fator, uma vez que algumas palavras são notoriamente mais comuns do que outras e, por isso, acabam se repetindo muito mais vezes, gerando uma tendência de desbalanceamento.

Por fim, outro ponto importante a se considerar na avaliação de modelos criados para adoção em ambiente real é o tempo médio de classificação de uma imagem em milissegundos (ms). Nesse caso, o valor obtido não diz respeito à convergência do modelo a um ponto ótimo, mas à complexidade da arquitetura adotada para geração das inferências.

Conforme explorado no Capítulo 1, embora dispositivos mais simples, baratos e móveis possuam maior chance de adesão como solução em larga escala, esses equipamentos possuem limitações de *hardware* que inviabilizam a execução da maior parte dos modelos de aprendizado

profundo existentes no estado da arte. Dentre as abordagens possíveis para alcançar melhores resultados no tempo médio da classificação de imagens, destacam-se:

- Redução na resolução das imagens de entrada;
- Redução no número de camadas e neurônios da rede;
- Simplificação das características extraídas para o modelo;
- Uso de escala de cinza;
- Quantização do modelo gerado.

Ainda assim, os pontos supracitados costumam afetar diretamente os resultados do modelo e diminuir sua robustez, devendo ser devidamente aplicados para gerar a melhor troca entre acurácia e custo computacional.

Por fim, a Seção 3.2 mostra que apenas uma minoria dos trabalhos recentes de reconhecimento de sinais se preocupam com a métrica de tempo de execução e uso de recursos computacionais dos modelos gerados, e nenhum deles aborda a adoção da solução fora de ambiente laboratorial, dada a complexidade do sistema criado. Neste trabalho, por sua vez, a velocidade de inferência e quantidade de parâmetros de cada um dos modelos gerados serão devidamente documentados, bem como as especificações de *hardware* e *software* utilizados nos experimentos, a fim de destacar as soluções com maior balanço entre as métricas supracitadas.

2.1.8 Considerações finais da Seção

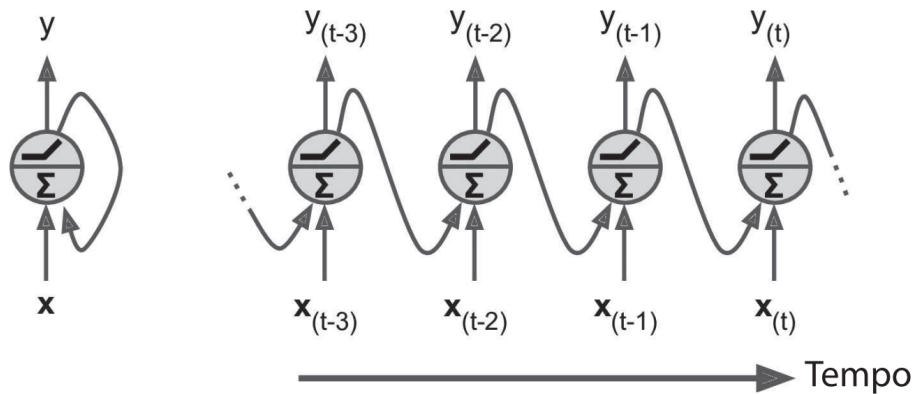
Conforme apresentado, as redes neurais são algoritmos capazes de resolver diversos tipos de problemas, desde os mais simples, como os linearmente separáveis, até os de maior complexidade. Isso ocorre graças aos diversos avanços alcançados ao longo dos anos, viabilizando o treinamento de redes neurais profundas através de técnicas que mitigaram problemáticas como a explosão ou sumiço do gradiente, *overfitting* e *underfitting* e estratégias que facilitam a convergência do modelo no espaço de busca de parâmetros. Ainda assim, o uso desses modelos não possui garantia de sucesso e a escolha dos hiperparâmetros que configuram o treinamento deve ser realizada de maneira cuidadosa, a depender do problema tratado. Na Seção 2.2, as redes neurais recorrentes serão detalhadas, sendo uma variação das redes neurais discutidas nesta seção que são comumente empregadas para problemas que envolvem séries temporais.

2.2 Redes neurais recorrentes

As redes neurais recorrentes, ou *recurrent neural networks* (RNNs) (Elman, 1990) em inglês, são um tipo de rede neural que, diferentemente das *feedforwards*, apresentam uma retroalimentação, ou seja, o neurônio transmite informação para ele mesmo. Essa abordagem é comum para resolver problemas que envolvam séries temporais, sendo uma das técnicas muito utilizadas para o reconhecimento de sinais (conforme discutido na Seção 3.2), pois permite representar a informação de movimento que se desenrola no decorrer dos quadros do vídeo, a qual compõe os parâmetros que formam os sinais, conforme discutido no Capítulo 1.1. Nesse contexto, as RNNs assumem o papel do classificador presente no final da arquitetura da CNN, após a convolução bidimensional das imagens, conforme visto na Seção 2.3.3. Dessa forma, enquanto a operação de convolução nos quadros do vídeo extrai as informações espaciais e as transformam em um vetor de características, o modelo recorrente analisa a série produzida para gerar o resultado final.

O funcionamento desse modelo gira em torno da estrutura do neurônio recorrente, o qual possui dois valores de entrada $\mathbf{x}_t$ e $\mathbf{y}_{(t-1)}$ considerando o instante de tempo t . A Figura 2.14 ilustra o desenrolar de um neurônio recorrente simples ao longo do tempo.

Figura 2.14: Neurônio recorrente e seu desenrolar através do tempo.



Fonte: Géron (2017).

Deste modo, cada camada de neurônios recorrentes possui dois conjuntos de pesos, $\mathbf{W}_x$ e $\mathbf{W}_y$, sendo que a matriz $\mathbf{W}_x$ é utilizada para modelar as entradas, representando o aprendizado da rede de maneira similar ao que foi discutido anteriormente, na Seção 2.1, enquanto a matriz $\mathbf{W}_y$ pondera as saídas do neurônio no intervalo de tempo anterior, modelando um fenômeno que pode ser comparado com uma memória. Assim, cada saída $\mathbf{y}_t$ de uma camada recorrente da rede é calculada de acordo com a Equação 2.28 (Géron, 2017):

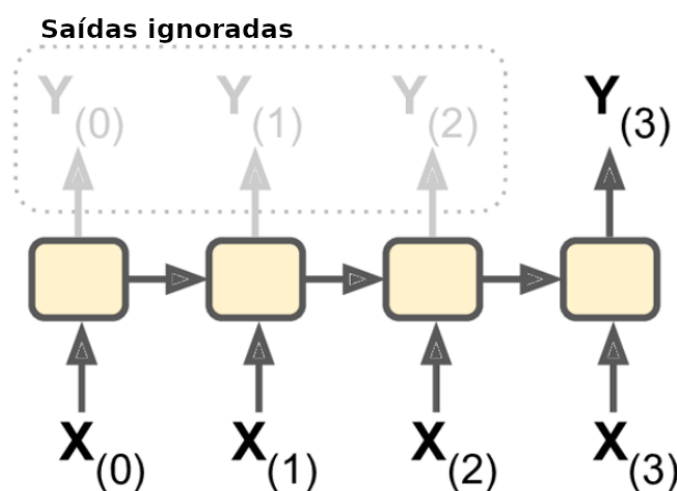
$$\mathbf{y}_{(t)} = \phi(\mathbf{W}_x^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_y^T \cdot \mathbf{y}_{(t-1)} + \mathbf{b}), \quad (2.28)$$

em que ϕ denota a função de ativação (usualmente uma tangente hiperbólica (Chung et al., 2014)) e $\mathbf{b}$ o termo de polarização da rede (ou viés, como também é conhecido).

Dessa forma, a saída dos neurônios recorrentes no instante de tempo t pode ser chamada de estado, por se tratar de uma função de suas entradas em intervalos de tempo anteriores. Estruturas de redes neurais que preservam algum tipo de estado são conhecidas como células de memória, ou simplesmente células (Géron, 2017). Embora em células mais básicas, como as camadas de neurônios recorrentes supracitados, o estado (denotado por $\mathbf{h}_{(t)}$) assumo o mesmo valor que a saída do neurônio ($\mathbf{y}_{(t)}$), esses valores se diferem em células mais complexas, visto que diferentes funções são adotadas para gerar as saídas e manter o valor do estado, conforme discutido nas Seções 2.2.1 e 2.2.2.

Além disso, as arquiteturas de RNNs podem lidar com entradas de tamanho dinâmico (como sentenças de frases ou quadros de vídeos) e apresentar diferentes padrões de saída, dependendo do objetivo pretendido, alterando a forma como a mesma é treinada. No caso do reconhecimento de sinais, a saída é do tipo sequência-para-vetor (Géron, 2017), onde a rede é alimentada com uma sequência de quadros do vídeo e todas as saídas são ignoradas, com exceção da última, a qual contém a previsão da classe, conforme ilustrado na Figura 2.15.

Figura 2.15: Arquitetura de RNN do tipo sequência-para-vetor.



Fonte: Adaptado de Géron (2017).

Por fim, arquiteturas profundas de RNNs podem ser criadas, empilhando as camadas de neurônios recorrentes de maneira similar ao que é feito no Perceptron multi-camadas, aumen-

tando a capacidade de extração de padrões complexos nas séries temporais. Ainda assim, treinar RNNs com muitos intervalos de tempo pode ser um desafio considerável, devido aos problemas de perda de memória, variação de magnitude dos gradientes e ao tempo de treinamento.

A perda de memória é um efeito natural que ocorre com as RNNs, onde a informação das primeiras entradas desaparece gradualmente, por conta da série de transformações realizadas no decorrer do modelo (Géron, 2017), fazendo com que o início da sequência seja praticamente desconsiderado e levando a possíveis erros na classificação. Outro problema característico desses modelos (Bengio et al., 1994) é a explosão ou sumiço dos gradientes, fazendo com que técnicas de otimização baseadas em gradiente (vistas na Seção 2.1.3) sejam pouco efetivas devido a essas variações de magnitude. Por fim, o tempo de treinamento também aumenta de maneira acentuada com o tamanho da sequência de entrada, dado o número de parâmetros e operações necessárias para retropropagar os gradientes pela rede (Géron, 2017).

Em função disso, diversos tipos de células recorrentes foram criadas a fim de mitigar os problemas supracitados, como a LSTM (*Long Short-Term Memory*), vista na Seção 2.2.1, e a GRU (*Gated Recurrent Unit*), explicada na Seção 2.2.2, viabilizando o treinamento dos modelos recorrentes por sequências mais longas e complexas.

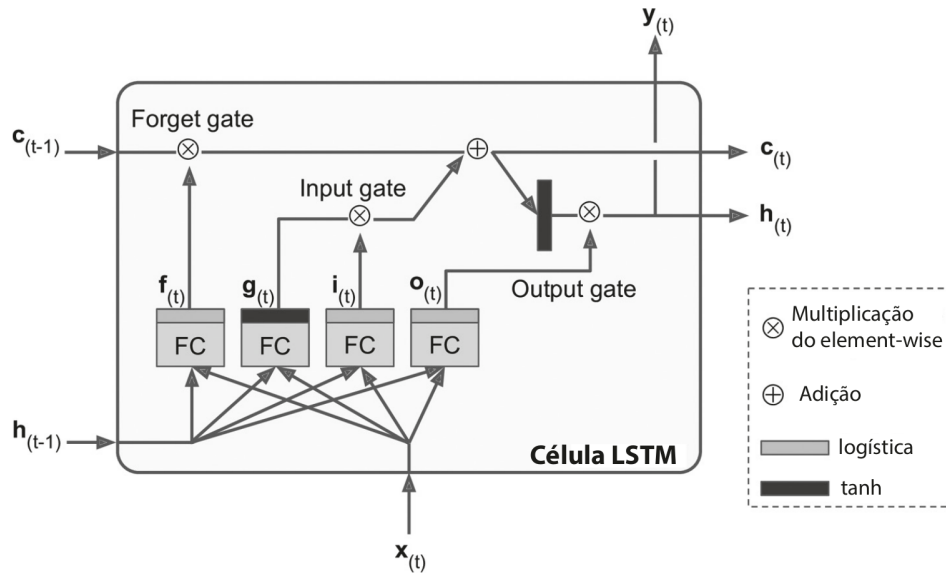
2.2.1 Células LSTM

As células do tipo Memória Longa de Curto Prazo, ou *Long Short-Term Memory* (LSTM) (Hochreiter e Schmidhuber, 1997) como são conhecidas, apresentam um desempenho muito superior ao neurônio recorrente convencional, sendo amplamente adotadas em trabalhos da literatura (Seção 3.2). O principal objetivo desse tipo de célula é aprender quais informações devem ser armazenadas no estado de longo prazo e quais devem ser descartadas, fazendo com que o modelo seja capaz de processar sequências de dados mais longas.

Para tal, o mecanismo de memória das LSTMs é composto de dois vetores $\mathbf{h}_{(t)}$ e $\mathbf{c}_{(t)}$, podendo ser interpretados como uma memória de curto prazo e outra de longo prazo, respectivamente. Além disso, cada célula de LSTM possui quatro camadas de neurônios completamente conectados, também chamadas de *Fully Connected* (FC), servindo para gerar e controlar as informações que serão transmitidas para intervalos de tempo posteriores, conforme ilustrado na Figura 2.16.

A camada $\mathbf{g}_{(t)}$ é considerada a principal, pois é a responsável por produzir os valores de saída $\mathbf{y}_{(t)}$ a partir das entradas atuais $\mathbf{x}_{(t)}$ e das saídas do intervalo de tempo anterior $\mathbf{h}_{(t)}$, conforme

Figura 2.16: Célula LSTM.

Fonte: [Géron \(2017\)](#).

apresentado na Equação 2.29 ([Géron, 2017](#)):

$$\mathbf{g}_{(t)} = \tanh(\mathbf{W}_{xg}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g), \quad (2.29)$$

em que $\tanh$ é a função de ativação tangente hiperbólica (discutida na Seção 2.1.1.2) e as matrizes $\mathbf{W}_{xg}$ e $\mathbf{W}_{hg}$ são os pesos utilizados para modelar as entradas e o estado de curto prazo, respectivamente.

Entretanto, diferentemente de um neurônio recorrente convencional, os valores produzidos pela camada $\mathbf{g}_{(t)}$ são modificados pelos controladores antes de serem atribuídos à saída atual $\mathbf{y}_{(t)}$ e à memória de curto prazo $\mathbf{h}_{(t)}$. Cada uma das camadas controladoras, também chamadas de *gates* (portões, em inglês), produzem valores de saída que variam de 0 a 1, devido ao uso da função de ativação logística (discutida na Seção 2.1.1.1), e são empregadas para diferentes propósitos.

A camada $\mathbf{f}_{(t)}$ (conhecida como “portão de esquecimento” ou *forget gate*, em inglês), por exemplo, controla as partes do estado de longo prazo $\mathbf{c}_{(t)}$ que serão apagadas no intervalo de tempo t através da multiplicação elemento por elemento (*element-wise*). Desse modo, valores próximos a 0 produzidos pela função logística dessa camada vão enfraquecer a informação da memória de longo prazo, enquanto valores próximos a 1 vão fazer com que a informação se mantenha através da sequência. A camada $\mathbf{f}_{(t)}$ pode ser expressada pela Equação 2.30 ([Géron, 2017](#)):

$$\mathbf{f}_{(t)} = \sigma(\mathbf{W}_{xf}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f), \quad (2.30)$$

em que σ representa a função logística (discutida na Seção 2.1.1.1) e as matrizes $\mathbf{W}_{xf}$ e $\mathbf{W}_{hf}$ são os pesos da camada $\mathbf{f}_{(t)}$.

De maneira similar, a camada $\mathbf{i}_{(t)}$ (conhecida como *input gate*) controla quais das saídas produzidas pela camada $\mathbf{g}_{(t)}$ serão adicionadas à memória de longo prazo, armazenando apenas as informações mais relevantes. Por fim, os novos valores presentes na memória de longo prazo são submetidos a uma função de ativação tangente hiperbólica, e a camada $\mathbf{o}_{(t)}$ (*output gate*) controla quais resultados serão produzidos como saída para o instante de tempo atual $\mathbf{y}_{(t)}$ e como memória de curto prazo para o próximo intervalo da rede recorrente ($\mathbf{h}_{(t)}$). As equações para definição das camadas $\mathbf{i}_{(t)}$ e $\mathbf{o}_{(t)}$ são idênticas à Equação 2.30, mudando apenas o fato de que cada camada possui as suas próprias matrizes de pesos.

Por fim, o processo de atualização da memória de longo prazo pode ser sintetizado através da Equação 2.31 (Géron, 2017):

$$\mathbf{c}_{(t)} = \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)}, \quad (2.31)$$

enquanto os valores de saída para o instante de tempo atual e para a memória de curto prazo ($\mathbf{y}_{(t)}$ e $\mathbf{h}_{(t)}$) são gerados de acordo com a Equação 2.32 (Géron, 2017):

$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)}). \quad (2.32)$$

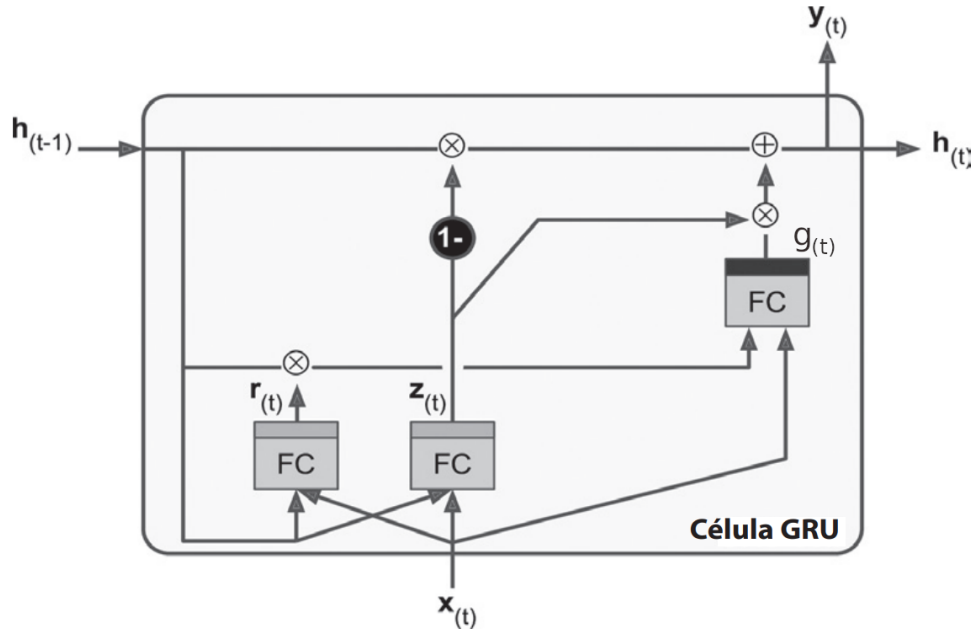
Desta forma, as LSTMs assumem a capacidade de aprender quais entradas são mais importantes para armazenamento a longo prazo (*input gate*), por quanto tempo as mesmas devem ser mantidas (*forget gate*) e quais valores devem ser produzidos para cada uma das sequências (*output gate*), atingindo sucesso em tarefas complexas de captura de padrões em sequências temporais, como análise de sentimentos em textos longos e classificação de vídeos (Géron, 2017), conforme evidenciado na Seção 3.2.

2.2.2 Células GRU

A célula do tipo *Gated Recurrent Unit* é uma variante popular à LSTM criada por Cho et al. (2014), ganhando destaque por gerar resultados similares ou até superiores, em alguns casos (Chung et al., 2014), enquanto apresenta uma arquitetura simplificada, conforme ilustrado na

Figura 2.17, contando com um único vetor de estado $\mathbf{h}_{(t)}$ e com menos portões de controle de informação, sendo eles o *reset gate* ($\mathbf{r}_{(t)}$) e o *update gate* ($\mathbf{z}_{(t)}$).

Figura 2.17: Célula GRU.



Fonte: Adaptado de [Géron \(2017\)](#).

Assim como nas LSTMs, a camada $\mathbf{g}_{(t)}$ é a principal responsável por gerar os resultados de saída ($\mathbf{y}_{(t)}$) e de estado ($\mathbf{h}_{(t)}$) da rede recorrente, porém, os valores do estado de tempo anterior ($\mathbf{h}_{(t-1)}$) são modificados pela camada $\mathbf{r}_{(t)}$ antes de serem utilizados como entrada, conforme expresso pela Equação 2.33 ([Géron, 2017](#)):

$$\mathbf{g}_{(t)} = \tanh(\mathbf{W}_{xg}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^T \cdot (\mathbf{r}_{(t)} \otimes \mathbf{h}_{(t-1)}) + \mathbf{b}_g), \quad (2.33)$$

em que $\mathbf{r}_{(t)}$ é definido pela Equação 2.34 ([Géron, 2017](#)):

$$\mathbf{r}_{(t)} = \sigma(\mathbf{W}_{xr}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hr}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_r). \quad (2.34)$$

Dessa forma, caso os valores produzidos pelo portão $\mathbf{r}_{(t)}$ sejam próximos a 0, a camada $\mathbf{g}_{(t)}$ é forçada a ignorar o vetor de estado $\mathbf{h}_{(t-1)}$, utilizando apenas a informação da entrada atual $\mathbf{x}_{(t)}$ ([Chung et al., 2014](#)). Assim, o principal objetivo da camada $\mathbf{r}_{(t)}$ é remover informações irrelevantes do vetor de estado $\mathbf{h}_{(t-1)}$, gerando uma representação mais compacta para a inferência dos valores no instante de tempo t .

Similarmente, a camada $\mathbf{z}_{(t)}$ também pode ser expressa pela Equação 2.34, alterando-se apenas as matrizes de pesos de $\mathbf{W}_{xr}^T$ e $\mathbf{W}_{hr}^T$ para $\mathbf{W}_{xz}^T$ e $\mathbf{W}_{hz}^T$. Ainda assim, o *update gate* atua

não só para controlar quanta informação do estado anterior $\mathbf{h}_{(t-1)}$ será mantida no vetor de estado atual $\mathbf{h}_{(t)}$, mas também para influenciar na adição dos valores gerados pela camada $\mathbf{g}_{(t)}$. Dito isso, o cálculo do estado $\mathbf{h}_{(t)}$ nas GRUs é dado pela Equação 2.35 (Géron, 2017):

$$\mathbf{h}_{(t)} = \mathbf{z}_{(t)} \otimes \mathbf{h}_{(t-1)} + (1 - \mathbf{z}_{(t)}) \otimes \mathbf{g}_{(t)}. \quad (2.35)$$

Com isso, unidades de GRU com *reset gates* ativos tendem a capturar informações de curto prazo, enquanto unidades com maior ativação nos *update gates* lidam com informações de longo prazo, fazendo com que essas células consigam aprender e capturar dependências em diferentes escalas temporais (Cho et al., 2014).

Por fim, embora as GRUs sejam mais simples e baratas computacionalmente do que as LSTMs (Chung et al., 2014), não há um consenso a respeito de qual célula apresenta os melhores resultados, podendo variar dependendo do tipo de problema e conjunto de dados empregado.

2.2.3 Redes recorrentes bidirecionais

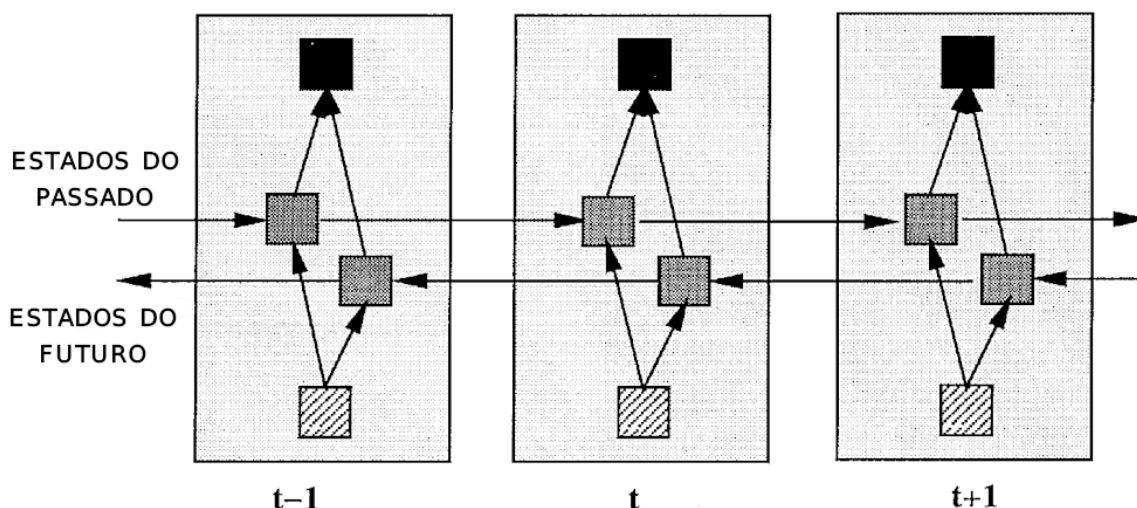
Em muitos problemas de reconhecimento de padrões em sequências temporais, a informação vinda de intervalos de tempo futuros ($t + 1, t + 2, \dots, t + n$) pode ser útil para auxiliar nas previsões geradas pelo modelo. Em tarefas de classificação textual, por exemplo, é possível observar a nítida influência que palavras em intervalos de tempo posteriores ($t + n$) podem ter no contexto e significado da informação processada no intervalo de tempo atual (t).

Ainda que a inclusão de informação futura possa ser parcialmente alcançada em arquiteturas tradicionais de RNNs, é desafiador encontrar o número ótimo de intervalos a serem atrasados para permitir que o modelo se beneficie dessa abordagem sem resultar em perda de acurácia (Schuster e Paliwal, 1997). Além disso, mesmo que dois modelos distintos sejam treinados, em que um recebe como entrada as sequências no intervalo t_0 até t_n e outro de t_n a t_0 , também seria necessário encontrar a melhor forma de unir suas previsões, o que pode variar dependendo do tipo de tarefa (Schuster e Paliwal, 1997).

Baseado nessas limitações, Schuster e Paliwal (1997) criaram a arquitetura de redes neurais recorrentes bidirecionais, ou *Bidirectional Recurrent Neural Networks* (BRNNs), em inglês, capazes de serem treinadas com toda a informação de entrada disponível, tanto do passado quanto do futuro, conforme ilustrado na Figura 2.18. Dessa forma, embora os estados da direção positiva (passado) e da direção negativa (futuro) não sejam diretamente conectados, essa informação é utilizada para gerar os resultados de saída em cada intervalo de tempo, influenciando a função

de custo do modelo e fazendo com que a rede aprenda a melhor processar ambas as direções temporais simultaneamente.

Figura 2.18: Representação da estrutura das BRNNs.



Fonte: Adaptado de [Schuster e Paliwal \(1997\)](#).

Além disso, os algoritmos utilizados no processo de treinamento e atualização de pesos das arquiteturas bidirecionais são os mesmos das RNNs unidirecionais ([Schuster e Paliwal, 1997](#)), visto que não há interação entre os dois estados de tempo, fazendo com que a adaptação para o uso desse tipo de célula seja simplificado. Assim, mesmo células mais complexas como as LSTMs e GRUs (discutidas nas Seções 2.2.1 e 2.2.2, respectivamente), podem ser adaptadas de maneira similar para se tornarem bidirecionais.

Por fim, embora as BRNNs possuam o dobro de parâmetros do que as RNNs convencionais, esse tipo de arquitetura tem sido amplamente utilizada em diferentes áreas de reconhecimento de padrões em séries temporais, incluindo o reconhecimento de língua de sinais (conforme mostrado na Seção 3.2), visto que esses modelos podem gerar ganhos expressivos nos resultados por possuírem mais informação contextual ([Schuster e Paliwal, 1997](#), [Graves e Schmidhuber, 2005](#), [Alawneh et al., 2020](#)).

2.2.4 Camadas de atenção

Ainda que células complexas como as LSTMs ou GRUs tenham facilitado o reconhecimento de padrões em sequências temporais mais longas, conforme discutido nas Seções 2.2.1 e 2.2.2, estes modelos precisam processar toda a sequência de informação de entrada a fim de conseguir

extrair padrões significativos para classificação. Porém, a maioria dos problemas com séries temporais possuem diferentes níveis de importância para cada intervalo de tempo. A execução de um sinal, por exemplo, é composta de inúmeros quadros com baixo valor semântico, por se tratar de uma transição entre a pose de descanso e o sinal propriamente dito.

Por conta disso, o uso de camadas de atenção (*attention layers*, em inglês) (Bahdanau et al., 2015) têm se popularizado, por ser uma técnica capaz de aumentar a atenção do modelo às partes mais importantes da sequência, garantindo que entradas com baixo valor semântico sejam ignoradas e facilitando a tarefa de classificação de séries temporais complexas.

Para que isso seja possível, um vetor de contexto $\mathbf{c}_i$ é gerado para cada saída i a partir de uma soma ponderada entre o estado da rede recorrente $\mathbf{h}_j$ do intervalo de entrada j e os pesos de atenção α_{ij} , conforme descrito na Equação 2.36 (Bahdanau et al., 2015):

$$\mathbf{c}_i = \sum_{j=1}^{T_x} \alpha_{ij} \mathbf{h}_j, \quad (2.36)$$

em que T_x é o número de intervalos de tempo da sequência de entrada $\mathbf{X}_T$, e os pesos α_{ij} para cada estado $\mathbf{h}_j$ são computados por uma função Softmax, conforme exibido na Equação 2.37 (Bahdanau et al., 2015):

$$\alpha_{ij} = \frac{\exp(\mathbf{e}_{ij})}{\sum_{k=1}^{T_x} \exp(\mathbf{e}_{ik})}, \quad (2.37)$$

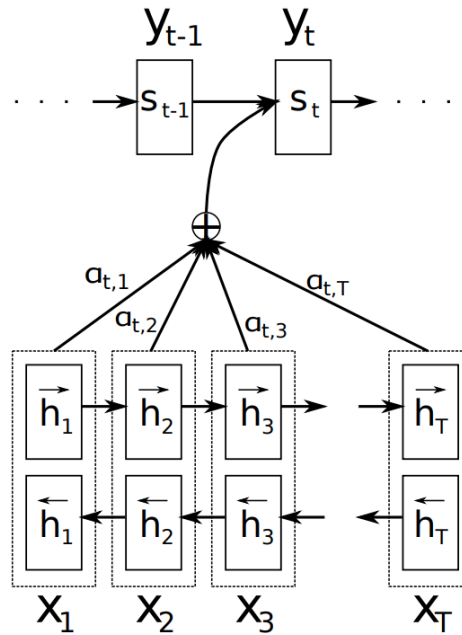
onde os valores de $\mathbf{e}_{ij}$, por sua vez, são obtidos através de um modelo de alinhamento a , representado na Equação 2.38 (Bahdanau et al., 2015):

$$\mathbf{e}_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j), \quad (2.38)$$

considerando que $\mathbf{s}_{i-1}$ é o estado do intervalo de tempo anterior da rede recorrente responsável por gerar as saídas, e o modelo a é uma rede neural *feedforward* com uma única camada. A Figura 2.19 exibe a arquitetura de um modelo de BRNN com uso da camada de atenção.

Dessa forma, os parâmetros de atenção que compõem a rede neural a são treinados junto aos demais parâmetros do modelo e são responsáveis por determinar o quanto o vetor de contexto $\mathbf{c}_i$ depende dos elementos de entrada da sequência. Feito isso, o contexto é utilizado por outra rede recorrente (geralmente chamada de decodificador, ou *decoder*) para gerar o valor de saída ($\mathbf{y}_i$).

Figura 2.19: Representação do uso de vetores de contexto em redes recorrentes bidirecionais.



Fonte: Bahdanau et al. (2015).

Assim, a normalização dos parâmetros através da função Softmax faz com que intervalos de tempo mais importantes recebam um valor próximo a 1, enquanto intervalos menos importantes sejam facilmente ignorados por assumirem valores próximos a 0, fazendo com que o modelo aprenda a destacar partes relevantes das sequências de entrada conforme sua convergência.

Por fim, embora o algoritmo de atenção possua um custo quadrático (Bahdanau et al., 2015), aumentando o número de parâmetros de acordo com os tamanhos das sequências de entrada e saída, a maioria dos problemas envolvendo séries temporais não possuem sequências tão longas a ponto do custo computacional inviabilizar a adoção dessa técnica.

2.2.5 Considerações finais da seção

A capacidade de extrair informação a partir de sequências temporais é uma habilidade de extrema importância para a resolução de determinados problemas, como o reconhecimento de língua de sinais. Conforme apresentado nesta seção, redes neurais recorrentes têm se popularizado por obter êxito nesse tipo de atividade, destacando principalmente as arquiteturas que lidam com o aprendizado de memória de forma mais granular, como o caso das LSTMs e GRUs, tornando-se opções viáveis para as tarefas de classificação baseadas em intervalos de tempo. A seguir, na Seção 2.3, as CNNs serão discutidas em mais detalhes, sendo a técnica utilizada no estado da arte para extração de características e reconhecimento de sinais, e que podem atuar

em conjunto com os modelos de RNNs supracitados.

2.3 Redes neurais convolucionais

Embora as redes neurais profundas tenham se provado mais capazes de resolver problemas complexos do que qualquer outro modelo de aprendizado de máquina, o custo computacional é um fator que não pode ser ignorado, especialmente em problemas de alta dimensionalidade. Isso acontece, por exemplo, com o caso do reconhecimento de imagens, devido à quantidade de informação presente nos dados. Considerando o exemplo de uma imagem RGB de 300×300 pixels, o número de neurônios na camada de entrada seria de $300 \times 300 \times 3 = 270.000$. Levando em conta uma rede densamente conectada, ou seja, em que todos os neurônios se conectam com todos da camada seguinte, a quantidade de parâmetros a serem aprendidos por essa rede neural tornaria o processo de convergência extremamente lento e custoso, além de promover o efeito de *overfitting*.

Tendo isso em vista, as CNNs têm sido amplamente adotadas nos últimos anos devido ao aumento de capacidade computacional e também aos bons resultados atingidos, assim como demonstrado pela arquitetura criada por [Krizhevsky et al. \(2012\)](#), sendo uma das primeiras a ganhar destaque em competições da área. Estes modelos possuem inspiração biológica no córtex visual humano e têm por objetivo possibilitar a tarefa de reconhecimento de imagens de maneira mais eficiente, através do uso de três principais etapas, sendo elas: a operação de convolução, descrita na Seção 2.3.1, o *pooling*, discutido na Seção 2.3.2, e os classificadores, apresentados na Seção 2.3.3.

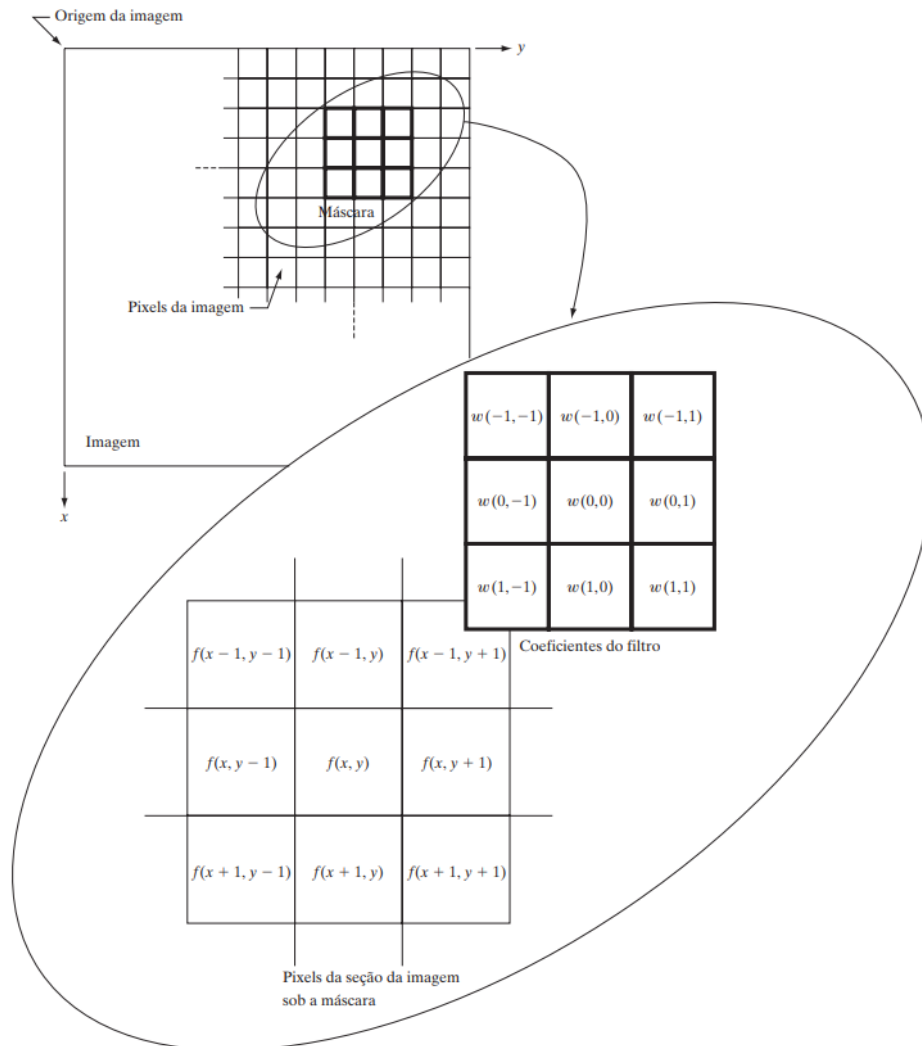
Além disso, esta Seção também visa explorar outras técnicas que têm sido amplamente adotadas em conjunto às CNNs para facilitar o treinamento desses modelos, como o *global average pooling* (Seção 2.3.4), o *batch normalization* (Seção 2.3.5), o aumento de dados (Seção 2.3.6) e a transferência de aprendizado (Seção 2.3.7). Ademais, arquiteturas criadas para fins específicos, como as CNNs 3D (Seção 2.3.8) para contextos temporais, as CNNs multicanais (Seção 2.3.10) para extração de características locais e fusão de informação e as CNNs 2+1D (Seção 2.3.9) para exploração temporal de maneira mais eficiente, também serão abordadas.

2.3.1 Operação de convolução

No contexto de processamento digital de imagens, a convolução é o processo de mover uma máscara $w(x,y)$ pela imagem $f(x,y)$ e calcular a soma dos produtos em cada posição, sendo

considerada uma operação de filtragem espacial (Gonzalez e Woods, 2009). A Figura 2.20 ilustra o processo, onde (x, y) representa um ponto na imagem.

Figura 2.20: Exemplo de uma operação de convolução.



Fonte: Gonzalez e Woods (2009).

A máscara, também chamada de *kernel*, é uma matriz de tamanho menor que a imagem de entrada e tem por objetivo atuar como um filtro espacial, podendo ser linear ou não linear (Gonzalez e Woods, 2009). A máscara deve ter seus coeficientes especificados a depender do objetivo que se pretende atingir pelo processo de convolução com a imagem, atenuando ou destacando características de interesse.

Uma das principais vantagens da introdução do aprendizado de máquina à convolução é que os valores dos coeficientes das máscaras são aprendidos automaticamente, compondo parte dos parâmetros que são adaptados com o treinamento da rede. Com isso, o modelo converge de maneira a destacar os componentes mais relevantes para a discriminação dos objetos na

imagem, eliminando a necessidade de considerar todos os *pixels* presentes e diminuindo o custo computacional. Em um classificador de gatos e cachorros, por exemplo, a CNN irá encontrar as melhores máscaras para destacar características como as patas, focinho, olhos e orelhas, pois são os aspectos que discriminam visualmente as duas classes de animais, gerando o menor erro possível ao passo que apenas os componentes que realmente importam são ponderados.

O processo iterativo de aplicação da máscara sobre a imagem, já ilustrado na Figura 2.20, pode ser expresso matematicamente pela Equação 2.39 (Gonzalez e Woods, 2009):

$$w(x, y) * f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x-s, y-t), \quad (2.39)$$

em que m e n são as dimensões da máscara, utilizadas em a e b , conforme definido pelas Equações 2.40 e 2.41 (Gonzalez e Woods, 2009):

$$a = (m - 1)/2, \quad (2.40)$$

$$b = (n - 1)/2. \quad (2.41)$$

Um ponto também explorado pela convolução é a dependência espacial inerente às imagens digitais, uma vez que as mesmas são formadas por um arranjo matricial que faz com que determinadas estruturas estejam sempre espacialmente próximas uma das outras (Gonzalez e Woods, 2009), o que pode ser considerado para o reconhecimento de padrões e redução de dimensionalidade da imagem.

Outras características importantes ressaltadas por essa operação são as conexões esparsas e o compartilhamento de parâmetros (Goodfellow et al., 2016). As conexões esparsas se dão pelo fato de que as entradas não estão diretamente correlacionadas a um parâmetro para gerar a saída, ao contrário do que ocorre com as redes densamente conectadas, reduzindo o número de parâmetros necessários para processar imagens com centenas de milhares de *pixels*. O compartilhamento de parâmetros, por sua vez, ocorre pelo fato de que cada coeficiente da máscara é usado para cada um dos *pixels* de entrada, fazendo com que apenas um único conjunto de parâmetros seja aprendido para representar diferentes localidades da imagem.

Dessa forma, as redes neurais baseadas em convolução conseguem aprender a extrair padrões complexos das imagens digitais, ao passo que são viáveis computacionalmente, tornando-se o modelo padrão para a maioria dos problemas da área.

2.3.2 Pooling

O *pooling* é uma das camadas que comumente fazem parte da composição de uma CNN, onde seu principal objetivo é modificar a saída da camada anterior (a de convolução, por exemplo) com uma estatística resumida das saídas próximas (Goodfellow et al., 2016). Uma das versões mais utilizadas desse algoritmo é o *max pooling*, que conserva apenas o maior valor dentro de uma vizinhança retangular. Com isso, a operação de *pooling* aumenta a invariância do modelo a pequenas translações na imagem de entrada (Goodfellow et al., 2016), o que é útil na maioria dos casos considerando que o objetivo principal é saber se uma característica está ou não presente na imagem, e não sua localização exata.

Outra vantagem importante dessa operação é que ela sumariza as respostas provenientes da convolução, reduzindo as dimensões da imagem e também o número de parâmetros da rede, resultando em maior eficiência computacional.

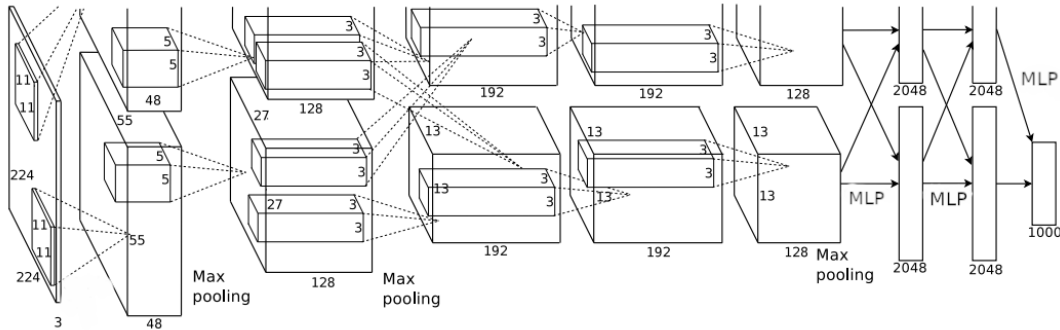
2.3.3 Classificadores

Uma vez que as características principais da imagem foram destacadas através de uma série de camadas de convolução e *pooling* empilhadas, os modelos de classificação são utilizados para capturar os padrões presentes nas características extraídas e gerar a previsão da classe do objeto que está na imagem, sendo a etapa final que compõe a arquitetura da CNN.

Embora qualquer tipo de classificador possa ser utilizado, as redes neurais multicamadas são comumente empregadas quando a informação de movimento é processada por meio do extrator de características (utilizando CNNs 3D, por exemplo) ou quando a tarefa não exige informação temporal, caso contrário, redes recorrentes costumam ser adotadas. O processo de inferência para classificação ocorre de maneira similar ao que foi discutido na Seção 2.1, com exceção de que as características provenientes das camadas anteriores geralmente precisam ser achatadas para um formato vetorial para servir como entrada para a rede. A função *Softmax* (Seção 2.1.1.5) também é utilizada na camada de saída, gerando a probabilidade de cada uma das classes estar presente na imagem que está sendo avaliada.

A Figura 2.21 exemplifica uma arquitetura de CNN que emprega todas as etapas supracitadas, chamada AlexNet, que foi proposta por Krizhevsky et al. (2012) e empregada com sucesso para classificação de imagens na competição LSVRC (*ImageNet Large Scale Visual Recognition Challenge*) (Russakovsky et al., 2015).

Figura 2.21: Exemplo de uma arquitetura de CNN com o empilhamento das camadas de extração de característica (convolução e *max pooling*) e o classificador (*Multilayer Perceptron*)



Fonte: Adaptado de Krizhevsky et al. (2012).

2.3.4 Global Average Pooling

O *global average pooling* (Lin et al., 2013) é outra técnica que tem sido muito utilizada em algumas arquiteturas de CNN, substituindo o papel dos classificadores discutidos na Seção 2.3.3. Ao invés de se utilizar uma rede neural, por exemplo, as médias dos valores de cada um dos mapas de características gerados pela última camada da CNN são obtidas e transmitidas diretamente para uma camada *Softmax* para realizar as classificações. Com essa abordagem, não se faz necessário empregar um outro modelo para extrair padrões e gerar as previsões, visto que a própria CNN é forçada a adaptar os mapas de característica (imagens geradas após a convolução) de maneira a corresponder diretamente a cada uma das classes, tornando-se algo similar a mapas de confiança de categoria.

Além disso, a técnica é menos suscetível a *overfitting*, pois nenhum parâmetro adicional precisa ser adaptado, além de reduzir o custo computacional e aumentar a invariância a pequenas translações, por se tratar de uma operação de *pooling* espacial.

2.3.5 Batch normalization

A normalização em lotes, ou *batch normalization* como normalmente é chamado, é uma técnica proposta por Ioffe e Szegedy (2015) e tem por objetivo acelerar o treinamento de redes neurais profundas através da normalização das entradas das camadas para cada lote de dados, possibilitando que a taxa de aprendizado seja maior e menos cuidado seja necessário para inicialização dos parâmetros.

A normalização das entradas garante que a distribuição de cada camada seja mais estável,

através do desvio padrão unitário e centralização da média em zero, o que torna o processo de treinamento mais simples e agiliza a convergência do modelo (Ioffe e Szegedy, 2015). O *batch normalization* pode ser definido pela Equação 2.42 (Goodfellow et al., 2016):

$$H' = \frac{H - \mu}{\sigma}, \quad (2.42)$$

onde H é o lote de dados com os valores das funções de ativação a serem normalizadas, arranjados em formato matricial, enquanto μ é a média e σ o desvio padrão, conforme definido pelas Equações 2.43 e 2.44 (Goodfellow et al., 2016):

$$\mu = \frac{1}{m} \sum_i H_i \quad (2.43)$$

e

$$\sigma = \sqrt{\delta + \frac{1}{m} \sum_i (H - \mu)_i^2} \quad (2.44)$$

Um valor positivo pequeno (δ) é utilizado para impedir que o gradiente seja $\sqrt{0}$. Para os dados de teste, os valores de μ e σ são substituídos por uma média móvel exponencial calculada no momento do treinamento, visto que o lote de dados não está disponível (Goodfellow et al., 2016). Ioffe e Szegedy (2015) conseguiram treinar um modelo de classificação de imagens estado da arte com 14 vezes menos etapas de treinamento, provando a eficiência da normalização em lotes e tornando esse um método muito adotado no contexto de *deep learning*.

2.3.6 Aumento de dados

O aumento de dados, também chamado de *data augmentation*, é uma técnica amplamente utilizada na área de classificação de imagens para reduzir o efeito de *overfitting* e tornar o modelo mais robusto (Géron, 2017). Essa técnica visa aplicar aleatoriamente modificações nas imagens, como transformações geométricas e de intensidade, gerando novas instâncias para o treinamento.

Uma das principais vantagens dessa abordagem é o aumento da invariância do modelo a mudanças no posicionamento, escala, cor e luminosidade dos objetos, garantindo o reconhecimento da figura em diferentes circunstâncias e possibilitando a adoção do modelo em condições adversas. A Figura 2.22 ilustra diferentes operações de aumento de dados aplicadas a um sinal em LIBRAS.

Dentre as operações mais comuns para aumento de dados podemos citar as transformações

Figura 2.22: Aumento de dados aplicado no sinal ÔNIBUS em LIBRAS através de transformações geométricas e de intensidade.



Fonte: Autor.

geométricas e de intensidade. Transformações geométricas são caracterizadas pelo mapeamento do pixel na imagem original para uma nova posição na imagem resultante, gerando deslocamentos espaciais como a rotação, translação, cisalhamento e aumento de escala. As transformações de intensidade, por outro lado, dizem respeito à alteração no valor do pixel através de fórmulas matemáticas (*log*, potência, raiz quadrada e entre outros), alterando as propriedades de brilho e cor (Gonzalez e Woods, 2009).

Geralmente, a aplicação do aumento de dados ocorre durante do treinamento do modelo, fazendo com que as novas imagens sejam geradas apenas no momento em que serão usadas. Isso evita o uso de espaço extra de armazenamento, além de reduzir ainda mais as chances de *overfitting*, pois a base de dados nunca será a mesma em cada época de treinamento dada a estocasticidade da aplicação das operações.

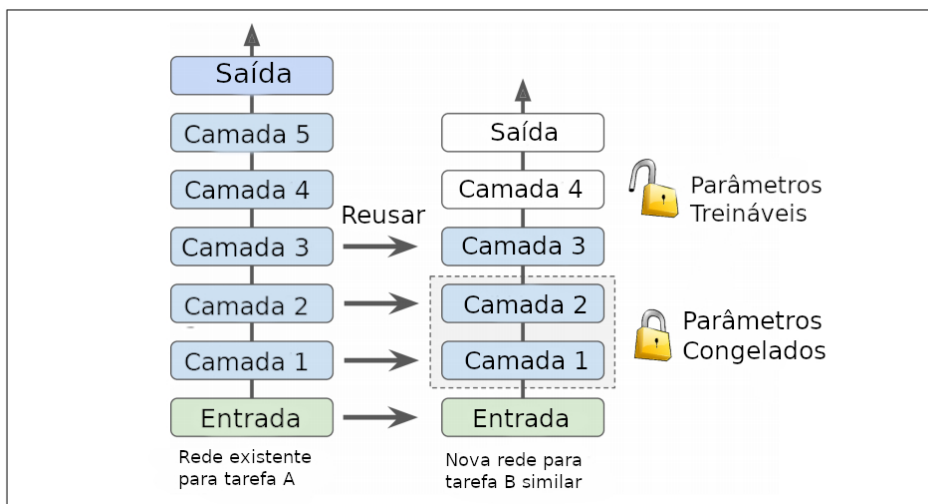
Por fim, o estudo e aplicação dessas técnicas é especialmente útil no contexto da língua de sinais, visto a dificuldade na criação ou obtenção de conjuntos de dados em larga escala, conforme discutido no Capítulo 3.1. Embora alguns trabalhos já tenham aplicado o aumento de dados e provado o ganho de acurácia nesta área, é importante que a escolha dos tipos de transformações seja cuidadosa, pois além de agregarem complexidade computacional também existe o risco de degradação da performance do modelo (Cubuk et al., 2019, Carneiro et al., 2021).

2.3.7 Transferência de aprendizado

Outra abordagem amplamente utilizada para mitigar os efeitos já discutidos de *overfitting* e também permitir o treinamento de modelos robustos mesmo sem uma grande quantidade de dados é a transferência de aprendizado (Géron, 2017). Essa técnica consiste basicamente em

reutilizar ou transferir os parâmetros que foram aprendidos em outro problema de domínio similar através do treinamento com milhares ou até milhões de imagens. A Figura 2.23 ilustra a forma como esse processo normalmente é realizado.

Figura 2.23: Exemplo de uma arquitetura de transferência de aprendizado.



Fonte: Adaptado de [Géron \(2017\)](#).

Neste processo, as camadas e os parâmetros de uma CNN treinada em outro problema são reutilizados, substituindo-se apenas as camadas superiores que serão específicas à tarefa atual. Além disso, os pesos transferidos são inicialmente “congelados”, ou seja, a retropropagação do erro não atualiza seus valores. Isso acontece pois, no início do treinamento, os gradientes gerados são muito altos, o que pode fazer com que o extrator de características já otimizado se degrade rapidamente. Após um treinamento inicial (chamado de aquecimento, ou *warmup*), o ajuste fino (ou *fine tuning*) é realizado, onde algumas camadas transferidas são então descongeladas para serem melhor adaptadas ao problema atual. Quanto mais dados disponíveis no conjunto, maior o número de camadas a serem descongeladas ([Géron, 2017](#)).

É muito comum que sejam utilizados conjuntos de dados de larga escala como base para o treinamento dos modelos utilizados na transferência de aprendizado, como o ImageNet ([Deng et al., 2009](#)) ou o COCO (*Common Objects in Context*) ([Lin et al., 2014](#)), uma vez que o grande número de classes e imagens favorecem o aprendizado de características básicas como cores, bordas e formas geométricas que servem de propósito geral para classificação. Ainda assim, quanto mais específico o conjunto de dados, maiores são as chances de se atingir bons resultados, visto que a fase de ajuste fino dos parâmetros é facilitada devido à similaridade das características a serem aprendidas.

Baseado nisso, é possível inferir que a área de reconhecimento de sinais é especialmente

beneficiada pelo uso de tal técnica, uma vez que os parâmetros que compõe um sinal são os mesmos para todas as linguagens de sinais ao redor do mundo, fazendo com que as características aprendidas por um modelo de determinado idioma sejam facilmente transferidas para outros, reduzindo as barreiras da adoção de uma ferramenta de acessibilidade em larga escala. Conforme exibido no Capítulo 3, alguns trabalhos tem aplicado com êxito a mesma arquitetura de reconhecimento em bases de sinais de diferentes idiomas.

Por outro lado, uma desvantagem dessa técnica é a necessidade de se utilizar a mesma arquitetura base do modelo treinado, não sendo possível aproveitar os parâmetros entre diferentes arquiteturas. Além disso, o ajuste fino dos parâmetros costuma ser complexo, uma vez que pode levar à rápida degradação das características aprendidas ou ao *overfitting* do modelo, além de possuir um custo computacional considerável, visto que o modelo base geralmente possui milhões de parâmetros.

2.3.8 Redes neurais convolucionais 3D

Conforme discutido, as CNNs desempenham um papel fundamental para a extração de características espaciais, sendo determinantes para o reconhecimento de imagens. Ainda assim, o reconhecimento de sinais também depende da dimensão temporal, dada a existência de sinais polissêmicos que fazem com que seja necessária a análise contextual da fala para escolha adequada do sinal, conforme apresentado na Seção 1.1, além do próprio parâmetro de movimento, que compõe e difere diversas palavras. Nesse âmbito, a dimensão temporal é dada pela sequência de quadros que formam o vídeo capturado na execução dos sinais. Uma abordagem comum para cenários como esse, onde há o interesse na extração de características que se desdobram ao longo do tempo, é o uso das redes neurais convolucionais de três dimensões, chamadas simplesmente de CNNs 3D.

Estes modelos vêm sendo aplicados em trabalhos recentes no reconhecimento de ações e gestos em vídeos, dada a sua capacidade de exploração espaço-temporal, atingindo resultados estado da arte. Conforme exibido no Capítulo 3, o reconhecimento de sinais também têm se beneficiado do uso de tais arquiteturas. Embora possua algumas mudanças na forma de processar a informação, a fim de extrair as características da sequência de imagens RGB empilhadas, as CNNs 3D seguem conceitos similares ao que foi mostrado na Seção 2.3, utilizando operações de convolução e *pooling* para reconhecer padrões, adaptando a máscara espacialmente e temporalmente.

Tran et al. (2015), por exemplo, propuseram um método de 3D CNN chamado por eles de C3D, com o objetivo de ser uma solução genérica, compacta, simples de implementar e eficiente computacionalmente, atingindo resultados estado da arte em quatro *benchmarks* diferentes. A criação da arquitetura tridimensional se inspirou em paradigmas vistos em modelos 2D, como o uso de campos receptivos pequenos de apenas 3x3. Os autores padronizaram a entrada da rede neural a 16 quadros além de propor 3 dimensões de profundidade para o *kernel*.

Embora tenha atingido bons resultados, a C3D possui duas principais limitações. Primeiro, sua arquitetura é muito rasa devido à quantidade de parâmetros, possuindo apenas oito camadas convolucionais e cinco *max poolings*, enquanto algumas arquiteturas bidimensionais chegam a possuir mais de cento e cinquenta camadas, como o caso das redes residuais (He et al., 2016). Além disso, a transferência de aprendizado não é aplicada à C3D, deixando de aproveitar os parâmetros já aprendidos em conjuntos de larga escala e sendo necessário iniciar o treinamento do zero.

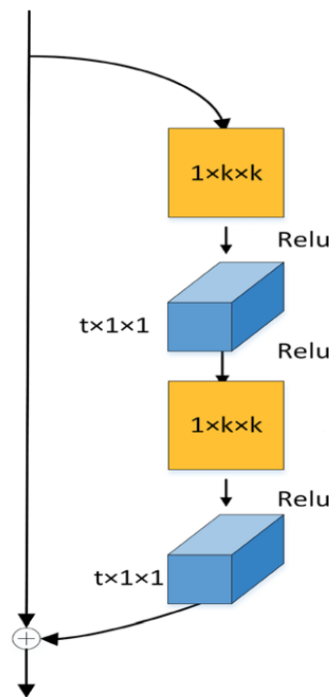
Como proposta a essas limitações, Carreira e Zisserman (2017) criaram a I3D (*Two-Stream Inflated 3D ConvNet* em inglês). Esse modelo foi desenvolvido a partir de arquiteturas estado da arte no reconhecimento de imagens, como o Inception-v1 (Ioffe e Szegedy, 2015), inflando os filtros e operações de *pooling* para atender à dimensão temporal. Além disso, os autores foram capazes de utilizar os parâmetros já aprendidos por essas arquiteturas 2D no conjunto de dados do ImageNet como inicialização para o treinamento. O método foi avaliado em diversos conjuntos de vídeos para reconhecimento de gestos, sendo o Kinetics o mais notório (Kay et al., 2017), com 400 ações diferentes e mais de 400 vídeos para cada ação, coletados a partir do Youtube. Os resultados mostram a superioridade do I3D em relação a outros métodos existentes, alcançando 15% mais acurácia do que a arquitetura do C3D, além de diminuir o número de parâmetros, facilitando o treinamento. Por fim, os autores também mostraram ser possível utilizar os parâmetros aprendidos em um domínio 2D para um domínio 3D, aumentando a acurácia em mais de 3% com a transferência de aprendizado.

Ainda assim, as arquiteturas 3D podem ser complexas de se adequarem aos dados, podendo levar ao *overfitting* do modelo (Suhajito et al., 2018) se a base de dados for muito pequena ou se os parâmetros não forem cuidadosamente ajustados. Além disso, o custo computacional (Tran et al., 2015) é uma limitação que não pode ser ignorada, visto que estes modelos costumam possuir uma grande quantidade de parâmetros.

2.3.9 Redes Neurais Convolucionais 2+1D

As CNNs 2+1D (Tran et al., 2018) também são voltadas para a extração espaço-temporal de características, porém, têm ganho destaque por reduzir o número de parâmetros e a consequente complexidade e custo computacional do modelo (Wang et al., 2021) enquanto melhora os resultados. Para tal, ao invés de utilizar um filtro de dimensões $t \times k \times k$ para extrair as características, assim como acontece na convolução 3D, onde k representa as dimensões espaciais da imagem e t a dimensão temporal, a arquitetura em questão divide as operações de convolução em 2D e 1D, com dimensões de $1 \times k \times k$ e $t \times 1 \times 1$, respectivamente, aplicando-as separadamente e em sequência a uma função de ativação Relu (Seção 2.1.1.3), conforme ilustrado pela Figura 2.24.

Figura 2.24: Operação básica de convolução 2+1D.



Fonte: Adaptado de Wang et al. (2021).

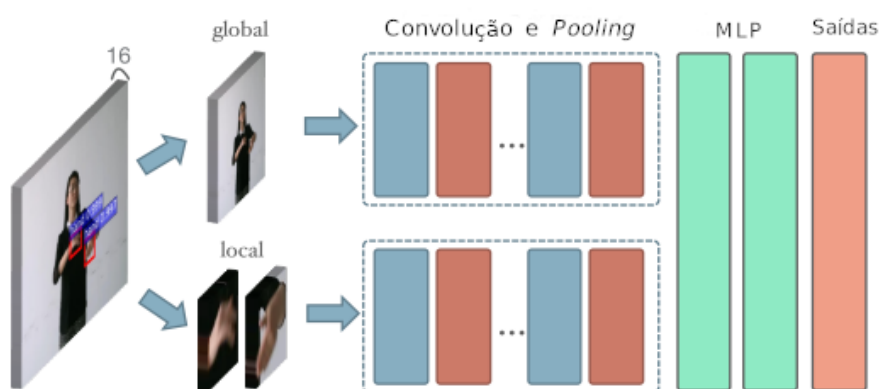
Isto faz com que cada bloco de convolução possua o dobro de funções não lineares, devido à adição da Relu entre as duas operações, permitindo que o modelo consiga representar funções mais complexas. Além disso, esta arquitetura possui conexões residuais inspiradas nas ResNets (He et al., 2016) e tem se mostrado mais fácil de ser otimizada do que as CNNs 3D, apresentando menores erros, especialmente quando a profundidade da rede é maior (Tran et al., 2018).

Por fim, além de obter bons resultados na tarefa de reconhecimento de ações em vídeos, as redes 2+1D também podem ser empregadas para o contexto da língua de sinais, conforme abordado no Capítulo 3.

2.3.10 Redes neurais convolucionais multicanais

As CNNs multicanais são abordagens adotadas comumente para o problema de reconhecimento de ações em vídeos, onde o objetivo é utilizar características adicionais presentes na imagem que sejam relevantes para classificação do gesto sendo realizado (TU, 2018). Ainda assim, conforme discutido nos Capítulos 3 e 5, diversos autores têm utilizado essa abordagem no contexto do reconhecimento de sinais para extrair dicas visuais da imagem que favoreçam a discriminação dos sinais, como as mãos e o rosto do intérprete, fazendo com que o modelo retenha mais atenção a essas partes de interesse. A Figura 2.25 ilustra essa técnica.

Figura 2.25: Exemplo de uma arquitetura de CNN multicanal no contexto do reconhecimento de sinais.



Fonte: Adaptado de Huang et al. (2018).

Conforme exibido na Figura 2.25, são utilizados extratores de características baseados em CNNs (podendo ter ou não arquiteturas semelhantes entre os canais), tanto para a imagem global quanto para a local, unindo os vetores de características como entrada para um classificador (geralmente uma rede neural completamente conectada). Essa abordagem é conhecida como fusão a nível de características (ou antecipada), pois combina as imagens em uma única representação vetorial para o classificador.

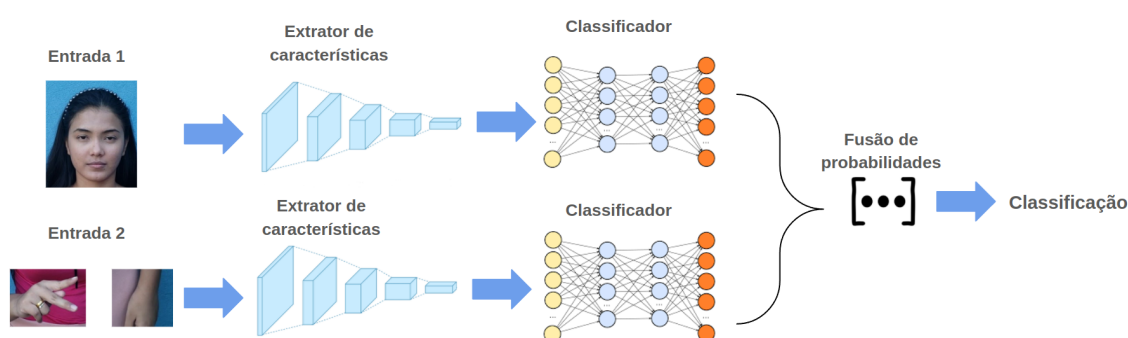
Os capítulos subsequentes mostram os ganhos expressivos de acurácia do modelo gerados pela abordagem multicanal, podendo ser explicado pelo fato de que as dicas visuais (mãos

e face), embora importantes para composição do sinal, representam uma parte proporcionalmente pequena da imagem, fazendo com que suas características sejam dificilmente destacadas pelos filtros convolucionais, ainda mais considerando o comportamento das CNNs de resumir as informações espaciais.

Além do foco local, outras características como fluxo óptico e informações de pose também são comuns de serem processadas em canais distintos e unidas posteriormente. Entretanto, a combinação de características distintas em uma única representação para o classificador é um desafio considerável para o aspecto de convergência dos modelos (Snoek et al., 2005).

A fusão tardia de características (ou fusão a nível de decisão) (Snoek et al., 2005), ilustrada pela Figura 2.26, também é um método alternativo para o processamento dos canais. Nessa abordagem, as características são consideradas separadamente em classificadores distintos, unindo-se apenas as probabilidades dos classificadores para tomada final de decisão. Embora o custo computacional seja maior, tanto para treinamento quanto para inferência, essa modalidade costuma ser mais simples de abstrair diferentes características em uma única tarefa de classificação, pois as representações e padrões são aprendidos separadamente. Ainda assim, existem inúmeras formas de se unir as probabilidades dos classificadores, desde a adoção de uma média simples até o uso de algoritmos de meta-aprendizado (Sagi e Rokach, 2018), sendo necessário descobrir a solução que melhor se adapta ao problema tratado.

Figura 2.26: Processo de fusão tardia para classificação.



Fonte: Autor.

Por fim, ainda que estas abordagens tenham o potencial de melhorar os resultados, a preparação dos dados de entrada da arquitetura também se torna mais complexa, pois muitas das vezes é preciso utilizar algoritmos especializados para extrair as novas características, como o caso dos detectores de objetos, abordados na Seção 2.4, e os estimadores de pose, vistos na Seção 2.5. Além disso, o custo computacional extra exigido em todas as etapas da arquitetura

deve ser cuidadosamente considerado para viabilizar a execução da mesma em tempo real ou em dispositivos mais simples.

2.3.11 Considerações finais da Seção

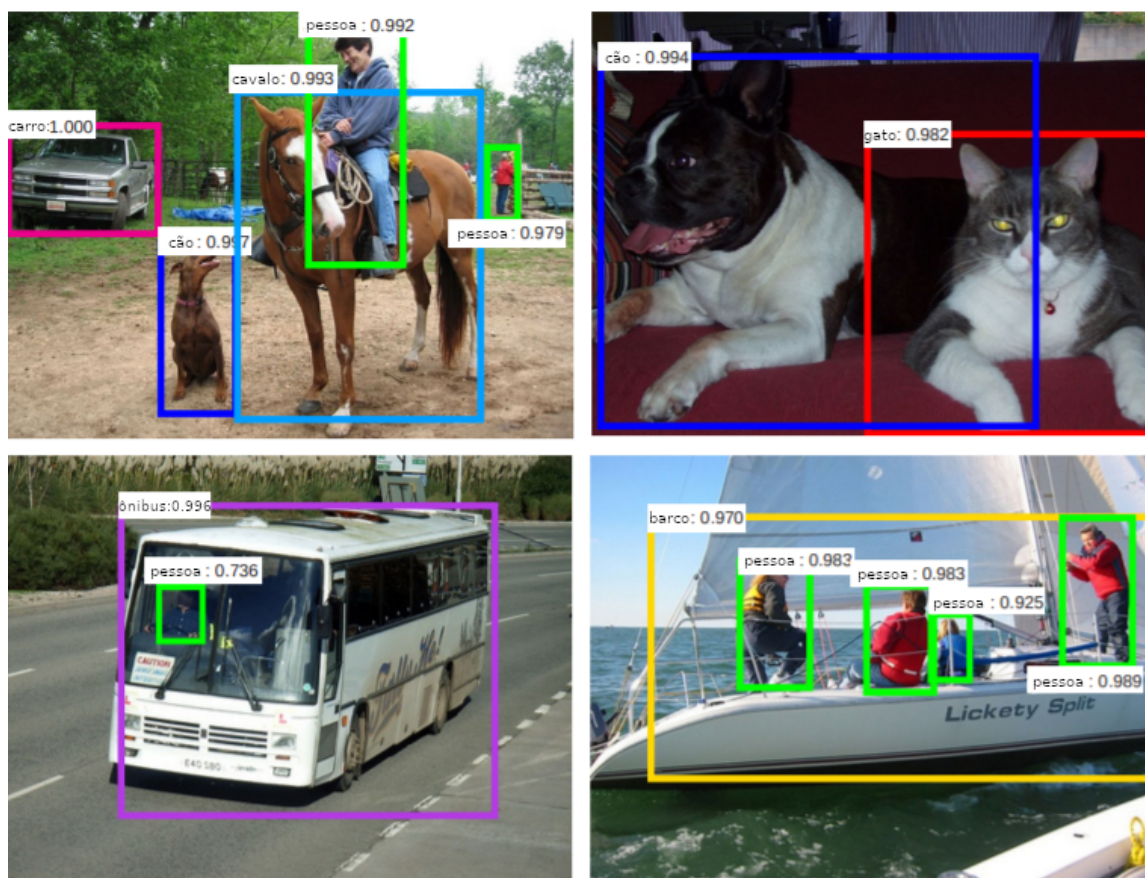
As CNNs têm sido cada vez mais adotadas no estado da arte para diferentes tarefas de reconhecimento de padrões em imagens, devido à sua capacidade de generalização e troca de aprendizado entre contextos similares. Conforme explorado nesta seção, diversas arquiteturas foram criadas com diferentes propósitos e funcionamentos. Enquanto todas elas conseguem extrair características espaciais, algumas são especializadas na dimensão temporal, como as CNNs 3D ou 2+1D. Além disso, combinações com diferentes canais, transferência de aprendizado e aumento de dados são técnicas que têm permitido o treinamento de classificadores robustos mesmo com um conjunto de dados de menor escala. Ainda assim, é importante considerar a complexidade de se treinar determinados modelos do gênero, não só por conta dos requisitos computacionais mas também por conta do *overfitting* e dos possíveis vieses presentes nas imagens. Baseado nisso, as seções subsequentes exploram os detectores de objetos e os estimadores de pose corporal, sendo técnicas utilizadas para auxiliar no processo de extração de características das imagens e facilitar o treinamento das CNNs.

2.4 Detecção de objetos

A detecção de objetos é uma técnica utilizada quando o objetivo da tarefa não é apenas classificar uma imagem de acordo com a presença ou não de um determinado objeto, mas também localizar a posição espacial do mesmo na cena. Dentre os casos de uso, essa abordagem costuma ser empregada quando mais de uma classe aparece simultaneamente na imagem, sendo necessário distinguir cada uma delas através da localização ou quando se pretende segmentar o objeto de interesse, permitindo manipulá-lo de maneira individual. As caixas delimitadoras são o nome utilizado para identificar as formas geométricas coloridas que são utilizadas para localizar os objetos na imagem. Além de indicar o posicionamento, as caixas também acompanham o nome da classe em questão e o valor de confiança da previsão gerada, conforme ilustrado na Figura 2.27.

Dito isto, o objetivo desta seção é explorar detalhes relativos a este tipo de modelo. A Seção 2.4.1 detalha a métrica de avaliação comum para detectores de objetos. A Seção 2.4.2, por sua vez, expõe algumas das principais arquiteturas presentes na literatura da área. Por fim, a Seção

Figura 2.27: Exemplo do uso da detecção de objetos em diferentes cenas.



Fonte: Adaptado de Ren et al. (2015).

2.4.3 aborda as considerações finais.

2.4.1 Avaliação dos detectores de objetos

A métrica de avaliação mais comum na área de detecção de objetos é a média da precisão média ou *mean Average Precision* (mAP). Para entender como a métrica é calculada e o motivo de ser utilizada, é preciso definir de antemão o que é considerado uma previsão correta e incorreta na detecção de objetos.

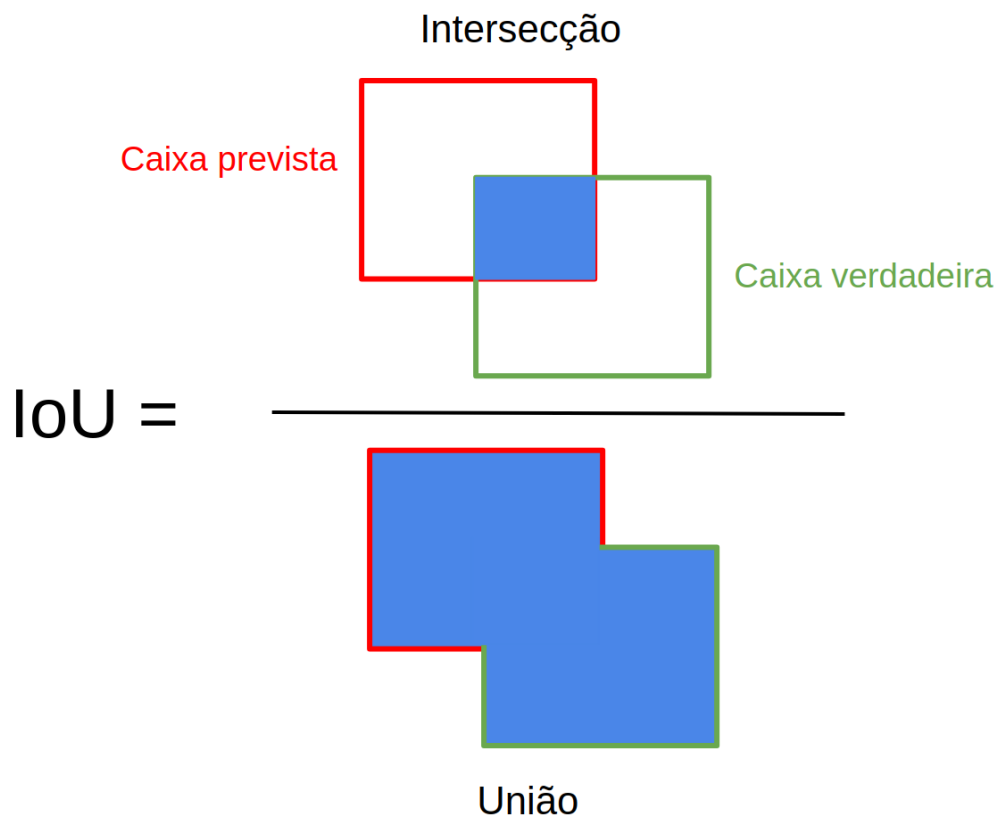
Para isso, o cálculo da intersecção sobre união, ou *Intersection Over Union* (IOU), também conhecida como índice de Jaccard, é utilizado a fim de estabelecer uma pontuação para a caixa delimitadora que foi inferida pelo modelo. Essa pontuação mensura o quão corretamente posicionada a caixa delimitadora prevista está em relação à caixa verdadeira do objeto, que é criada manualmente na anotação do conjunto de dados para possibilitar o treinamento do modelo. A IOU pode ser definida pela Equação 2.45 (Everingham et al., 2010):

$$IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}, \quad (2.45)$$

em que B_p representa a caixa delimitadora prevista pelo modelo, enquanto B_{gt} é a caixa verdadeira. A IOU também é ilustrada na Figura 2.28. Por fim, o valor de IOU é submetido a um limiar, geralmente de 50% (Everingham et al., 2010), que define se a inferência está correta ou não. Limiares mais rígidos de 75% e 95% também são propostos em trabalhos recentes, tornando-se um importante critério para avaliar o desempenho dos modelos, visto que maiores índices de IOU indicam a precisão do modelo em localizar os objetos na imagem. Com isso, as métricas base para detecção de objetos podem ser definidas da seguinte forma:

- Verdadeiro Positivo (VP): Detecção correta ($IOU \geq \text{limiar}$).
- Falso Positivo (FP): Detecção incorreta ($IOU \leq \text{limiar}$).
- Falso Negativo (FN): Objeto não detectado.

Figura 2.28: Exemplo de intersecção sobre união.



Fonte: Autor.

As métricas descritas acima são utilizadas para calcular a precisão, que indica o número de objetos localizados corretamente dentre todas as inferências do modelo, e a revocação, que exibe o percentual de objetos localizados pelo modelo em relação a todos os objetos presentes na cena, definidos pelas Equações 2.46 e 2.47, respectivamente:

$$p = \frac{VP}{VP + FP}. \quad (2.46)$$

$$r = \frac{VP}{VP + FN}. \quad (2.47)$$

Feito isto, a curva de precisão e revocação (Géron, 2017) é criada através da ordenação de todas as inferências feitas pelo modelo de maneira decrescente baseado em sua confiança. Por fim, a precisão média, ou *Average Precision* (AP) é calculada para sumarizar o formato da curva, considerando a média de 11 pontos de revocação igualmente espaçados conforme definido pela Equação 2.48 (Everingham et al., 2010):

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \dots, 1\}} P_{interp}(r), \quad (2.48)$$

onde a precisão para cada nível de revocação r é interpolada para atenuar o impacto gerado pelos desvios na ordenação das classificações, considerando apenas o valor máximo de precisão mensurada para a revocação que excede r , conforme exibido na Equação 2.49 (Everingham et al., 2010):

$$P_{interp}(r) = \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r}), \quad (2.49)$$

em que $p(\tilde{r})$ representa a precisão no ponto de revocação $\tilde{r}$. A métrica de AP força com que o modelo seja preciso em todos os níveis de revocação, penalizando métodos que são precisos apenas em um subconjunto de exemplos (Everingham et al., 2010).

Finalmente, embora utilizado de maneira intercambiável em alguns trabalhos, o mAP diz respeito a média de AP de todas as classes presentes no conjunto de dados, servindo como um valor único para representar o desempenho do detector de objetos.

2.4.2 Principais arquiteturas de detecção de objetos

Diversas arquiteturas de detecção de objetos têm sido propostas na literatura a fim de melhorar os resultados da tarefa e a tornar mais simples e barata computacionalmente. Dentre elas, os principais destaques são: Faster-RCNN (Ren et al., 2015), *Single-Shot Multibox Detector* (SSD) (Liu et al., 2016b), *You Only Look Once* (YOLO) (Redmon et al., 2016), CenterNet (Zhou et al., 2019) e a EfficientDet (Tan et al., 2020), sendo que cada uma possui uma série de vantagens e desvantagens que devem ser consideradas para a escolha da arquitetura na resolução da tarefa.

A Faster-RCNN, por exemplo, é um método baseado em algoritmos de proposta de regiões que conseguiu atingir resultados estado da arte e uma maior velocidade de execução do que os métodos anteriores, como o Fast-RCNN (Girshick, 2015), chegando a até 5 FPS de inferência utilizando redes convolucionais profundas. A arquitetura proposta pelos autores se baseia em três principais etapas, sendo a extração de características, a rede de proposta de regiões e a classificação.

A extração de características, assim como em um problema de classificação, utiliza uma arquitetura de CNN para gerar o mapa de características da imagem de entrada, evidenciando as regiões com maior incidência de similaridade a um determinado objeto que se pretende detectar. Feito isso, a rede de proposta de regiões é adotada por meio de uma janela deslizante sobre o último mapa de características da CNN que converte, através de uma rede neural, cada região da imagem em um vetor de menor dimensionalidade, utilizando como referência 9 caixas âncoras. As chamadas caixas âncoras são figuras geométricas de diferentes escalas e proporções que são adotadas na tentativa de melhor encaixar a região do objeto, de maneira invariante à translação do mesmo. Além disso, cada vetor gerado utilizando as caixas supracitadas serve como entrada para um classificador, que prevê a probabilidade do segmento da imagem conter ou não um objeto, e um regressor, que ajusta as coordenadas das caixas delimitadoras das regiões de proposta.

Por fim, a técnica de supressão não máxima, ou NMS (*non maximum suppression*) é utilizada para reduzir o número de regiões candidatas, removendo áreas de baixa confiança e com muita intersecção. Ademais, as regiões propostas são submetidas a uma rede de detecção de objetos baseada no Fast-RCNN, que também aplica uma operação de classificação e regressão, definindo a classe do objeto presente na região de interesse e gerando mais ajustes às caixas delimitadoras.

Embora os autores tenham conseguido bons resultados, chegando a até 70,4% de mAP na

base do desafio PASCAL VOC 2012 (*Pattern Analysis, Statistical Modelling and Computational Learning Visual Object Classes*) (Everingham et al., 2010), o custo computacional e os requisitos de memória da arquitetura são altos (Huang et al., 2017), não só por conta do extrator de características baseado em uma rede VGG com 16 camadas (Simonyan e Zisserman, 2015), mas também pelo fato da arquitetura ser dividida em dois estágios, um para a rede de propostas de regiões e outro para a classificação e regressão das propostas.

Por conta disso, detectores mais recentes propõem uma única iteração com a imagem para extrair toda a informação necessária para o detector, descartando as redes de proposta de regiões. A YOLO é uma delas, tornando mais simples o processo de treinamento e inferência do modelo e sendo capaz de atingir até 45 FPS na tarefa de detecção.

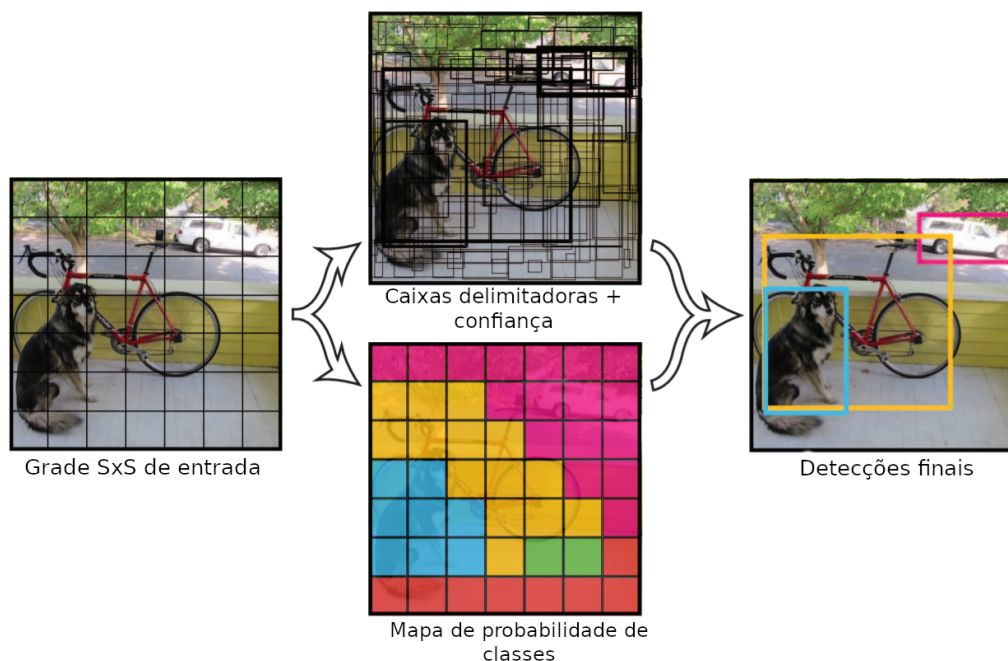
Para que isso seja possível, os autores criaram uma CNN baseada na GoogleLeNet (Szegedy et al., 2015) composta por 24 camadas convolucionais para extrair as características das imagens, e duas camadas completamente conectadas, responsáveis por prever a probabilidade das classes e as coordenadas das caixas delimitadoras.

Além disso, a arquitetura YOLO divide a imagem de entrada em uma grade de dimensões $S \times S$ e, para cada célula da grade, prevê B caixas delimitadoras junto a uma pontuação de confiança, que reflete o quão confiante o modelo está da coordenada da caixa e também da mesma conter o objeto. Por fim, cada célula recebe um valor C de probabilidade condicional de conter uma classe específica, gerando um mapa de probabilidade de cada uma das classes. O mapa das classes e as caixas delimitadoras são combinadas para gerar a detecção final do objeto, junto ao método de NMS para remover as caixas excedentes, conforme ilustrado na Figura 2.29.

Ainda que eficiente, a YOLO possui algumas limitações que podem prejudicar a tarefa de detecção de objetos (Redmon et al., 2016), como dificuldades em detectar objetos de menor proporção e que apareçam muito próximos uns dos outros na cena, devido à quantidade de caixas delimitadoras por célula. Além disso, como a proporção das caixas delimitadoras é aprendida durante o treinamento e caixas âncoras não são utilizadas, o objeto pode não ser detectado caso apareça em proporções inusitadas. A YOLO atingiu 57,9% de mAP no conjunto PASCAL VOC 2012, mais de 10% inferior aos resultados obtidos pelo Faster R-CNN.

O SSD, por sua vez, é capaz de executar a tarefa de detecção em até 59 FPS e com menores requisitos de memória do que a Faster-RCNN (Huang et al., 2017), enquanto mantém uma alta acurácia. Para atingir tais resultados, os autores utilizam a mesma CNN base do Faster-RCNN como extrator de características, a VGG-16, mas adicionam novas camadas de convolução ao

Figura 2.29: Arquitetura de detecção de objetos YOLO.



Fonte: Adaptado de Redmon et al. (2016).

fim da CNN para criar mapas de características de diferentes resoluções. Feito isso, cada novo mapa de características gerado, além do último mapa da própria CNN base, são submetidos a 6 caixas âncoras com diferentes escalas e proporções que geram uma pontuação para a presença ou não de determinada classe na região e calculam as coordenadas da caixa delimitadora considerando seu formato padrão.

Com essa abordagem, a arquitetura do SSD consegue bons resultados mesmo em imagens de menor resolução, com 300x300 *pixels*, visto que diferentes mapas de características são explorados, facilitando a captura de detalhes sutis na imagem. Por fim, a técnica de NMS é utilizada para remover todos os candidatos com intersecção ou baixa confiança e gerar as previsões finais das caixas delimitadoras dos objetos. Além da possibilidade de executar em tempo real, o SSD obteve 4,5% mais mAP do que o Faster-RCNN na base PASCAL VOC 2012.

A CenterNet, por outro lado, adota uma abordagem mais simples, porém efetiva, para detectar objetos. Nessa arquitetura, cada objeto presente na cena é tratado como um ponto, representando o centroide de sua caixa delimitadora. Uma vez que o ponto central do objeto é detectado, as dimensões da caixa delimitadora são estimadas a partir de um regressor, fazendo com que o problema de detecção de objetos se assimile a um problema de estimativa de pontos chaves (*keypoints*).

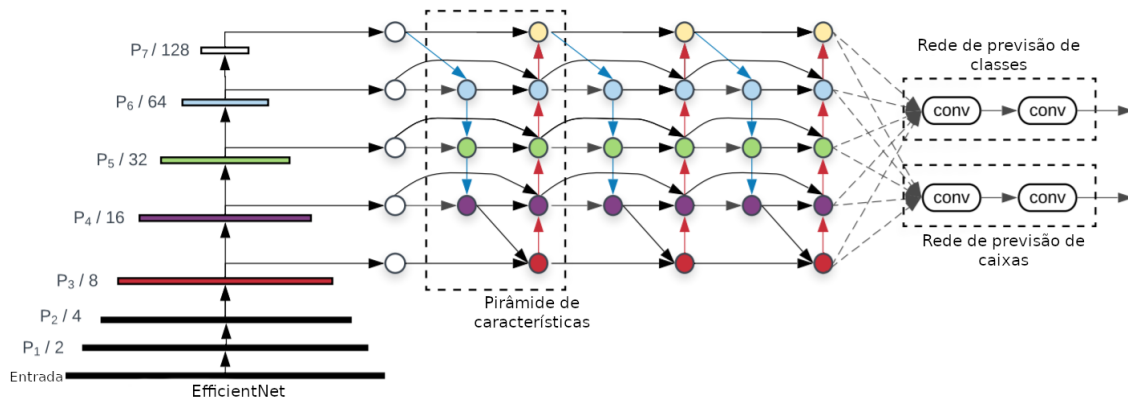
Assim como nas demais arquiteturas, a imagem de entrada é submetida a uma CNN para extração de características. Porém, neste caso, ao invés do uso de caixas âncora, a janela deslizante é responsável apenas por estimar a existência ou não de um *keypoint* de uma determinada classe. Após estimar os pontos chave por toda a imagem de cada um dos objetos, um mapa de calor é gerado a fim de extrair os picos de inferência, considerando uma vizinhança de 8 *pixels*. Por fim, cada um dos picos são considerados como um centroide do objeto e um regressor é utilizado para estimar as medidas exatas das caixas delimitadoras, sem precisar de técnicas adicionais de pós processamento como o NMS.

Por conta de sua abordagem, a CenterNet possui um baixo custo computacional e menos hiperparâmetros para serem ajustados, executado em até 52 FPS enquanto atinge resultados comparáveis aos demais modelos do estado da arte. Ainda assim, este modelo possui limitações quanto a cenas muito populadas e falha em casos de colisão de centroides, podendo não ser adequado para todos os problemas.

Por fim, a EfficientDet é uma das arquiteturas criadas recentemente que mais se destaca por conseguir resultados similares ao estado da arte enquanto mantém o custo computacional baixo. Para isso, os autores utilizam a EfficientNet (Tan e Le, 2019) como CNN base, bem como uma rede de pirâmide de características bidirecional com múltiplas iterações. As pirâmides de características (Lin et al., 2017) são técnicas utilizadas em detectores recentes para melhorar a capacidade de detecção de objetos pequenos ou até de objetos muito próximos uns dos outros. Para isso, este algoritmo combina os mapas de características em estágios iniciais da rede base, que possuem maior resolução de imagem mas pouco valor semântico, com mapas de características dos estágios finais da CNN, com baixa resolução mas alto valor semântico. Dessa forma, os mapas de características resultantes são uma versão reconstruída da combinação de múltiplas escalas, agregando maior resolução e valor semântico à saída da CNN. As características unidas em múltiplas escalas são então utilizadas como entrada para uma rede que produz as caixas delimitadoras e classifica os objetos, conforme ilustrado na Figura 2.30.

Além disso, os autores também destacam o uso do método de escalabilidade composta, sendo adotado para otimizar os hiperparâmetros do detector de objetos e conseguir o melhor equilíbrio entre custo computacional e acurácia. Diferente de trabalhos anteriores, as dimensões da CNN base, da rede de pirâmide de características, do detector de caixas delimitadoras/classes e a resolução da imagem de entrada são escalonados simultaneamente. Os resultados obtidos evidenciam que a arquitetura treinada e otimizada é capaz não só de obter melhores resultados

Figura 2.30: Arquitetura de detecção de objetos EfficientDet.



Fonte: Adaptado de [Tan et al. \(2020\)](#).

do que outras arquiteturas modernas, mas também reduz o número de parâmetros e a latência na inferência de imagens.

2.4.3 Considerações finais da seção

Diversas arquiteturas têm sido criadas nos últimos anos para melhorar os resultados da tarefa de detecção de objetos, utilizadas como base por inúmeros sistemas embarcados ou como pré-processamento para segmentação de locais da imagem que serão utilizados em outras tarefas de classificação. Levando isto em consideração, o custo computacional é um fator de extrema importância para a tarefa, pois impacta diretamente no uso de memória e tempo de inferência do sistema.

Ainda assim, nem todos os trabalhos apresentam preocupações com os requisitos computacionais ([Ren et al., 2015](#)) ou acabam prejudicando a performance do modelo em razão dos mesmos ([Redmon et al., 2016](#)). Portanto, arquiteturas que visam reduzir o número de parâmetros através da detecção de objetos em um único estágio, como o SDD, e que otimizem a extração e aproveitamento de características relevantes, como a EfficientDet, são mais vantajosas por conseguir bons resultados, enquanto mantêm um modelo simples e barato. Na Seção 2.5, a detecção de poses corporais será exibida em maiores detalhes, sendo uma técnica alternativa muito utilizada para segmentação de partes da imagem.

2.5 Estimativa de pose corporal

A estimativa de pose corporal é outra técnica que tem ganho muito destaque recentemente, motivado pela criação de novas arquiteturas que permitem realizar a tarefa de maneira efetiva.

O objetivo do método, conforme o nome sugere, é criar uma representação para o corpo humano através da detecção de pontos articulares e de sua consequente ligação com os demais membros do corpo, a partir de uma imagem RGB bidimensional. Os membros são representados através de vértices e arestas, também chamadas de esqueleto, e são conectados de maneira a formar um grafo, o qual pode ser utilizado como entrada para modelos de aprendizado de máquina que podem realizar a classificação de poses, gestos e até sinais em língua de sinais. Além disso, de maneira similar ao que acontece com a detecção de objetos, as regiões do esqueleto podem ser adotadas para referenciar e segmentar partes do corpo da imagem, como as mãos e o rosto, servindo para criação de dicas visuais, conforme discutido na Seção 3.2. A Figura 2.31 ilustra o resultado da inferência de um modelo de poses corporais.

Figura 2.31: Resultados da inferência de poses corporais em diferentes cenários e situações.



Fonte: Cao et al. (2019).

Dentre os modelos presentes na literatura, destaca-se o uso das abordagens *top-down* e *bottom-up*, com suas respectivas vantagens e desvantagens, sendo discutidas em mais detalhes nas seções subsequentes.

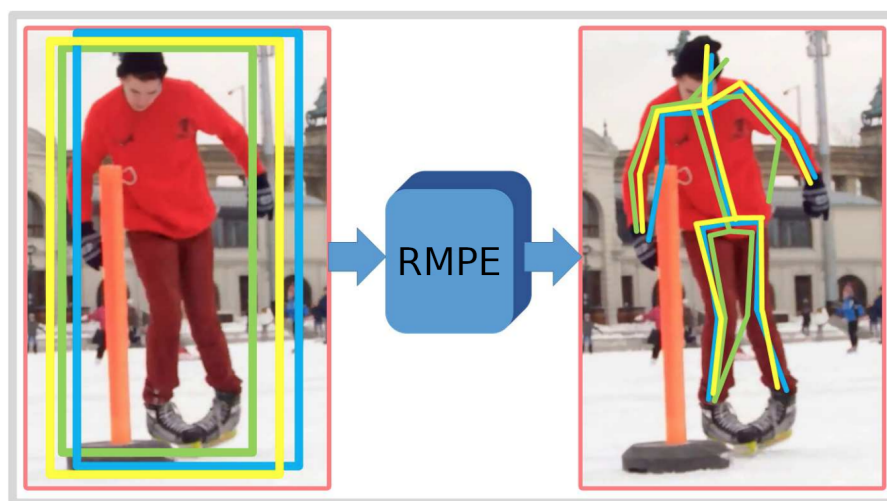
2.5.1 Abordagem *top-down*

A abordagem conhecida como *top-down* emprega um detector de objetos na etapa de pré-processamento da arquitetura, sendo responsável por localizar todas as pessoas presentes na imagem e estimar as partes do corpo dentro de cada caixa delimitadora de maneira independente. A principal desvantagem dessa técnica é a dependência do sucesso do detector de objetos

em gerar previsões acuradas, o que pode ser especialmente complexo em cenas muito popladas (Jin et al., 2017).

Por conta disso, arquiteturas como a RMPE (*Regional Multi-person Pose Estimation*) (Fang et al., 2017) têm se destacado nessa modalidade por empregar inúmeros métodos para extrair poses precisas, mesmo com caixas delimitadoras pouco confiáveis. Para tal, os autores adotaram uma rede de transformação espacial, ou STN (*spatial transformer network*) (Jaderberg et al., 2015), responsável por extrair a região de interesse com a maior proporção ocupada por um ser humano, atenuando o efeito de detecções imprecisas. Feito isso, o extrator de poses é empregado para cada caixa delimitadora com proposta de regiões, além de um regularizador paralelo para corrigir a posição espacial das previsões geradas. Por fim, uma técnica de *Non-Maximum Suppression* (NMS) é utilizada a fim de remover todas as poses redundantes geradas pelo processo, visto que a pessoa pode ser detectada mais de uma vez na imagem e ter múltiplas inferências, conforme ilustrado na Figura 2.32.

Figura 2.32: Múltiplas poses estimadas por problemas na detecção de objetos.



Fonte: Adaptado de Fang et al. (2017).

Dessa forma, a pose com maior confiança pelo modelo é selecionada como referência, e todas as demais poses muito próximas a ela e com muita similaridade (baseado em um cálculo de distância) são recursivamente eliminadas, mantendo apenas uma única inferência.

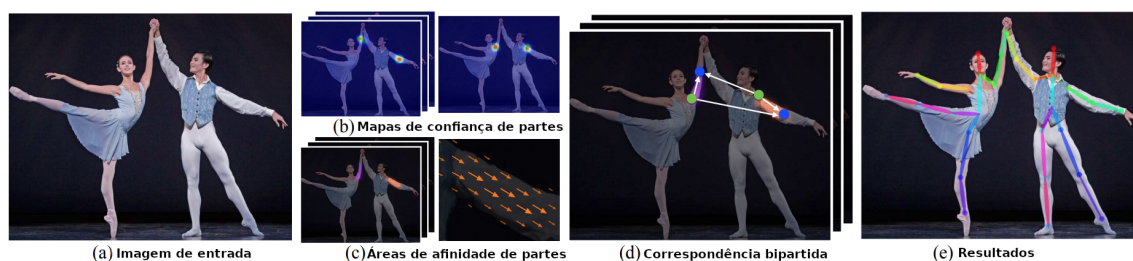
Os resultados reportados pelos autores evidenciam o desempenho estado da arte da arquitetura criada, atingindo até 3,3 mais mAP do que trabalhos anteriores. Ainda assim, abordagens do tipo *top-down* crescem em custo computacional proporcionalmente ao número de previsões geradas pelo detector de objetos, podendo ser inviável para certos tipos de problemas.

2.5.2 Abordagem *bottom-up*

A abordagem *bottom-up*, por outro lado, gera primeiro a estimativa de partes do corpo em toda a imagem e depois agrupa as partes pertencentes à mesma pessoa, gerando as poses individuais. A desvantagem dessa técnica se encontra principalmente quando dois ou mais indivíduos estão muito próximos uns dos outros, gerando ambiguidades nas previsões (Fang et al., 2017).

A arquitetura chamada OpenPose (Cao et al., 2019) é uma das mais utilizadas da área, por obter melhores resultados e menor custo computacional do que as técnicas anteriores. Para que isso seja possível, diversas etapas são aplicadas na imagem para extrair a pose de todas as pessoas presentes na cena simultaneamente, fazendo com que o sistema seja eficiente mesmo que a inferência seja realizada em múltiplos indivíduos. A Figura 2.33 resume a arquitetura supracitada.

Figura 2.33: Arquitetura do OpenPose.



Fonte: Adaptado de Cao et al. (2019).

O primeiro passo para seu funcionamento ocorre através da aplicação da imagem de entrada a uma CNN do tipo VGG, gerando mapas de características que são utilizados nas fases subsequentes. Feito isso, uma arquitetura multicanais e multiestágios é empregada a fim de extrair os mapas de confiança de partes (onde “parte” refere-se aos pontos chaves anatômicos que formam o esqueleto) e os campos de afinidade de partes, sendo que cada parte do corpo possui seu respectivo mapa de confiança e cada membro (pares de partes) possui um vetor de afinidade, que codifica o grau de associação entre duas partes do corpo. Dessa forma, cada canal da arquitetura extrai um tipo de informação da imagem, enquanto cada estágio é empregado para refinar as previsões geradas.

Uma vez que os candidatos com maior confiança de cada uma das partes são previstos, as áreas de afinidade são utilizadas para inferir o posicionamento e orientação dos pares de partes, sendo fundamental para gerar uma medida de confiança de associação entre os membros de indivíduos distintos. Por fim, a correspondência bipartida é empregada como uma etapa

do processo de supressão não máxima, a fim de remover os pontos inferidos incorretamente, gerando combinações precisas entre os membros.

Conforme reportado pelos autores, a arquitetura desenvolvida conseguiu resultados estado da arte na inferência de poses corporais, além de atingir um tempo de execução inferior a outros métodos, especialmente em cenas muito populadas.

2.5.3 Considerações finais da seção

Diversas abordagens e arquiteturas têm sido desenvolvidas recentemente a fim de melhorar os resultados na tarefa de inferência de poses, destacando-se a *top-down* e a *bottom-up*, com suas respectivas vantagens e desvantagens. Ainda que o custo computacional desta técnica seja superior à da detecção de objetos, visto que além de detectar a pessoa ou as partes do corpo na imagem também é necessário unir os vértices correlatos a fim de gerar o esqueleto, muitos trabalhos têm se beneficiado de sua adoção, por permitir representar o corpo humano de maneira simples e gerar segmentações na imagem original. Dessa forma, a Seção 3.2.1.1 do capítulo subsequente destaca a aplicação de tais metodologias na área do reconhecimento de sinais, sendo amplamente utilizadas.

2.6 Considerações finais do capítulo

Neste capítulo, os principais aspectos teóricos a respeito das diferentes arquiteturas de redes neurais e do aprendizado profundo foram abordados. Conforme discutido, as CNNs são técnicas cada vez mais populares por obter resultados estado da arte em diferentes tarefas de reconhecimento de padrões em imagens, capazes de extrair características espaço-temporais utilizadas para classificação. Ainda assim, métodos auxiliares como a detecção de objetos e a extração de poses corporais são importantes por facilitar o foco em regiões de interesse ou extrair informações relevantes da imagem que auxiliem a tarefa dos extratores de características, fazendo com que arquiteturas menos profundas ou complexas possam ser utilizadas para classificação, facilitando o treinamento e reduzindo os requisitos computacionais. Dito isto, o Capítulo 3 tem por objetivo explorar e comparar alguns dos principais trabalhos da literatura de reconhecimento de sinais, evidenciando as técnicas e metodologias utilizadas para a extração de características e classificação, bem como as vantagens e desvantagens de cada uma delas.

3 Reconhecimento de língua de sinais com aprendizado profundo

Embora a tarefa de reconhecimento de sinais possua inúmeros trabalhos na literatura, apenas uma parcela deles se atentam aos critérios impostos nesta pesquisa, com o objetivo de garantir uma comparação mais justa e assertiva entre as soluções propostas. Primeiro, é importante delimitar as diferenças entre os dois principais segmentos desta área de pesquisa, sendo o reconhecimento contínuo de sinais e o reconhecimento a nível de palavras, ou isolado, como também é conhecido. Embora as duas subáreas compartilhem a tarefa de reconhecer um sinal através da extração de características de uma sequência de imagens, a modalidade contínua tem por objetivo utilizar gravações ininterruptas de longa duração, contendo inúmeras frases e diálogos e com um alinhamento temporal não tão preciso entre as anotações das classes e o momento da execução do sinal.

Um dos conjuntos de dados mais utilizados nessa subárea, por exemplo, o RWTH-PHEONIX-WEATHER (Forster et al., 2012) é composto por mais de 3 horas de gravações de um programa meteorológico de uma estação televisiva alemã, onde o significado de cada sinal executado na gravação foi anotado de maneira semi-supervisionada através do reconhecimento da fala dita pelo apresentador do programa. Desafios como alinhamento temporal e semântico entre a legenda produzida pelo modelo e o vídeo processado, memória a longo prazo para contextualização da fala e segmentação de início e fim das frases são comuns e muito explorados para o reconhecimento contínuo de sinais, sendo que o BLEU (*Bilingual Evaluation Understudy*) (Papineni et al., 2002) é a métrica de avaliação adotada para essa tarefa.

O reconhecimento de sinais isolados, por outro lado, possui o conjunto de dados segmentado em diversos vídeos de pequena duração, onde cada vídeo representa uma única palavra/classe. Desta forma, a tarefa se limita ao escopo de extrair as características do sinal e classificar corretamente o maior número de palavras possível, sendo a acurácia (vista na Seção 2.1.7) a métrica mais comum de avaliação dessa subárea. Portanto, é importante ponderar que, ainda que o foco deste trabalho seja o reconhecimento de sinais isolados, as técnicas exploradas para melhorar a eficiência de classificação também podem ser aplicadas ao contexto do reconhecimento contínuo em trabalhos futuros.

Ademais, esta pesquisa também foca em trabalhos que se utilizam de sensores RGB, facilitando a implantação da solução criada, conforme discutido na Seção 1.2, e que capturem todos

os parâmetros linguísticos, evitando a negligência de informações relevantes para a classificação dos sinais.

Dito isso, a Seção 3.1 apresenta os diferentes conjuntos de dados disponíveis publicamente para o reconhecimento de sinais isolados, bem como as técnicas mais comumente utilizadas na criação dos mesmos. A Seção 3.2 apresenta os trabalhos da literatura que utilizam de modelos de aprendizado profundo para a tarefa de reconhecimento de sinais, além de agrupá-los pelo tipo de pré-processamento adotado. Por fim, a Seção 3.3 apresenta considerações finais do capítulo.

3.1 Conjuntos de dados para o reconhecimento de sinais isolados

Conforme discutido nos Capítulos 1 e 2, muito dos trabalhos recentes no reconhecimento de sinais são baseados em CNNs, sendo uma abordagem vantajosa por conseguir lidar com bases de dados complexas, dada a sua capacidade de generalização, e pela possibilidade de reutilização desses modelos em problemas de domínio similar, através da transferência de aprendizado (discutida na Seção 2.3.7).

Por outro lado, a grande quantidade de dados exigida para o treinamento desses modelos pode se tornar uma barreira na área, dada a dificuldade na criação de conjuntos de dados representativos. A LIBRAS, por exemplo, possui mais de 10 mil verbetes (de Castro et al., 2019), e uma parte considerável desses sinais precisa ser gravada e utilizada no treinamento de um modelo para que o sistema de tradução consiga, de fato, trazer acessibilidade ao dia a dia dos deficientes auditivos. Pensando nisto, esta Seção visa documentar as principais formas de criação de conjuntos de dados utilizadas na literatura, bem como os desafios de cada uma delas, fazendo um levantamento das principais bases disponíveis abertamente. A Seção 3.1.1 discorre a respeito da coleta de dados realizada de maneira manual pelos seus criadores. A Seção 3.1.2, por outro lado, discorre sobre uma abordagem baseada em mineração de dados para criação de conjuntos em larga escala utilizando conteúdo disponível na *internet*. Finalmente, a Seção 3.1.3 apresenta as considerações finais da seção.

3.1.1 Coleta manual de dados

A coleta manual de dados para criação de conjuntos de sinais nessa área costuma ser muito onerosa, pois além de envolver a gravação de inúmeros sinais distintos de maneira repetitiva, também necessita de diversos intérpretes ou surdos que sejam aptos a realizar os sinais corretamente. Além disso, conforme evidenciado na pesquisa de de Castro et al. (2019), a falta de

uma metodologia de gravação ou conjunto de dados padronizado para uma determinada língua de sinais torna impraticável o comparativo entre trabalhos distintos, visto que a base de dados tem grande influência nos resultados obtidos pelo método (Maruyama et al., 2021) e que nem sempre a mesma se encontra disponível publicamente.

Dentre os conjuntos de dados criados nesta categoria, um dos maiores na quantidade de sinais documentados é o ASLLVD (*American Sign Language Lexicon Video Dataset*) (Neidle et al., 2012), contendo mais de 2793 palavras distintas na língua americana de sinais, executadas por 6 intérpretes. Ainda assim, a pequena quantidade de pessoas e repetições executadas por sinal (variando de 1 a 6, totalizando 9794 vídeos) podem se tornar vieses significativos para os resultados, considerando o número de sinais propostos.

Outra alternativa que possui um vasto vocabulário é o conjunto Devisign (Chai et al., 2014), com 2000 palavras executadas por 8 intérpretes, totalizando 24000 vídeos. Ainda assim, a base possui os mesmos problemas relacionados à pequena quantidade de intérpretes e repetições do sinal (que variam de 1 a 2), podendo dificultar o treinamento e a generalização de uma CNN.

Por conta disto, trabalhos recentes têm surgido com o objetivo de criar conjuntos de dados que possam ser utilizados como base para diferentes estudos na área. Um dos exemplos disso é o conjunto da língua turca de sinais chamado AUTSL (*Ankara University Turkish Sign Language*) (Sincan e Keles, 2020), contendo 226 sinais e 43 intérpretes em 20 cenários distintos gravados em ambiente real. O conjunto de dados foi base para a competição ChaLearn (Sincan et al., 2021) de aprendizado de máquina, sendo explorada por diversas pesquisas, conforme exibido na Seção 3.2.

O BosphorusSign22k (Özdemir et al., 2020) é outro exemplo de conjunto de dados criado com o objetivo de servir como base para futuras pesquisas, possuindo um protocolo de avaliação detalhado que promove o comparativo entre métodos distintos. O conjunto em questão é derivado de uma versão anterior, o BosphorusSign (Camgöz et al., 2016), e contém 744 sinais da língua turca, gravados por 6 intérpretes. Embora o plano de fundo seja estático, o uso de uma tela verde padronizada (chamada de *Chroma Key*) facilita a substituição artificial do mesmo por diferentes cenários.

A base de dados CSL (*Chinese Sign Language*) (Pu et al., 2016), por sua vez, conta com 50 intérpretes e até 125000 vídeos para documentar 500 sinais distintos da língua chinesa. Mesmo com a maior diversidade de indivíduos, as gravações apresentam condições laboratoriais e contam com apenas um único plano de fundo estático, podendo prejudicar o sistema de

reconhecimento ao tentar classificar sinais em condições mais adversas.

Ainda na língua chinesa de sinais, podemos citar as bases NCSL (Wang et al., 2021) e NMFS-CSL (*Non-Manual Features Chinese Sign Language* (Hu et al., 2021), as quais possuem 300 sinais executados por 30 intérpretes e 1.067 sinais executados por 10 intérpretes, respectivamente, ambas filmadas em condições laboratoriais semelhantes às do conjunto CSL. Entretanto, a base NMFS-CSL se destaca por possuir uma grande variedade de sinais similares, diferenciando-se principalmente pelas expressões faciais, tornando a tarefa de classificação mais complexa.

Por fim, outras bases de menor relevância também podem ser citadas, como o INCLUDE (Sridhar et al., 2020) da língua indiana de sinais que possui apenas 7 intérpretes e condições padronizadas de plano de fundo, o LSA64 (Língua de Sinais Argentina) (Ronchetti et al., 2016) em que os intérpretes usam luvas coloridas nas mãos, gerando uma condição que dificilmente seria reproduzível fora de ambiente laboratorial e o SMILE (Ebling et al., 2018) que apresenta pouca variedade de sinais, filmados em um fundo verde.

Deste modo, é possível notar que a complexidade de se criar um conjunto de dados de língua de sinais se reflete em limitações no número de palavras, intérpretes ou planos de fundo, visto que além da necessidade de recurso humano, também é preciso possuir equipamento adequado e, muitas das vezes, *softwares* específicos (Neidle et al., 2012, Ebling et al., 2018) para auxiliar na gravação e anotação das classes.

A Tabela 1 sumariza todos os conjuntos de dados criados manualmente e disponíveis publicamente com pelo menos 5 intérpretes e 50 sinais, visto que conjuntos com uma dimensão menor que a supracitada foram considerados muito simples para a tarefa de reconhecimento de sinais, uma vez que bases menores tendem a omitir sinais mais complexos como os polissêmicos, por exemplo, conforme explicados na Seção 1.1.

3.1.2 Criação de bases de dados através da mineração de conteúdo

Uma abordagem alternativa para criação de conjuntos de dados de língua de sinais é a mineração de conteúdo em plataformas públicas, que além de ser mais rápido e barato do que o método anterior, também tem um grande potencial de criar bases em larga escala. Dentre as abordagens, destaca-se o processo conhecido como *web scraping*, onde algoritmos são utilizados para automatizar a captura e anotação de vídeos disponíveis em plataformas *online*.

Tabela 1: Comparação dos conjuntos de dados em larga escala (mais do que 50 sinais) criados manualmente e com pelo menos 5 intérpretes.

Conjunto de dados	N. sinais	N. Intérpretes	Língua	Vídeos
AUTSL (Sincan e Keles (2020))	226	43	Turca	38.336
BosphorusSign22k (Özdemir et al. (2020))	744	6	Turca	22.542
BosphorusSign (Camgöz et al. (2016))	855	6	Turca	22.670
ASLLVD (Neidle et al. (2012))	2.793	6	Americana	9.794
Devisign (Chai et al. (2014))	2.000	8	Americana	24.000
CSL (Pu et al. (2016))	500	50	Chinesa	125.000
NCSL (Wang et al. (2021))	300	30	Chinesa	90.000
NMFS-CSL (Hu et al. (2021))	1.067	10	Chinesa	32.010
INCLUDE (Sridhar et al. (2020))	263	7	Indiana	4.287
LSA64 (Ronchetti et al. (2016))	64	10	Argentina	3200
SMILE (Ebling et al. (2018))	100	30	Suíço-Alemã	9000

Fonte: Autor.

Joze e Koller (2019), por exemplo, criaram uma base de dados intitulada por eles de MS-ASL (*American Sign Language*), contendo mais de 20000 vídeos, 200 intérpretes e 1000 classes de sinais distintas. Os autores utilizaram de técnicas de segmentação temporal, reconhecimento ótico de caracteres e reconhecimento facial para extração e anotação dos vídeos disponíveis em sites de comunicação e estudo da Língua Americana de Sinais (LAS). Com essa abordagem, além da criação de um conjunto de dados em larga escala de forma muito mais simples e barata, também há maior diversidade de planos de fundos, condições de iluminação, posicionamento da câmera e intérpretes.

Li et al. (2020) também se destacam pelo conjunto de dados criado, o WLASL (*Word-Level American Sign Language*), contendo mais de 21000 vídeos, 119 intérpretes e 2000 palavras distintas em LAS, através da coleta de vídeos disponíveis em *websites*. Os autores utilizam diversas técnicas de pós-processamento para preparar o conjunto de dados de maneira adequada, como o uso de um detector de pessoas para aumentar o foco no corpo do intérprete e atenuar o plano de fundo, e detecção facial para garantir uma diversidade mínima de 3 intérpretes por sinal, além de separar os intérpretes do conjunto de treinamento dos de teste.

Embora a mineração seja vantajosa, é necessário criar diversas estratégias para corrigir as anotações e eliminar dados ruidosos do conjunto, sem contar que não há garantia no conteúdo obtido, visto que a cada execução do processo novos vídeos são adicionados nas fontes *online* e vídeos antigos podem acabar se perdendo por não estarem mais disponíveis publicamente. Por fim, nem todas as palavras podem ser encontradas com o número mínimo de intérpretes ou repetições necessárias, limitando a base aos sinais mais comuns do cotidiano (Li et al., 2020). A Tabela 2 compara os conjuntos de dados criados com o auxílio da metodologia supracitada.

Tabela 2: Comparação de conjuntos de sinais em larga escala criados através da mineração de conteúdo.

Conjunto de dados	N. sinais	N. Intérpretes	Língua	Vídeos
MS-ASL (Joze e Koller (2019))	1.000	200	Americana	20.000
WLASL (Li et al. (2020))	2.000	119	Americana	21.000

Fonte: Autor.

3.1.3 Considerações finais da Seção

Na Seção 3.1 foram apresentadas as principais bases de dados utilizadas em trabalhos anteriores, bem como a metodologia de sua criação. Considerando que o conjunto de dados é o alicerce para modelos modernos de aprendizado profundo, ainda mais quando se trata de problemas complexos ou com uma grande variedade de classes, a escolha do método de criação ou da base utilizada para testar a solução é primordial para o sucesso do sistema, visto que fatores como plano de fundo, número de sinais e independência ao intérprete são fundamentais para o contexto de reconhecimento de sinais, conforme discutido na Seção 3.2. A seguir, são apresentados os principais trabalhos e métodos adotados na literatura para o reconhecimento de sinais,

bem como as limitações e similaridades de cada um deles.

3.2 Trabalhos na área do reconhecimento de língua de sinais

Embora a tarefa de reconhecimento de sinais compartilhe similaridades com outras áreas, como o reconhecimento de ações em vídeos, fica claro que a estrutura semântica e os parâmetros linguísticos apresentados na Seção 1.1 têm grande influência nos trabalhos recentes desta área, principalmente no que tange ao pré-processamento do sistema e à exploração de dicas visuais.

As dicas visuais são segmentações de partes específicas do corpo do intérprete (mãos, face), as quais têm as suas características exploradas de maneira individual, permitindo que os modelos recebam informações mais relevantes da imagem e gerando melhorias nos resultados. As abordagens de pré-processamento, por sua vez, são empregadas para destacar, segmentar ou agregar partes de interesse da imagem, sendo fundamentais para a extração das dicas visuais ou para criação de novas características utilizadas na classificação dos sinais.

Desta forma, além da escolha do tipo de modelo base, as técnicas que visam melhor aproveitar as características dos parâmetros linguísticos são de suma importância para os sistemas de reconhecimento de língua de sinais. Baseado nisso, o objetivo desta Seção é apresentar os principais trabalhos presentes na literatura dos últimos 6 anos (de 2016 até 2022), considerando as pesquisas que tenham utilizado pelo menos 50 sinais distintos no conjunto de dados.

A limitação temporal das pesquisas da área ocorre para considerar as abordagens e técnicas mais recentes e avançadas de reconhecimento de padrões, inteligência artificial e pré-processamento dos sistemas criados. O limiar da quantidade de sinais, por sua vez, se dá pelo fato de que o número de classes influencia diretamente na complexidade do sistema (Li et al., 2020, Maruyama et al., 2021, Hosain et al., 2021), e um número menor que o proposto foi considerado insuficiente para representar a diversidade no vocabulário da língua de sinais.

Por fim, as pesquisas foram divididas entre as que utilizaram as CNNs como modelo base (Seção 3.2.1) e as que empregaram as redes convolucionais em grafos (Seção 3.2.2), por serem as duas abordagens mais comuns para extração de características e classificação nesta área.

3.2.1 Redes Neurais Convolucionais

Os modelos baseados em CNNs têm sido amplamente adotados como estado da arte no reconhecimento de sinais, integrando arquiteturas robustas a fatores como iluminação, cenário e intérpretes distintos. Por conta disto, esta seção visa documentar as pesquisas que se baseiam

em tais modelos para classificação de sinais, segmentando-as pelo tipo de pré-processamento adotado, uma vez que essa etapa define a forma como a imagem será tratada e transmitida ao modelo. A Seção 3.2.1.1, por exemplo, exhibe a tendência atual de extrair poses corporais por meio de esqueletos, possibilitando a exploração de informações espaciais de maneira simplificada. A Seção 3.2.1.2 agrupa os trabalhos que utilizaram detectores de objeto para focar em locais de maior importância na imagem. Por fim, a Seção 3.2.1.3 mostra outras técnicas diversas utilizadas na área com a finalidade de melhorar os resultados dos sistemas criados.

3.2.1.1 Extração de pose corporal

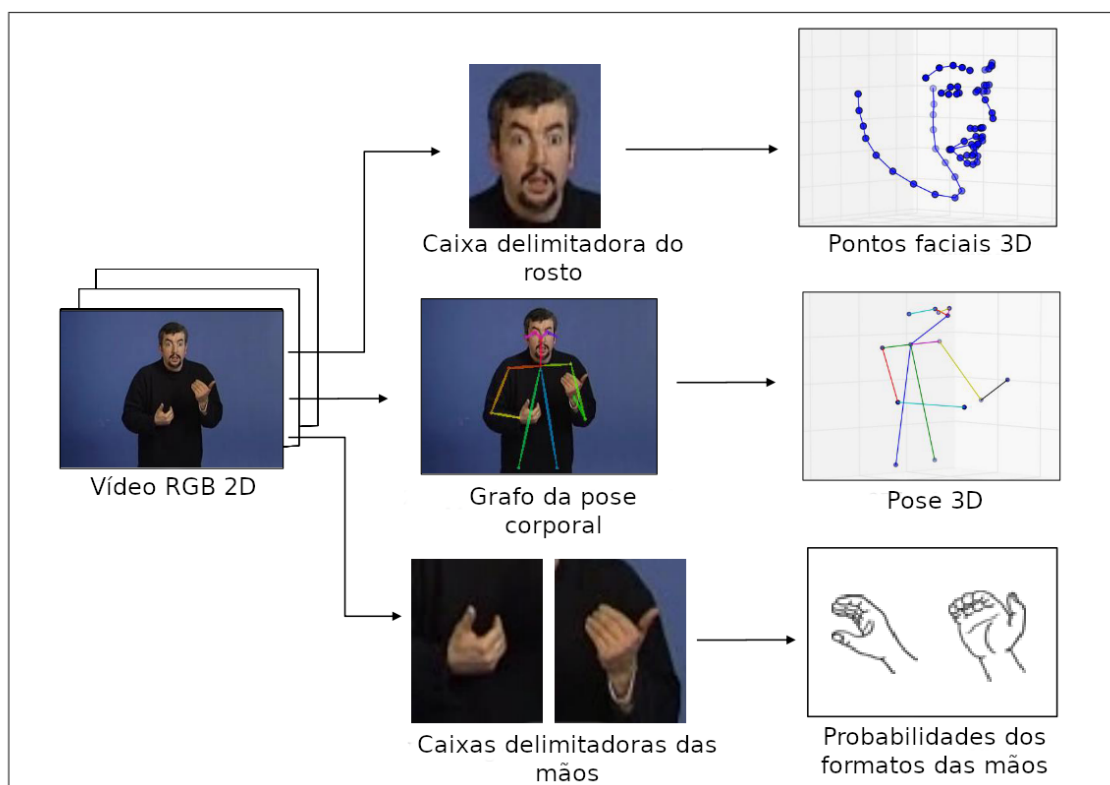
No que diz respeito à exploração dos parâmetros linguísticos, técnicas de inferência de pose corporal e esqueleto têm sido vastamente utilizadas na área, por permitir representar as dependências espaciais do corpo humano de maneira simplificada através de vértices e arestas.

Pu et al. (2016), por exemplo, obtiveram o esqueleto corporal durante a etapa de captura dos dados através do Kinect e o utilizaram para criar uma imagem de contexto de forma (que representa a distribuição dos vértices vizinhos a cada vértice de referência) e segmentar as mãos do intérprete na imagem. Feito isso, os autores utilizaram uma CNN 2D chamada LeNet (Jia et al., 2014) e uma arquitetura 3D com 5 camadas convolucionais criadas por eles para extração das características da imagem de contexto de forma e das mãos, respectivamente. Por fim, empregaram um modelo do tipo *Support Vector Machine* (SVM) (Noble, 2006) para classificar os 500 sinais da base CSL e obter uma acurácia de 85,8%. Embora os autores tenham demonstrado sucesso na classificação utilizando um modelo simples de SVM, as marcações não manuais foram negligenciadas pelo sistema, além de dependerem de um sensor específico (Kinect) para obtenção da pose corporal.

Belissen et al. (2019), por outro lado, utilizaram o OpenPose (discutido em mais detalhes na Seção 2.5.2) para obter as marcações corporais da pose do intérprete sobre a justificativa de gerar independência ao plano de fundo e também às variações de iluminação da cena. Feito isso, os autores utilizaram a informação da pose para segmentar a imagem do intérprete entre corpo, mãos e cabeça. Os canais do rosto e do corpo tiveram seus pontos de marcação projetados em um plano tridimensional, enquanto as características das mãos foram extraídas por meio de um modelo pré-treinado para estimativa de pose em imagens RGB, conforme ilustrado na Figura 3.1.

Após, o vetor de características dos três canais foi submetido a uma rede recorrente bidirecional do tipo LSTM, atingindo até 80% de *F1 score*. Porém, o trabalho apresenta limitações

Figura 3.1: Arquitetura multimodal utilizada por Belissen et al. (2019).



Fonte: Adaptada de Belissen et al. (2019).

em imagens com resolução mais baixa ou más condições de iluminação, destacando problemas como o borrão de movimento, que impactam diretamente na confiança da estimativa de pose. Além disso, conforme destacado pelos próprios autores, a aplicação não foi pensada para uso em tempo real, o que é reforçado pelos inúmeros passos necessários no pré-processamento das imagens e o seu consequente custo computacional.

Kindiroglu et al. (2019) focam na captura do movimento do intérprete e no formato das mãos ao invés da aparência propriamente dita. Para isso, são gerados mapas de calor acumulados temporalmente a partir das juntas do esqueleto obtidas pelo sensor Kinect, representando o movimento do sinal através de uma imagem no espaço de cor HSV (*Hue, Saturation, Value*), além do uso de segmentação semântica para destacar o formato das mãos e complementar a informação de movimento. Em seguida, uma CNN bidimensional simples é utilizada para extrair as características do mapa de calor, gerando uma acurácia de 81,58% no conjunto Bosphorus. No entanto, as expressões faciais não são utilizadas pelo classificador, além de que o custo computacional da segmentação e obtenção de esqueletos podem ser vistos como limitações do sistema.

De Coster et al. (2020) combinaram os pontos extraídos por meio do OpenPose e as carac-

terísticas aprendidas por uma CNN 2D do tipo ResNet-34 (He et al., 2016) em um *transformer* multimodal. Considerando o conjunto de dados da língua flamenga de sinais (indisponível publicamente), com 100 classes e 67 intérpretes, a acurácia obtida em teste foi de 74,70%. Segundo os autores, boa parte das características relevantes foram extraídas através das CNNs 2D, além de que muitos erros foram observados nas poses geradas, dada a baixa resolução dos vídeos de entrada e ao borrão de movimento.

Gökçe et al. (2020) utilizaram o OpenPose para segmentar as dicas visuais da imagem, separando a informação em três canais distintos, sendo eles: o corpo todo, a face, e ambas as mãos unidas horizontalmente. Além disso, os autores utilizam a informação de movimento das mãos obtida através da distância Euclidiana dos pontos chave (*keypoints*) dos dedos para determinar a mão dominante e padronizar a forma como as mãos são unidas em uma única imagem, além de ajudar no processo de amostragem dos quadros do vídeo. Feito isso, uma rede convolucional 3D chamada por eles de MC3 foi empregada para extração de características de cada um dos canais, e a média das probabilidades foi utilizada para gerar a classificação final. Os experimentos mostraram que utilizar o segmento específico das mãos gera melhores resultados do que trabalhar com o corpo todo, atingindo 86,25% de acurácia contra apenas 75,73%, podendo ser explicado pelo fato de que as mãos são responsáveis pela maioria dos parâmetros linguísticos que formam os sinais, conforme discutido no Capítulo 1, e apresentam pequena proporção em imagens de corpo todo. Além disso, o uso do canal da face gera até 2,08% de ganho nos resultados, sendo efetivo para remover ambiguidade de alguns sinais. Os autores atingiram uma acurácia final de 94,94% no conjunto BosphorusSign22k.

De Coster et al. (2021) também utilizaram o OpenPose para localizar as mãos do intérprete na imagem e um modelo do tipo VIT (*Visual Image Transformer*) (Vaswani et al., 2017) para classificação do sinal. Dentre os apontamentos, os autores destacam que o aumento da proporção das mãos gerado pelo foco local produz resultados superiores em relação ao uso da imagem completa, com 90,13% de acurácia contra 82,03% no conjunto AUTSL, respectivamente. Por fim, outra variação do modelo é criada, considerando não só as mãos, mas também o ângulo e a magnitude dos pontos a fim de capturar a informação de movimento, eliminando a redundância de alguns sinais e aumentando a acurácia para 91,51%. Ainda assim, nenhuma característica foi utilizada para representar de maneira eficiente o ponto de contato e as expressões faciais do intérprete, podendo gerar ambiguidades entre alguns sinais.

Maruyama et al. (2021) utilizaram um conjunto de modelos do tipo I3D para extrair as

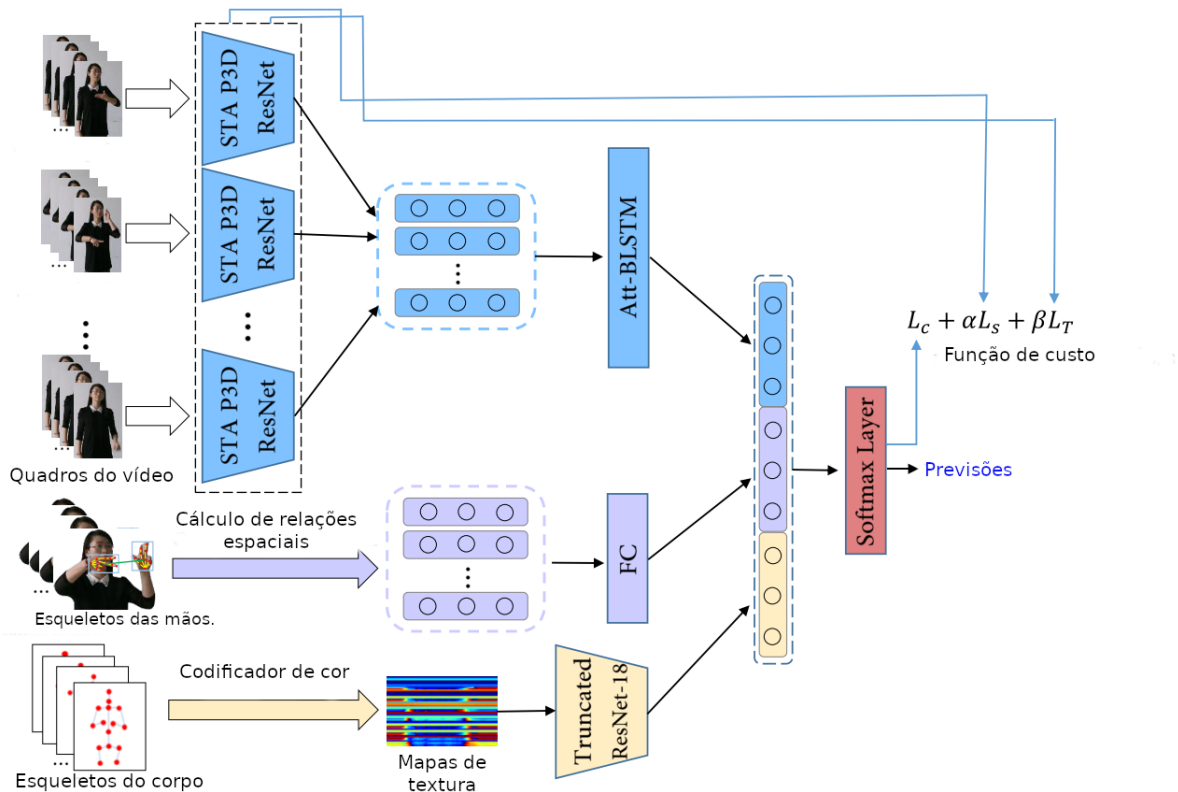
características das mãos, cabeça e do corpo todo em canais individuais, bem como a informação de movimento através de uma imagem de fluxo óptico. Os autores contaram com o OpenPose para segmentação das dicas visuais, além de submeter o esqueleto gerado a uma rede ST-GCN (*Spatial-Temporal Graph Convolutional Network*), especializada em extrair informação espaço-temporal dos vértices e arestas que compõe o esqueleto. Em seguida, os autores fizeram a média das probabilidades de todos os modelos para obter a previsão do sinal. Os resultados evidenciam os ganhos gerados pela exploração das dicas visuais, chegando em até 10% de melhoria em um subconjunto de 100 sinais da base MS-ASL. Além disso, é possível observar o impacto do tamanho do vocabulário na efetividade do sistema, visto que os resultados são reportados em diferentes subamostras do conjunto WLASL. A amostra com 100 sinais chega a uma acurácia de 81,38%, enquanto a com 1000 e 2000 sinais diminui para 63,61% e 47,26%, respectivamente.

Hosain et al. (2021) também utilizam do OpenPose para localizar as mãos do intérprete e realizar a operação espaço-temporal de *pooling* da convolução de maneira guiada a esta região de interesse da imagem, ajudando a arquitetura I3D a extrair informações mais relevantes para classificação do sinal. Por fim, os autores utilizam a fusão de diferentes versões desse modelo para chegar a uma acurácia de 38,84% nos 2000 sinais do WLASL. Ainda assim, é possível que melhores resultados fossem obtidos se outros parâmetros linguísticos também fossem valorizados pela arquitetura, conforme observado por Maruyama et al. (2021).

Xiao et al. (2021), por sua vez, separam a imagem em três canais distintos, sendo a imagem do corpo todo, a relação espacial das juntas das mãos e a imagem de textura gerada a partir da sequência do esqueleto do intérprete. As juntas e o esqueleto foram obtidos na captura dos dados do conjunto CSL por meio do Kinect, e os autores utilizaram uma arquitetura de rede convolucional chamada STA P3D (*Spatiotemporal Attention-based Pseudo-3D Residual Networks*) para extração de características espaço-temporais, um Perceptron multicamadas para a informação das juntas e uma rede do tipo ResNet com 18 camadas para os mapas de textura, conforme ilustrado na Figura 3.2. Por fim, um módulo de atenção para aplicar uma máscara sobre as regiões que não são de face ou mãos foi criado, além de desconsiderar as partes abaixo da cintura do intérprete. Os resultados obtidos na base CSL reportam uma acurácia de 89,9%.

Kratimenos et al. (2021) utilizam do OpenPose para extrair o esqueleto e uma biblioteca chamada SMPL-X (*Skinned Multi-Person Linear Model*) (Pavlakos et al., 2019) para criar uma projeção 3D do corpo humano a partir dos vértices e arestas obtidos. Os autores obtiveram uma acurácia de 94,77% na classificação de mais de 300 sinais em um conjunto da língua

Figura 3.2: Arquitetura multimodal utilizada por Xiao et al. (2021).



Fonte: Adaptada de Xiao et al. (2021).

grega de sinais chamado GSLL (*Greek Sign Language Lemmas*)(indisponível publicamente), mostrando que o método é superior ao uso de redes 3D e fluxo óptico em imagens RGB, sobre a justificativa de que a reconstrução 3D faz um mapeamento mais preciso dos detalhes corporais. Ainda assim, o método é caro computacionalmente e diretamente dependente das inferências geradas pelo OpenPose, que podem não funcionar bem em situações com maior movimento ou condições de iluminação mais adversas (De Coster et al., 2021, Belissen et al., 2019, Hosain et al., 2021). Além disso, a base de dados utilizada nos experimentos conta com apenas 2 intérpretes, podendo gerar dependência à aparência dos intérpretes ou à forma como os mesmos executam os sinais.

Gruber et al. (2021) empregam o OpenPose para segmentar as dicas visuais através do uso de máscaras na imagem, além de definir 15 quadrantes distintos no corpo do intérprete por meio da informação do esqueleto e da biblioteca Dlib (King, 2009) para extração dos pontos faciais, calculando a distância Euclidiana de cada uma das mãos em relação ao centro dos quadrantes, gerando um vetor de 30 dimensões que representam o ponto de contato do sinal. Por fim, os autores utilizaram uma união de 13 modelos distintos de I3D do tipo ResNet-50, três modelos de incorporação de linguagem visual (*Visual Language Embedding*) propostos por eles e um *pose*

transformer, para extrair as características da imagem RGB com máscara, das imagens das mãos do intérprete com os vetores de localização e do esqueleto do corpo, respectivamente. Embora a acurácia tenha chegado a até 95,56% no conjunto AUTSL, os autores não se preocuparam com as limitações de custo computacional.

Em suma, a Tabela 3 exibe um comparativo entre os trabalhos citados anteriormente, organizados pela data de publicação e base de dados utilizada.

Tabela 3: Comparação entre trabalhos que utilizam informações de pose.

Modelo	N. sinais	Pré-Proc.	Acc.
BI-LSTM (Belissen et al. (2019))	-	OpenPose+ CNN	80,00%
2DCNN (Kindiroglu et al. (2019))	855 (BosphorusSign)	TAF	81,58%
MC3 (Gökçe et al. (2020))	744 (BosphorusSign22k)	OpenPose	94,94%
LeNet+3DCNN+SVM (Pu et al. (2016))	500 (CSL)	OpenPose+ Contexto	85,80%
STA P3D+ResNet-18+FC (Xiao et al. (2021))	500 (CSL)	Mapas de Textura	89,90%
ResNet-34+ViT (De Coster et al. (2021))	226 (AUTSL)	OpenPose	86,90%
ResNet-34+ViT (De Coster et al. (2021))	226 (AUTSL)	OpenPose	90,13%
I3D+VLE+PT (Gruber et al. (2021))	226 (AUTSL)	OpenPose+ Dlib	95,56%
I3D (Hosain et al. (2021))	2.000 (WLASL)	OpenPose+ Pooling guiado	38,84%
I3D + ST-GCN (Maruyama et al. (2021))	2.000 (WLASL)	OpenPose+ Fluxo Óptico	47,26%
I3D + ST-GCN (Maruyama et al. (2021))	1.000 (MS-ASL)	OpenPose+ Fluxo Óptico	84,22%
BI-LSTM (Kratimenos et al. (2021))	300 (CSSL)	OpenPose+ SMPL-X	94,77%

Fonte: Autor.

3.2.1.2 Detecção de objetos

A detecção de objetos, por sua vez, se destaca por ser uma técnica mais simples e barata computacionalmente do que a extração de poses, além de ser mais robusta ao fator de borrão de movimento.

[Liao et al. \(2019\)](#) utilizaram duas bases de dados de grande escala em sua pesquisa, sendo o conjunto CSL, com 500 sinais distintos, e uma versão reduzida da Devisign, também com 500 sinais. Em sua abordagem, os autores utilizaram o Faster R-CNN para detectar a posição das mãos do intérprete e uma arquitetura chamada por eles de B3D ResNet, utilizando a convolução 3D para extração das características da imagem e as redes recorrentes bidirecionais do tipo LSTM para aumentar a capacidade de captura de contexto no sinal executado. Os resultados obtidos mostram uma acurácia de 89,8% no conjunto Devisign e 86,9% no CSL. Ainda assim, os parâmetros faciais que podem ser fundamentais para discernir alguns sinais são desconsiderados na pesquisa, além de ignorarem a normalização da posição das mãos na imagem, gerando dependência no posicionamento do intérprete frente à câmera para interpretação do ponto de contato.

[Agrawal e Urolagin \(2020\)](#), por outro lado, criaram um conjunto próprio contendo 200 palavras na língua árabe de sinais, utilizando o SSD como modelo de detecção de objetos para localizar o intérprete e recortar a imagem a uma resolução de apenas 64x28 *pixels*. Feito isso, os autores criaram uma imagem diferencial para representar o movimento corporal e a utilizaram como entrada para uma CNN, seguida de uma rede recorrente do tipo LSTM, chegando a 88,67% de acurácia. Embora a resolução reduzida e a pequena quantidade de detalhes presentes na imagem diferencial tenham permitido que os autores conseguissem uma acurácia alta utilizando um modelo com poucas camadas e parâmetros, os mesmos resultados dificilmente seriam atingidos em bases de dados mais complexas, visto que a imagem diferencial negligencia parâmetros com pouco movimento (expressões faciais) e detalhes relacionados ao formato das mãos, que podem ser fundamentais para reduzir ambiguidades entre sinais.

Por fim, [Zhang e Zhang \(2021\)](#) utilizaram dois canais para representar a informação proveniente do sinal, sendo um deles com a imagem completa do corpo do intérprete e o outro apenas do segmento das mãos, obtido utilizando a rede Mask R-CNN ([He et al., 2017](#)). As características do corpo todo foram extraídas por meio de uma rede adaptada pelos autores chamada de AM-ResC3D (*Attention-Model Resnet C3D*), que é baseada na arquitetura C3D ([Tran et al., 2015](#)), porém utiliza a rede ResNet como extrator de características. O seguimento das mãos,

por outro lado, têm as características obtidas por meio de uma arquitetura de rede 3D-VGGNet. Finalmente, os resultados obtidos pelos dois canais são unidos a fim de gerar a previsão final, atingindo uma acurácia de 91% em uma subamostra do conjunto de dados Devisign com 500 sinais.

Na Tabela 4 é possível acompanhar o comparativo detalhado entre os trabalhos citados.

Tabela 4: Comparação entre os trabalhos que utilizam detectores de objetos.

Modelo	N. sinais	Pré-Proc.	Acc.
BLSTM-3D (Liao et al., 2019)	500 (Devisign)	Faster RCNN	89,8%
BLSTM-3D (Liao et al., 2019)	500 (CSL)	Faster RCNN	86,9%
AM-ResC3D+3D-VGGNet (Liao et al., 2019)	500 (Devisign)	Mask-RCNN	91,0%
CNN + LSTM (Zhang e Zhang, 2021)	200 (próprio)	SSD+diferencial	88,67%

Fonte: Autor.

3.2.1.3 Demais técnicas de pré-processamento

Por fim, alguns trabalhos utilizam técnicas alternativas de pré-processamento nas imagens RGB, destacando ou agregando características importantes, enquanto outros apenas trabalham com a imagem completa como entrada para o classificador.

Wang et al. (2021), por exemplo, propõem o uso de CNNs (2+1)D (discutidas na Seção 2.3.9), por serem capazes de resolver problemas mais complexos enquanto mantêm ou até diminuem o número de parâmetros em relação às convoluções 3D convencionais. A arquitetura proposta, chamada de (2+1)D-SLR, utilizou toda a imagem RGB e não demandou nenhum tipo de pré-processamento, atingindo até 96,4% de acurácia na classificação dos 300 sinais distintos do conjunto NCSL. Os autores evidenciam que a rede proposta consegue ser até 54 milissegundos mais rápida que uma rede I3D, porém, não testam a capacidade de generalização da mesma em conjuntos de dados com uma quantidade maior de sinais ou com condições mais adversas.

Boukdir et al. (2021) compararam o uso de redes convolucionais 2D seguidas de redes LSTM em relação às redes 3D infladas (I3D) para imagens do corpo todo, chegando a uma média de 92,08% de acurácia contra 99,00%, respectivamente. Ainda assim, o comparativo não se estendeu a outra base além da criada pelos próprios autores, contendo apenas 56 sinais.

Sincan e Keles (2022) exploram as informações de movimento de forma detalhada através de uma imagem de histórico de movimento, também chamada de MHI (*Motion History Images*), destacando com maior brilho as regiões de movimento ao longo de uma série de quadros. Os autores utilizaram uma CNN do tipo ResNet-50 para extrair as características da imagem de movimento e uma I3D para as imagens do corpo todo do intérprete, fazendo a fusão tardia das características para classificação do sinal e atingindo 93,53% e 94,83% no conjunto AUTSL e BosphorusSign22k, respectivamente.

Han et al. (2022b) propõem uma alteração no modelo (2+1)D, adicionando mecanismos de atenção do tipo *Channel-Temporal Attention* (CTA) e (*Spatial-Temporal Attention*) (STA) em cada um dos blocos residuais. Os experimentos na base de dados CSL mostram que a modificação foi bem sucedida ao atingir até 97,45% de acurácia na classificação dos sinais, além de demonstrar resultados até 6% superiores às CNNs 3D.

Por fim, Han et al. (2022a) argumentam a respeito do elevado custo computacional que os modelos de CNN 3D geralmente possuem, propondo a criação de três novas arquiteturas com menor quantidade de parâmetros, a fim de viabilizar a execução em tempo real do sistema. Os autores mostram diferentes variações das arquiteturas criadas, evidenciando a compensação entre complexidade e acurácia. Sem nenhum pré-processamento adicional, os resultados obtidos chegam a 95% de acurácia na base CSL através de uma MobilenetV2 3D com apenas 3 milhões de parâmetros, enquanto um modelo alternativo de (2+1)D chega a até 47 milhões. Apesar dos resultados promissores, bases de dados com condições mais adversas também poderiam ser utilizadas para testar a arquitetura.

A Tabela 5 compara os trabalhos supracitados.

3.2.1.4 Considerações finais da Seção

Conforme destacado, diversos trabalhos adotam o uso de CNNs para imagens RGB bidimensionais, dada as vantagens já citadas desse tipo de modelo. Ainda assim, a técnica de pré-processamento e a base de dados são os principais fatores que diferenciam as pesquisas da área, devido à influência destes itens nos resultados obtidos pelo sistema, sendo divididos entre inferência de poses corporais, detecção de objetos e demais abordagens. Dito isso, a Seção 3.2.2 apresenta um paradigma diferente baseado em modelos especializados em processamento de grafos.

Tabela 5: Comparação entre os trabalhos que utilizam outras ou nenhuma técnica de pré-processamento.

Modelo	N. sinais	Pré-Proc.	Acc.
(2+1) D-SLR (Wang et al., 2021)	300 (NCSL)	-	96,4%
I3D (Boukdir et al., 2021)	56 (Própria)	-	92,08%
ResNet-50-CNN + I3D (Sincan e Keles, 2022)	226 (AUTSL)	MHI	93,53%
ResNet-50-CNN + I3D (Sincan e Keles, 2022)	744 (BosphorusSign22k)	MHI	94,83%
(2+1) D STCA (Han et al., 2022b)	500 (CSL)	-	97,45%
3D MobilenetV2 (Han et al., 2022a)	500 (CSL)	-	95,12%

Fonte: Autor.

3.2.2 Redes Convolucionais em Grafos

As redes convolucionais em grafos ou *graph convolutional networks* (GCN), são modelos alternativos às CNNs que têm sido utilizadas por simplificar a etapa de inferência e treinamento, visto que se baseiam unicamente nos vértices e arestas extraídos do corpo do intérprete.

de Amorim et al. (2019) se inspiraram em uma arquitetura de GCN espaço-temporal empregada no reconhecimento de ações em vídeos (Yan et al., 2018) para a tarefa de reconhecimento de sinais, analisando o posicionamento e movimentação dos pontos corporais ao longo do tempo para determinar a classe. Todavia, os resultados foram de apenas 20,85% de acurácia na base ASLLDV, mostrando que o modelo proposto não foi capaz de convergir, possivelmente por não considerar informações detalhadas das mãos e do rosto do intérprete, conforme relatado pelos autores.

Tunga et al. (2021) utilizaram uma GCN em conjunto com uma arquitetura baseada em transformadores chamada BERT (*Bidirectional Encoder Representations from Transformers*) (Devlin et al., 2019) para a codificação temporal da pose (Devlin et al., 2019). Os autores adotaram dois subconjuntos da base WLASL com 100 e 300 sinais, e obtiveram uma acurácia de 60,15% e 42,18%, respectivamente, evidenciando a dificuldade do modelo em diferenciar sinais espacialmente similares.

Por fim, Meng e Li (2021) utilizaram uma GCN com um mecanismo de atenção em múlti-

plas escalas para permitir que a rede consiga mapear vértices correlatos, ainda que não conexos, como o caso das mãos. Além disso, os autores utilizaram a biblioteca PoseNet para extrair as informações de pose do corpo e normalizaram a posição do esqueleto na imagem baseado no vértice do pescoço, gerando independência ao posicionamento da pessoa frente à câmera. Os resultados obtidos mostram uma acurácia de 98,08% e 65,57% nas bases de sinais CSL e DEVISIGN, respectivamente.

A Tabela 6 compara os trabalhos desta categoria.

Tabela 6: Comparação de trabalhos que utilizam modelos do tipo GCN.

Modelo	N. sinais	Pré-Proc.	Acc.
ST-GCN (Yan et al., 2018)	2.793 (ASLLVD)	OpenPose	20,85%
GCN+BERT (Tunga et al., 2021)	300 (WLASL)	OpenPose	42,18%
MSSTA+MSA+ATCN (Meng e Li, 2021)	500 (CSL)	PoseNet	98,08%
MSSTA+MSA+ATCN (Meng e Li, 2021)	2.000 (Devisign)	PoseNet	65,57%

Fonte: Autor.

3.3 Considerações finais do Capítulo

Diversos trabalhos relevantes na área de reconhecimento de sinais a nível de palavras têm sido realizados nos últimos anos, destacando-se principalmente pela forma de pré-processamento das imagens a fim de melhor explorar os parâmetros linguísticos. Ainda assim, há espaço para melhorias significativas, principalmente no que tange à redução de complexidade e custo computacional das soluções propostas. A este respeito, embora alguns trabalhos tenham atingido taxas altas de acurácia, muitos deles utilizam arquiteturas com múltiplos canais ou empregam um conjunto de modelos distintos para gerar a previsão, inviabilizando a execução do sistema em tempo real ou em dispositivos de borda como os aparelhos celulares.

Ademais, os métodos baseados na inferência de poses corporais, embora muito adotados por permitir a representação de informações espaço-temporais relevantes para classificação do sinal de maneira simplificada através de um único vetor de vértices e arestas, implicam em um custo computacional alto para o sistema (Cao et al., 2019), além de sofrer com fatores como o borrão

de movimento, muito comuns na execução dos sinais. A detecção de objetos, por outro lado, é menos impactada pelo movimento rápido das mãos, além de possibilitar o treinamento de arquiteturas com custo computacional reduzido para execução em dispositivos mais acessíveis (Huang et al., 2017).

Por fim, outras técnicas de pré-processamento ou modelos podem ser empregados a fim de explorar as informações presentes na imagem de maneira eficiente. Dito isto, o Capítulo 4 apresenta a proposta deste trabalho, que visa se beneficiar da estrutura da língua de sinais para extrair informações apenas das partes que realmente importam na imagem, simplificando a etapa de coleta e classificação do sinal.

4 Desenvolvimento do Projeto

Conforme discutido no Capítulo 3, boa parte dos trabalhos propostos na literatura utilizam de aprendizado profundo para o reconhecimento de sinais, principalmente as CNNs, consideradas estado da arte na tarefa. Além disso, diferentes técnicas de pré-processamento podem ser aplicadas a fim de alimentar o modelo com características mais relevantes e gerar ganhos nos resultados.

Por outro lado, apenas uma parcela minoritária das pesquisas apresentam uma preocupação com o custo computacional e a escalabilidade do sistema em situações diversas de velocidade de movimento, iluminação e enquadramento, podendo ser limitantes para a adoção da solução criada fora de ambiente laboratorial.

Dito isso, este trabalho introduz uma nova técnica de pré-processamento baseada em detecção de objetos que visa segmentar as partes mais importantes da imagem sem perder a informação relativa ao movimento e posicionamento espacial, fazendo com que seja possível realizar a classificação dos sinais com uma quantidade reduzida de informação. Ainda que a execução do sistema em dispositivos mais simples, como aparelhos celulares, não seja o foco desta pesquisa, a troca entre custo computacional e ganho de acurácia do método proposto pode beneficiar futuros trabalhos neste sentido.

Além disso, diferentes detectores de objetos são treinados e comparados em termos de precisão e tempo de inferência, promovendo o melhor modelo para integrar a etapa de pré-processamento do sistema. O restante deste capítulo aborda a respeito dos materiais e métodos que foram empregados na pesquisa, vistos na Seção 4.1, a metodologia utilizada para condução e validação dos experimentos, apresentada na Seção 4.2, e as considerações finais do capítulo, na Seção 4.3.

4.1 Materiais e métodos

Os materiais e métodos contemplam os principais insumos que suportaram o desenvolvimento da pesquisa, como as tecnologias e o *hardware* utilizados (Seção 4.1.1), as bases de dados para classificação de sinais (Seção 4.1.2) e detecção de objetos (Seção 4.1.3), os métodos de pré-processamento (Seção 4.1.4) e as arquiteturas de aprendizado profundo baseadas em CNNs propostas para o reconhecimento de sinais (Seção 4.1.5).

4.1.1 Tecnologias e *hardware*

Dentre as tecnologias utilizadas, destaca-se o uso da linguagem de programação Python (versão 3.8.10) (Van Rossum et al., 2007), sendo base para todo o desenvolvimento do sistema. Além disso, bibliotecas como TensorFlow (versão 2.8.2) (Abadi et al., 2016), PyTorch (versão 1.13.0) (Ketkar et al., 2021) e Keras (versão 2.8.0) (Chollet et al., 2015) foram empregadas para criação e manutenção das arquiteturas de redes neurais profundas, bem como para seu treinamento. A especificação do versionamento de tais ferramentas é relevante por impactar na performance dos modelos reportada na pesquisa. Ademais, o mesmo *hardware* foi utilizado no treinamento e avaliação de todos os modelos, composto principalmente por uma CPU (*Central Processing Unit*) Intel Core I5-10400 e uma GPU (*Graphics Processing Unit*) Nvidia RTX 3060.

Por fim, bibliotecas de processamento e manipulação de dados como o Scipy (Virtanen et al., 2020), Numpy (Harris et al., 2020), Pandas (McKinney et al., 2011) e OpenCV (Bradski, 2000) foram adotadas para suportar as etapas de pré e pós processamento da solução. Tanto a arquitetura de detecção de objetos quanto a da classificação de sinais foram disponibilizadas de maneira *open-source* em repositórios do GitHub¹², aumentando a contribuição da pesquisa para com trabalhos futuros.

4.1.2 Bases de dados para classificação

Conforme discutido na Seção 3.1, a criação manual de uma base de dados é muito onerosa e complexa, mesmo que sejam adotados métodos semi-supervisionados para ajudar na automação da tarefa. Por conta disto, esta pesquisa não teve por objetivo propor uma nova base de dados de língua de sinais, mas utilizar conjuntos de dados disponíveis publicamente. Para tal, ao menos duas bases distintas foram escolhidas, permitindo que a avaliação do método proposto seja realizada entre diferentes idiomas e tamanhos de vocabulário, além de possibilitar o comparativo com os demais trabalhos da literatura.

O primeiro conjunto é o AUTSL da língua turca de sinais (mostrado na Seção 3.1.1), por ser amplamente explorado em pesquisas recentes da área, além de contar com uma grande variedade de intérpretes, cenários distintos e repetições por sinal, sendo ideal para desafiar o desempenho de sistemas de reconhecimento de sinais. A Figura 4.1 ilustra alguns exemplos da

¹<https://github.com/AlvaroCavalcante/hand-face-detector>

²<https://github.com/AlvaroCavalcante/multi-cue-sign-language>

base de dados em questão.

Figura 4.1: Exemplos de imagens da base de dados AUTSL [Sincan e Keles \(2020\)](#)



Fonte: [Sincan e Keles \(2020\)](#).

Todos os vídeos desta base de dados foram capturados a uma taxa de 30 quadros por segundo e uma resolução de 512×512 *pixels*, possuindo uma duração média por vídeo de 2,05 segundos e 61 quadros. A divisão do conjunto entre treinamento, teste e validação seguiu as diretrizes estabelecidas pela competição ([Sincan et al., 2021](#)), fazendo com que cerca de 77,5% dos dados fossem utilizados para treinamento, 12,1% para validação e 10% para teste. Além disso, as bases de treinamento, teste e validação são compostas por 31, 6 e 6 intérpretes, respectivamente, divididos de maneira independente, ou seja, sem intersecção do mesmo intérprete em diferentes conjuntos de dados.

Como forma alternativa de avaliar a proposta desenvolvida, o conjunto CSL da língua chinesa de sinais também foi utilizado, apresentando um número maior de intérpretes e sinais. Conforme sugerido pelos autores ([Liu et al., 2016a](#)), a base foi dividida de acordo com os intérpretes, mantendo 36 indivíduos no conjunto de treinamento e 14 no conjunto de teste, totalizando 90000 (72%) e 35000 (28%) vídeos para cada um dos conjuntos, respectivamente. Ademais, as gravações foram disponibilizadas a uma resolução de 1280×720 *pixels* e uma taxa de captura de 30 quadros por segundo, possuindo uma duração média por vídeo de 3,17 segundos e 95 quadros, além de manter um enquadramento fixo de 1,5 metros de distância entre a câmera e o intérprete, conforme ilustrado na Figura 4.2. Ainda que o conjunto em questão possua condições laboratoriais de gravação, o amplo número de intérpretes e classes podem representar um desafio considerável para o sistema, além do fato de que os sinais executados são compostos por gestos mais longos, possuindo uma duração média maior.

Figura 4.2: Exemplos de imagens da base de dados CSL [Pu et al. \(2016\)](#)



Fonte: Autor.

Por fim, as seções subsequentes apresentam em mais detalhes as diferenças entre os dois conjuntos de dados empregados por esta pesquisa e os resultados obtidos em cada um deles.

4.1.3 Bases de dados para detecção de objetos

O primeiro desafio relacionado ao treinamento dos detectores de objetos foi a obtenção de uma base de dados contendo as anotações das mãos e do rosto dos indivíduos. Nesse sentido, [Mohaimenianpour e Vaughan \(2018\)](#) fizeram uma coleção de 9 bases de dados públicas possuindo, ao todo, mais de 94.500 rótulos de mãos e 58.000 de faces, conforme ilustrada pela Figura 4.3. Dentre as bases de dados que foram compiladas pelos autores, os conjuntos EgoHands ([Bambach et al., 2015](#)) e Viva ([Das et al., 2015](#)) não foram utilizados para o treinamento do detector de objetos, por serem compostos apenas por imagens das mãos em um ângulo de primeira pessoa, não sendo condizente com o domínio de linguagem de sinais. Com isso, o número de imagens foi reduzido para um total de 40.064, contendo 66.572 mãos e 56.881 faces.

Ainda assim, o conjunto de dados em questão apresenta situações muito adversas de enquadramento, ângulo e número de indivíduos nas imagens, e embora essas características possam ser relevantes em termos de generalização para um detector de mãos e faces de propósito geral, é possível presumir que, no contexto da língua de sinais, haverá apenas um único indivíduo centralizado na imagem e em condições mais previsíveis, conforme ilustrado nas Figuras 4.1 e 4.2.

Figura 4.3: Exemplos de imagens com as mãos e a face anotadas das bases de dados públicas compiladas por Mohaimenianpour e Vaughan (2018)



Fonte: Autor.

Dessa forma, treinar os detectores em uma base de língua de sinais pode ser mais adequado ao propósito da tarefa, facilitando a convergência do modelo e permitindo uma hiperparametrização específica para este domínio. Pensando nisso, uma nova base de dados foi proposta, contendo imagens do conjunto AUTSL anotadas de maneira semi-supervisionada através de um modelo pré-treinado. Embora a base de dados CSL contenha um número maior de vídeos e intérpretes, o conjunto AUTSL conta com uma ampla variedade de planos de fundo e condições de iluminação, sendo adequado para o treinamento de um detector robusto e, por conta disso, escolhido para compor a primeira versão da base de dados.

Para a criação semi-supervisionada das anotações, um detector inicial do tipo EfficientDet D0 (Tan et al., 2020) foi treinado utilizando a base de dados de propósito geral supracitada. Feito isso, as inferências geradas pelo modelo no conjunto AUTSL foram convertidas para o formato adequado através da biblioteca auto-annotate³. Além disso, para aumentar o número de imagens anotadas, o limiar mínimo de apenas 35% de confiança foi adotado para que uma previsão seja considerada correta e tenha sua anotação gerada. Ainda assim, como forma de reduzir o esforço na curadoria dos dados, apenas as previsões das duas mãos e da face com maior confiança na imagem foram mantidas, além de descartar imagens em que algum dos três

³https://github.com/AlvaroCavalcante/auto_annotate

objetos de interesse não tenha sido detectado com o mínimo de confiança necessária.

Por fim, as anotações geradas foram avaliadas e corrigidas manualmente com o auxílio do *software* LabelImg (Lin, 2015), ajustando as proporções das caixas delimitadoras e eliminando as inferências falso positivas. O novo conjunto de dados proposto contém um total de 477.480 imagens, sendo que cada imagem possui exatamente uma face e duas mãos anotadas, e a divisão entre treinamento, teste e validação seguiu as mesmas diretrizes descritas na Seção 4.1.2, separando os intérpretes de maneira a evitar intersecções entre as bases, fazendo com que o conjunto de teste e validação sejam compostos por indivíduos não observados durante o treinamento. Além de ser utilizado para treinar os detectores de objetos específicos para o domínio, as anotações também foram disponibilizadas publicamente⁴ para uso em futuras pesquisas.

4.1.4 Pré-processamento do sistema

A arquitetura proposta visa explorar diferentes aspectos dos sinais para promover a extração de características relevantes da maneira mais eficiente possível. Para tal, além do modelo de CNN base empregado para extração de características, a etapa de pré-processamento possui um papel fundamental no sistema. Dito isto, esta Seção visa explicar as principais técnicas de pré-processamento propostas pela pesquisa, sendo elas: os métodos de detecção de objetos para extração de dicas visuais, vistos na Seção 4.1.4.1; a criação de um vetor de atributos para representar o ponto de contato e o movimento do sinal, explicado na Seção 4.1.4.2; a geração de uma figura geométrica triangular como alternativa para representação do ponto de contato, discutido na Seção 4.1.4.3; e a exploração das técnicas de aumento de dados, na Seção 4.1.4.4.

4.1.4.1 Detecção de dicas visuais para língua de sinais

Conforme relatado por diversos trabalhos recentes, as dicas visuais, principalmente relacionadas ao foco nas mãos do intérprete, têm gerado melhores resultados para sistemas de reconhecimento de sinais (Pu et al., 2016, Belissen et al., 2019, Gökçe et al., 2020, De Coster et al., 2021, Maruyama et al., 2021, Kratimenos et al., 2021). Portanto, um dos primeiros objetivos da pesquisa é o de propor um método para localizar e extrair os segmentos das mãos e do rosto, permitindo com que os mesmos sejam manipulados de maneira individual e com maior proporção na imagem.

Para isso, diversas arquiteturas de detecção de objetos foram treinadas e comparadas, entre

⁴<https://github.com/AlvaroCavalcante/hand-face-detector>

elas o SDD, YOLO, Faster-RCNN, CenterNet e o EfficientDet, discutidas em detalhes na Seção 2.4. Diferentemente dos métodos de extração de pose corporal, os detectores de objetos possuem menos requisitos computacionais e são capazes de serem executados em tempo real em dispositivos móveis (Martinez-Alpiste et al., 2022), podendo ser uma solução mais viável no que diz respeito à acessibilidade dos deficientes auditivos.

Além disso, algumas adaptações foram realizadas nos modelos supracitados, uma vez que as configurações sugeridas pelos autores foram pensadas para detectores de propósito geral, visando lidar com inúmeros objetos de diferentes formas, tamanhos e proporções. Por conta das mudanças arquiteturais realizadas em alguns dos detectores, a técnica de transferência de aprendizado foi inviabilizada e, devido a isso, não foi utilizada em nenhum dos modelos, garantindo um comparativo mais justo.

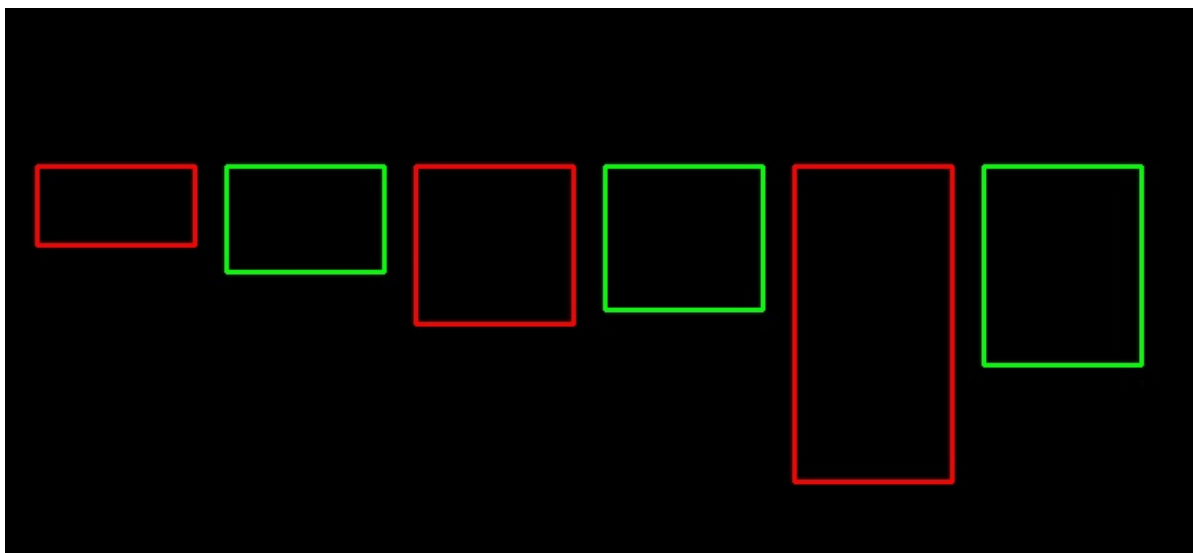
A primeira simplificação realizada em todos os modelos foi reduzir o número máximo de previsões por classe e o total de caixas delimitadoras de 100 para apenas 2 e 4, respectivamente, pelo fato de que o modelo deve detectar apenas duas mãos e uma face na imagem, pertencentes ao indivíduo centralizado que se encontra executando o sinal. Dessa forma, menos iterações são necessárias pelo algoritmo para filtrar previsões incorretas ou aplicar o NMS para remover caixas sobrepostas.

Outra modificação fundamental foi a redução no número de proporções de caixas âncoras de 3 para 2, além de adequar seus valores iniciais para o conjunto de dados utilizado no treinamento. Conforme visto na Seção 2.4.2, as caixas âncoras são utilizadas como uma estratégia de subamostragem em uma determinada região da imagem, onde figuras de diferentes proporções e escalas são avaliadas a fim de encontrar a que melhor se ajusta aos objetos ali presentes. Dito isso, diminuir o número de proporções pode reduzir a capacidade do detector de localizar objetos de determinados formatos ou em posições inusitadas, porém, é uma abordagem útil neste contexto, visto que as mãos e faces possuem uma proporção pouco mutável.

Ademais, o valor da proporção entre altura e largura das caixas âncora pode ser alterado de acordo ao problema tratado, aumentando a chance de sucesso em localizar os objetos. Dessa forma, um algoritmo de agrupamento não supervisionado chamado Kmeans (Sinaga e Yang, 2020) foi utilizado para encontrar as proporções das caixas com maior índice de união sobre intersecção no conjunto de dados de treinamento, pré-ajustando as figuras ao domínio do problema e, com isso, facilitando a convergência do modelo. A Figura 4.4 exemplifica esse processo, comparando três caixas âncora utilizadas por padrão no algoritmo SSD (em vermelho)

ao lado de três caixas com proporção otimizada para a tarefa de detecção de mãos e face (em verde).

Figura 4.4: Comparação entre diferentes proporções de caixas âncoras, considerando o padrão do modelo SSD (em vermelho) e as otimizadas para o contexto de língua de sinais (em verde).



Fonte: Autor.

Além das alterações supracitadas, aplicadas para todos os modelos (com exceção da CenterNet, por não depender de caixas âncora, e do Yolo, por detalhes arquiteturais explicados a seguir), as demais modificações foram realizadas de maneira específica, a depender da arquitetura. No caso do EfficientDet, por exemplo, a profundidade da rede de pirâmide de características foi reduzida de 5 para 4, eliminando a camada mais superficial, pelo fato de que os objetos de interesse não são tão pequenos. Além disso, o número de iterações da pirâmide bidirecional foi reduzida de 3 para 2, os filtros por mapa de características foram reduzidos de 64 para 48 e a quantidade de camadas das redes responsáveis pela classificação e regressão das caixas delimitadoras foram diminuídas de 3 para 2. Tais alterações foram motivadas pela hipótese de que o problema em questão é mais simples do que o problema que o modelo foi desenvolvido para resolver, visto que o mesmo precisa detectar apenas duas classes (mãos e face) em condições similares.

No caso do SSD, as alterações foram semelhantes ao EfficientDet, reduzindo-se o número de mapas de características adicionais de 5 para 4 e o número de camadas do previsor de caixas delimitadoras de 4 para 3. Ademais, diferentemente do trabalho original, a CNN base utilizada no SSD foi uma MobileNet V2, conforme observado no trabalho de [Li et al. \(2018b\)](#), reduzindo o custo computacional do modelo sem grades impactos nos resultados. Por fim, duas versões dis-

tintas do modelo foram adotadas neste trabalho, identificadas pelos nomes SSD320 e SSD640, utilizando imagens de entrada com resolução de 320x320 e 640x640 *pixels*, respectivamente.

O Faster-RCNN, por sua vez, contou com duas alterações específicas, sendo elas a diminuição no número de escalas das caixas âncora e a redução no número de propostas geradas pela primeira etapa da rede de 300 para apenas 50. Diferentemente da proporção, que dita o formato das figuras, as escalas são utilizadas para controlar o tamanho da caixa âncora na imagem, sendo um importante elemento para identificar objetos de diferentes tamanhos ou distâncias da câmera. Dito isso, o maior valor de escala sugerido pelos autores não foi utilizado, pelo fato de que as mãos e a face não são objetos muito grandes e o intérprete deve manter uma distância mínima da câmera, para permitir o enquadramento das mãos durante a execução dos sinais. As propostas de regiões, por outro lado, foram reduzidas pelo fato de que poucos objetos de interesse irão aparecer na imagem simultaneamente, reduzindo o esforço para geração das inferências pela segunda etapa do modelo. Por último, conforme alternativa sugerida pelos próprios autores, a CNN base utilizada foi a ResNet, porém, em sua versão com apenas 50 camadas de profundidade, reduzindo o custo da arquitetura.

O CenterNet, entretanto, não recebeu nenhuma alteração adicional, por ser um modelo com poucas opções de melhorias específicas a um problema, sendo que a única modificação em relação ao trabalho original foi a adoção da MobileNet como extrator de características, assim como realizado no SSD. Por fim, é importante mencionar que treinamentos experimentais foram conduzidos mediante a alteração de cada um dos parâmetros supracitados nas arquiteturas, garantindo que a convergência do modelo não fosse prejudicada.

Por fim, a sétima versão do modelo Yolo foi utilizada por esta pesquisa, chamada simplesmente de YoloV7 (Wang et al., 2022), sendo um dos modelos mais recentes e com maiores otimizações para detecção de objetos em tempo real. A arquitetura em questão utiliza um previsor auxiliar para gerar as inferências a partir das iterações de uma pirâmide de características bidirecional, atuando como uma forma de *ensemble* dentro de um mesmo modelo. Por conta disso, a profundidade e o número de camadas da pirâmide não foi alterado, bem como a quantidade de caixas âncoras propostas, a fim de evitar mudanças muito profundas no funcionamento da arquitetura. Ainda assim, o detector foi simplificado através da redução empírica no número de filtros convolucionais de todas as camadas, seguindo a hipótese da simplicidade do problema em questão, resultando em uma redução no número de parâmetros de 36.487.166 para apenas 14.169.470. Ademais, a resolução das imagens utilizadas como entrada para o modelo foi de

512x512 *pixels*, diferentemente dos 640x640 *pixels* sugeridos pelo trabalho original.

Além dos ajustes finos empregados nos modelos, as técnicas de aumento de dados utilizadas durante o processo de treinamento também foram padronizadas para todos os algoritmos, além de serem cuidadosamente escolhidas para o domínio em questão. Dessa forma, algumas transformações geométricas foram desconsideradas ou atenuadas para evitar que as mãos fossem cortadas da imagem, por exemplo, ou que a informação fique muito distorcida, como o caso das rotações verticais, visto que os intérpretes se encontram em apenas uma única posição vertical frente à câmera. Dentre as operações de intensidade, destaca-se o ajuste no brilho, contraste, saturação, conversão para escala de cinza, distorção aleatória nos canais de cor RGB, alterações no Hue e criação de *patches* pretos aleatórios na imagem. As operações geométricas, por sua vez, são compostas pela rotação horizontal, alterações na escala da imagem e recortes aleatorizados. A Figura 4.5 ilustra algumas das operações de aumento de dados.

Figura 4.5: Exemplo mostrando a imagem original (canto superior esquerdo) seguido de operações de aumento de dados, sendo elas: rotação horizontal, escala de cinza, adição de *patches* pretos, distorção nos canais de cor e ajustes no contraste.



Fonte: Autor.

4.1.4.2 Representação do movimento e do ponto de contato

Conforme ilustrado na Figura 1.1, a maior parte da imagem é utilizada para representar o ponto de contato, ou seja, a localização espacial das mãos no sinal em execução. O uso deste parâmetro gera maior demanda por poder computacional, dada a necessidade de processar todos os *pixels* presentes na região, além de aumentar a complexidade do sistema, pois é preciso considerar a invariância a fatores de cenário e iluminação.

Baseado nisso, esta pesquisa propõe um método para representar a informação do ponto de contato de maneira simplificada através da exploração da informação espacial gerada pelas posições centrais das caixas delimitadoras inferidas pelos detectores de objetos propostos na Seção 4.1.4.1. Para isso, o primeiro passo necessário é encontrar as coordenadas (x, y) da região central das figuras que demarcam a localização da face e das mãos do intérprete, conforme representado na Equação 4.1:

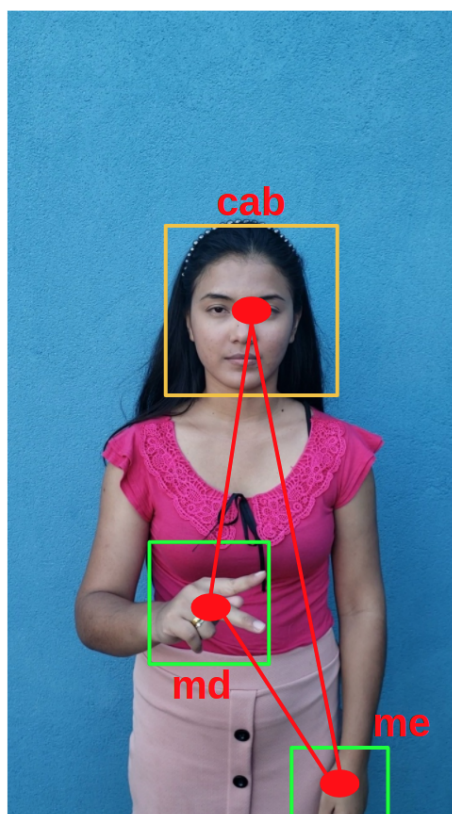
$$c(x_{min}, x_{max}, y_{min}, y_{max}) = \left[\frac{x_{min} + x_{max}}{2}, \frac{y_{min} + y_{max}}{2} \right], \quad (4.1)$$

em que x_{min} e x_{max} são os pontos mínimos e máximos da caixa delimitadora no eixo x da imagem, enquanto y_{min} e y_{max} são os pontos mínimos e máximos do eixo y . Feito isso, cada um dos pontos centrais podem ser interligados, gerando uma figura triangular, conforme ilustrado na Figura 4.6.

Como é possível notar, a figura geométrica interliga os locais da imagem que contêm maior informação a respeito do sinal, além de permitir criar uma referência espacial da posição das mãos em relação à face do intérprete, pois a mesma se mantém estática. Dentre as pesquisas presentes na literatura, apenas duas foram encontradas, até o presente momento, utilizando características similares às supracitadas. A primeira, desenvolvida por Xiao et al. (2020), foi voltada para o reconhecimento contínuo de sinais, onde os autores empregaram um detector de objetos do tipo Faster R-CNN para localizar as mãos e a face do intérprete e criar uma figura triangular, conforme ilustrado na Figura 4.7.

Conforme observado, os autores criam uma nova imagem contendo o desenho formado pelo triângulo, submetem a mesma a uma CNN para extração de características e, posteriormente, a uma rede recorrente do tipo LSTM para classificação. Ainda assim, gerar uma imagem e utilizá-la para treinar um novo modelo pode aumentar consideravelmente a complexidade e custo computacional do sistema, devido à quantidade de parâmetros necessários para extrair as características e padrões dos *pixels*.

Figura 4.6: Figura triangular gerada pela ligação entre os centroides das caixas delimitadoras.



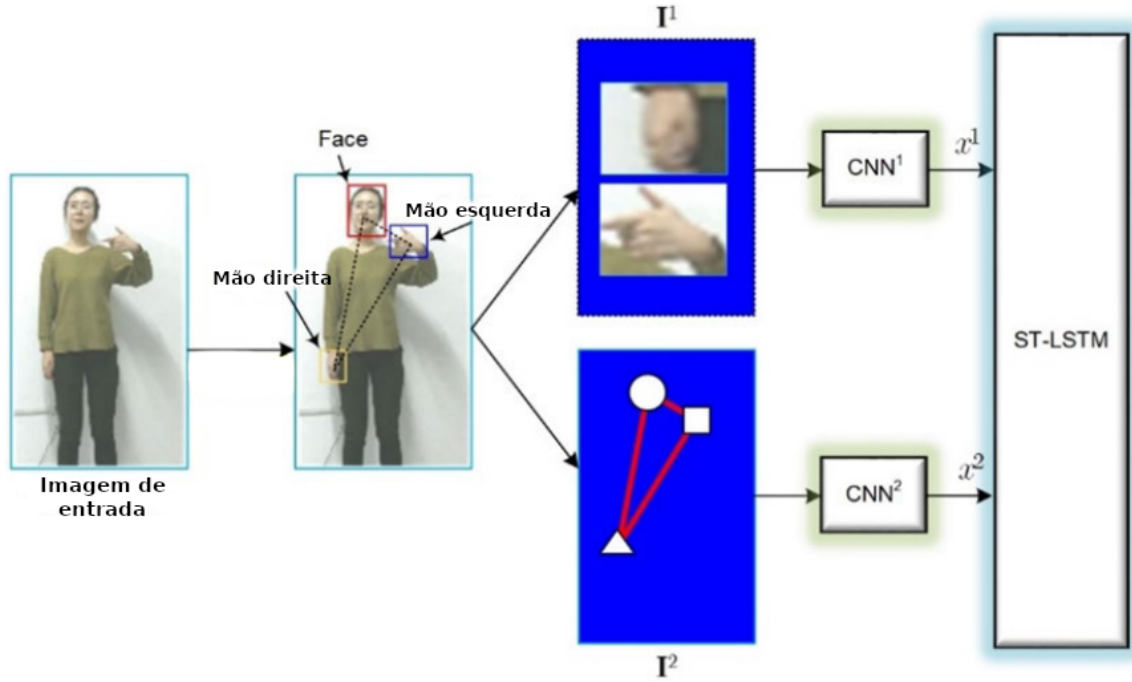
Fonte: Autor.

No trabalho de (Talukdar e Bhuyan, 2022), por outro lado, as coordenadas x e y da posição das mãos e da face do intérprete foram utilizadas em cada quadro do vídeo para compor o modelo, juntamente a outras três características provenientes das mãos, sendo os vetores de movimento dos centroides, a orientação do eixo principal de uma elipse que foi desenhada em torno das mãos e um histograma de gradientes orientados. Ainda assim, o objetivo principal dos autores foi o de gerar uma distinção entre gestos que representavam sinais e gestos transitórios, no contexto da linguagem contínua de sinais.

Dessa forma, diferentemente de trabalhos anteriores, a técnica proposta por este trabalho visa criar uma abstração vetorial para representar a informação do triângulo através de atributos que formam a figura, utilizando-os como entrada para classificação dos sinais. Com isso, os seguintes valores foram computados a fim de possibilitar a representação numérica da figura:

- I Distância euclidiana entre os centroides.
- II Ângulos internos e externos.
- III Perímetro e o semiperímetro.

Figura 4.7: Arquitetura de reconhecimento e criação de figura geométrica proposta por Xiao et al. (2020)



Fonte: Adaptado de Xiao et al. (2020).

IV Área.

V Altura.

O primeiro cálculo realizado é a distância dos centroides no espaço, sendo obtida através da Equação 4.2:

$$dist(a, b) = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}, \quad (4.2)$$

em que x e y representam os eixos da imagem no plano 2D e a e b identificam as coordenadas do ponto. Assim, o vetor com as distâncias das mãos e da face pode ser representado da seguinte forma: $\mathbf{d} = [dist(md, me), dist(md, cab), dist(me, cab)]$, em que md , me e cab são as coordenadas dos centroides da mão direita, esquerda e da cabeça do intérprete, respectivamente. Ao obter o tamanho dos lados por meio da distância relativa dos três centroides, é possível calcular o perímetro (pr) e o semiperímetro (spr) do triângulo, dado pelas Equações 4.3 e 4.4, respectivamente:

$$pr(\mathbf{d}) = \sum (d_i), \quad (4.3)$$

$$spr(pr) = \frac{pr}{2}. \quad (4.4)$$

A área do triângulo, por sua vez, pode ser calculada de diferentes formas dependendo do tipo de triângulo (equilátero, isósceles, escaleno, dentro outros), porém, visto que várias figuras distintas podem ser obtidas durante a execução dos sinais, a fórmula de Heron (Bényi, 2003) foi adotada para o cálculo genérico da área, representada na Equação 4.5:

$$\lambda = \sqrt{spr(spr - d_1)(spr - d_2)(spr - d_3)}. \quad (4.5)$$

A altura, por outro lado, é dada pela Equação 4.6:

$$h(\lambda, \mathbf{d}) = \frac{2\lambda}{d_1 + \gamma}, \quad (4.6)$$

em que a distância entre a mão direita e a mão esquerda (d_1) é utilizada por ser a base do triângulo e γ representa um pequeno valor constante positivo (1e-10) para evitar a divisão por zero. Por fim, a obtenção dos ângulos internos do triângulo é dada pela lei dos cossenos (Robusto, 1957), representada na Equação 4.7:

$$op^2 = adj_1^2 + adj_2^2 - 2adj_1adj_2 \cos \alpha, \quad (4.7)$$

em que op representa o tamanho do lado oposto ao ângulo α , enquanto adj_1 e adj_2 são os lados adjacentes. Feito isso, os três ângulos externos podem ser facilmente calculados através da soma dos ângulos internos não adjacentes a ele, conforme o teorema dos ângulos externos, expresso na Equação 4.8:

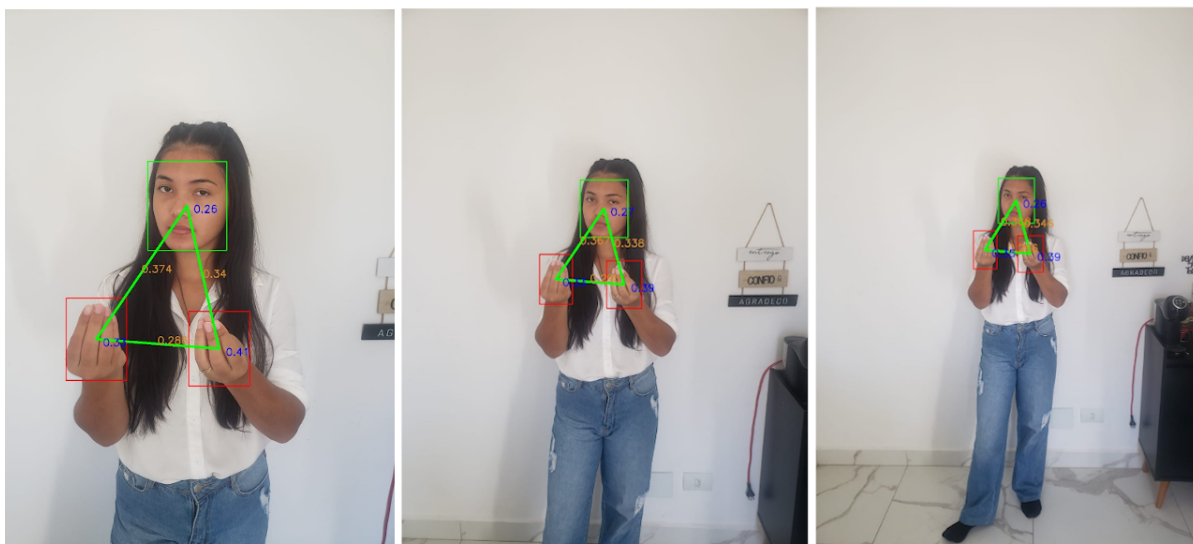
$$\hat{\alpha} = \alpha_a + \alpha_b, \quad (4.8)$$

em que α_a e α_b são os ângulos internos não adjacentes ao ângulo externo $\hat{\alpha}$.

Um dos desafios de representar o triângulo através das características supracitadas é a normalização dos valores, de maneira que figuras similares possuam valores similares, independentemente da distância do intérprete frente à câmera. Para que isso seja possível, as distâncias de cada um dos vértices foram divididas pelo perímetro do triângulo, fazendo com que as demais equações sigam a mesma proporção e garantindo que o vetor de características seja uma representação precisa do formato da figura, não sendo influenciado pelo enquadramento e distância

do intérprete em relação a câmera, conforme ilustrado pela Figura 4.8.

Figura 4.8: Comparação entre o formato e os valores do triângulo considerando o mesmo sinal capturado de formas diferentes. Os ângulos internos são representados em azul, enquanto as distâncias são mostradas em laranja.



Fonte: Autor.

Os valores dos ângulos internos (em azul) são: $[0,33,0,41,0,26]$, $[0,33,0,39,0,27]$, $[0,35,0,39,0,26]$, da imagem mais próxima para a mais distante, respectivamente, enquanto as distâncias entre os vértices, seguindo a mesma ordem, são de: $[0,286,0,34,0,374]$, $[0,295,0,338,0,367]$, $[0,286,0,346,0,368]$. Desse modo, fica claro como os atributos da figura são relacionadas ao formato que ela assume e não a fatores externos da imagem, mesmo considerando as pequenas variações provenientes da estocasticidade da inferência do detector de objetos, que acaba deslocando levemente o centroide das caixas delimitadoras.

Finalmente, os valores do perímetro e do semiperímetro foram removidos do vetor de atributos, uma vez que ambos assumem um valor fixo de 1 e 0.5, respectivamente, considerando o critério de normalização do triângulo. Com isso, um total de 11 características distintas foram obtidas do triângulo, gerando um vetor que foi utilizado como uma das entradas do classificador, junto às demais características dos segmentos de imagens extraídos pelo detector de objetos, conforme discutido em detalhes na Seção 4.1.5.

Com esta proposta, além de considerar todos os parâmetros linguísticos que formam os sinais, mitigando as margens para ambiguidades decorrentes da falta de informação discriminativa, o custo computacional tende a ser reduzido, pois as áreas pouco relevantes da imagem, como o plano de fundo, são substituídas por representações numéricas mais simples de serem

processadas.

Por fim, a técnica proposta possui duas principais limitações, sendo: 1) a incapacidade de representar o movimento em profundidade; e 2) a necessidade de que ambas as mãos do intérprete estejam visíveis durante toda a execução do sinal para possibilitar o cálculo dos atributos.

4.1.4.3 Figura do triângulo

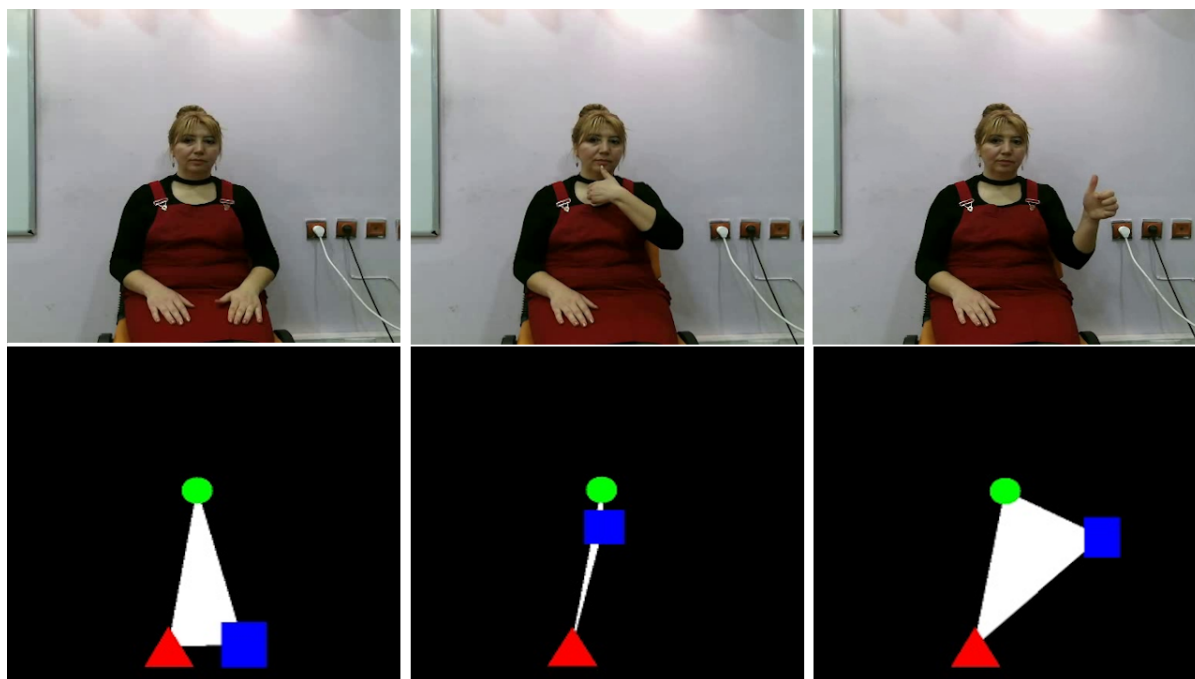
Conforme discutido na Seção 4.1.4.2, o trabalho de [Xiao et al. \(2020\)](#) é um dos que mais se assemelha à esta pesquisa, utilizando a informação referencial gerada pelas caixas delimitadoras do detector de objetos como um novo canal para auxiliar na discriminação dos sinais. Por conta disso, a técnica proposta pelos autores foi reproduzida por esta pesquisa, a fim de gerar um comparativo entre o uso da figura do triângulo em relação ao uso do vetor de atributos referenciais.

Ainda assim, algumas modificações foram realizadas na figura, com o objetivo de agregar maior valor semântico à mesma. Primeiro, o plano de fundo foi tratado como ausência de informação, gerando a cor preta através do preenchimento da imagem por zeros e, dessa forma, reduzindo a quantidade de informação a ser processada. Ademais, a área do triângulo foi preenchida pela cor branca, diferenciando-se em relação ao fundo da figura, enquanto as cores verde, vermelho e azul foram adotadas para identificar a face, e as mãos.

Além disso, as mesmas formas geométricas do trabalho original foram empregadas, sendo um círculo para a face e um triângulo e um quadrado para cada uma das mãos, geradas a partir da informação das caixas delimitadoras, da qual a única diferença é que o quadrado foi sempre atribuído à mão dominante, enquanto o triângulo foi utilizado para a mão não dominante, seguindo os critérios definidos na Seção 4.1.4.5. Para a criação do quadrado, por exemplo, apenas foi necessário preencher a caixa de uma das mãos com a cor designada, enquanto o triângulo, por sua vez, foi gerado a partir de duas novas arestas entre o ponto mínimo e máximo da aresta inferior com o ponto central da aresta superior da caixa delimitadora. A geração do círculo, por outro lado, foi realizada a partir do preenchimento do centroide da face por uma esfera com dimensão de 20 radianos. A forma resultante é exibida na Figura 4.9.

Conforme observado, a figura gerada distingue os elementos presentes na imagem não apenas pelo seu formato, mas também pela cor dos objetos, diferentemente do que foi realizado por [Xiao et al. \(2020\)](#), que utilizam as mesmas cores para todas as figuras e limitam a área do triângulo apenas por pequenas bordas vermelhas. Por fim, as figuras triangulares foram redimensionadas para a resolução de 128x128 *pixels* e utilizadas como um dos canais de informação

Figura 4.9: Figura do triângulo gerada a partir da execução dos sinais.



Fonte: Autor.

para classificação dos sinais, conforme detalhado na Seção 4.1.5.

4.1.4.4 Aumento de dados

Conforme discutido na Seção 2.3.6, o aumento de dados é de extrema importância para o contexto da classificação de imagens, promovendo a invariância do modelo. Seguindo esta tendência, diferentes transformações foram aplicadas aos segmentos de imagens das mãos e da face, sendo que as transformações possuem uma probabilidade de serem aplicadas durante o treinamento e cada imagem pode sofrer múltiplas transformações simultâneas.

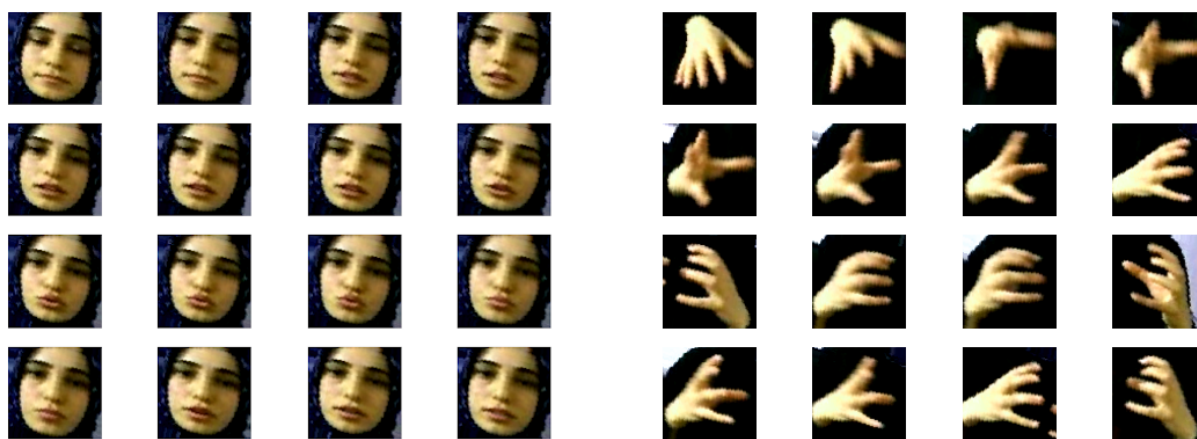
Dentre as transformações geométricas, foram aplicadas a rotação, cisalhamento, ampliação e desvio horizontal e vertical, sendo que cada operação possui um determinado intervalo de aplicação. A rotação horizontal, por exemplo, varia em torno de 20 graus (no sentido horário e anti-horário) para evitar que o sinal sofra alterações em sua semântica. Além disso, as operações de ampliação e desvio não foram aplicadas para o segmento das mãos, para evitar a possível perda de informação dos dedos do intérprete.

As transformações de intensidade, por sua vez, foram caracterizadas por alterações no brilho, contraste, *hue* e saturação da imagem. Ainda que a informação de cor (seja da pessoa, da vestimenta, ou do plano de fundo) não seja considerada um parâmetro linguístico para identificação do sinal (conforme discutido na Seção 1.1), é importante que o modelo seja invariante a

esse fator, visto que o conjunto de teste possui intérpretes não observados durante o treinamento, podendo prejudicar a performance do modelo caso o mesmo possua algum viés relacionado às cores.

A Figura 4.10 mostra um exemplo de aplicação de aumento de dados aleatório em um dos vídeos da base AUTSL. Conforme observado, as operações aplicadas são as mesmas tanto para a face quanto para as mãos por todos os quadros do vídeo.

Figura 4.10: Amostra de imagens da face e das mãos em um vídeo com aumento de dados aleatório.



Fonte: Autor.

4.1.4.5 Pré-processamento das imagens para classificação

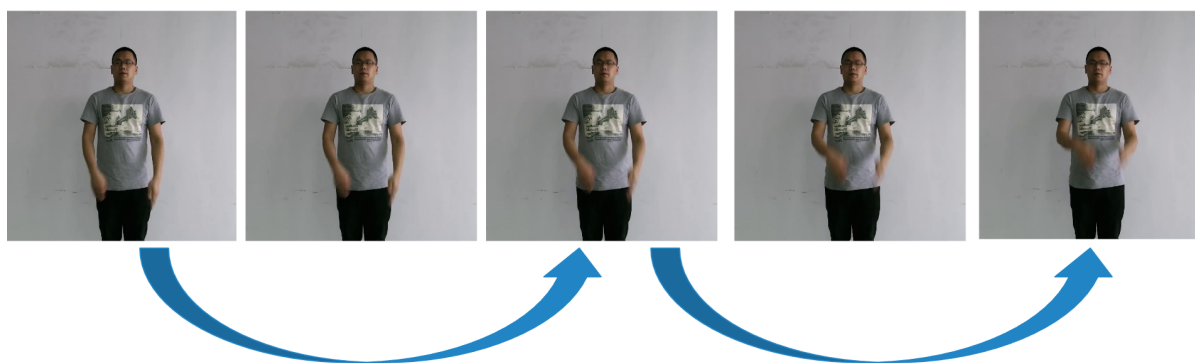
O objetivo da etapa de pré-processamento foi de iterar por cada um dos vídeos da base de dados a fim de extrair os quatro canais que compõe a arquitetura, sendo formados por ambas as mãos do intérprete, a face e as informações do triângulo. Além disso, diversos métodos e algoritmos foram implementados nesta etapa para garantir que o máximo de informação seja preservada para a classificação dos sinais.

Dessa forma, a cada iteração pelo conjunto de dados, um lote de 180 vídeos foram aleatoriamente selecionados e carregados sequencialmente para a memória. Feito isso, apenas 16 quadros de cada vídeo foram utilizados para representar a informação temporal dos sinais para o classificador, assim como observado no trabalho de [Zhang e Zhang \(2021\)](#). Embora ampliar o tamanho da sequência temporal minimize a perda de informação relevante no processo de subamostragem dos quadros, sequências mais longas podem aumentar consideravelmente a complexidade do treinamento das redes recorrentes, conforme discutido na Seção 2.2, além de elevar o custo computacional para geração de novas inferências pelo modelo. Por conta

disso, o processo de subamostragem foi cuidadosamente otimizado para conter o máximo de informação possível em apenas 16 quadros.

Para isso, o primeiro passo foi definir o valor de espaçamento utilizado como critério de amostragem, através da divisão do número total de quadros do vídeo por 16. Um vídeo com 39 quadros, por exemplo, resultaria em um espaçamento de 2, considerando o arredondamento para o menor valor, fazendo com que a cada quadro do vídeo selecionado outros dois subsequentes sejam descartados, garantindo uma amostragem igualmente espaçada pelo tempo, conforme ilustrado na Figura 4.11.

Figura 4.11: Uso do valor de espaçamento para amostragem dos quadros do vídeo.



Fonte: Autor.

Ainda assim, diversos problemas foram observados durante a subamostragem dos dados. Primeiro, os quadros iniciais e finais dos vídeos costumam ser compostos pela pose de descanso e finalização da execução do sinal, agregando pouca ou nenhuma informação discriminativa. Embora remover um número fixo de quadros do início e fim do vídeo possa resolver parte do problema, a quantidade de imagens descartáveis varia muito dependendo do intérprete, velocidade em que o sinal é executado e da própria gravação dos vídeos, podendo eliminar informações relevantes para a classificação.

Além disso, outro problema relevante foi percebido em sinais executados lentamente pelo intérprete ou com uma sequência de execução mais longa, como a palavra “HOSPITAL” no conjunto AUTSL, por exemplo, que é formada pelo sinal de “MÉDICO” seguido do sinal de “CONSTRUÇÃO”. Nesses casos, boa parte dos 16 quadros eram compostos por imagens similares e com pouca informação, visto que o intérprete tende a demorar na transição entre sinais ou na finalização do mesmo, mantendo as mãos estáticas.

Dito isso, como forma de atenuar os problemas supracitados, um critério para filtragem e seleção dos melhores quadros do vídeo foi estabelecido, baseado no histórico de movimento das

mãos durante a execução do sinal. Nessa abordagem, para cada um dos quadros selecionados, um valor de posição ρ foi calculado conforme definido na equação 4.9:

$$\rho = d_2 + d_3. \quad (4.9)$$

Pelo fato da cabeça do intérprete ser um ponto fixo durante a execução do sinal, a distância das mãos em relação à face pode ser utilizada como uma referência para o posicionamento espacial. Os valores das posições são armazenados em um vetor $\mathbf{p}$, representando o histórico de posições das mãos ao longo do vídeo. Feito isso, a partir do segundo quadro, um valor μ de movimento é calculado, conforme exibido na Equação 4.10:

$$\mu = |p_t - p_{t-1}|, \quad (4.10)$$

em que p_t representa a posição das mãos no instante de tempo t , enquanto p_{t-1} é a posição no quadro anterior do vídeo, no instante $t - 1$, fazendo com que o valor de movimento represente o grau de deslocamento das mãos entre um quadro e outro. Com isso, a informação de movimento é armazenada em um vetor $\mathbf{m}$ e utilizada como critério de seleção das imagens que irão compor o conjunto.

Para resolver o problema da pose de descanso, é necessário remover os quadros iniciais do vídeo com pouco ou nenhum movimento, por se tratar do início da execução do sinal e não possuir valor semântico. Dessa forma, após o primeiro quadro do vídeo, todos os demais são descartados até que o movimento das mãos exceda um limiar η , garantindo que o número correto de quadros sejam ignorados de acordo à especificidade de cada vídeo.

Além disso, após superar a pose de descanso, os três últimos intervalos de tempo do vetor de movimento são avaliados a cada iteração, e caso todos os valores de μ sejam menores que um limiar $\hat{\eta}$, o quadro é descartado. Essa abordagem objetiva resolver o problema gerado pelas transições lentas na execução dos sinais, eliminando quadros consecutivos que possuam muito pouco movimento e, conseqüentemente, sejam muito semelhantes uns dos outros. A Figura 4.12 ilustra os quadros iniciais de um dos vídeos da base AUTSL (em cima) e os quadros durante a execução do sinal (em baixo), exaltando os problemas tratados pelo critério de histórico de movimento. Por fim, os valores de η e $\hat{\eta}$ são relacionados à distância euclidiana e foram empiricamente definidos como 10 e 5, respectivamente, melhorando a amostragem dos quadros e evitando informações redundantes.

Figura 4.12: Quadros de diferentes intervalos de tempo (indicado pelos números em vermelho) de um dos vídeos da base AUTSL.



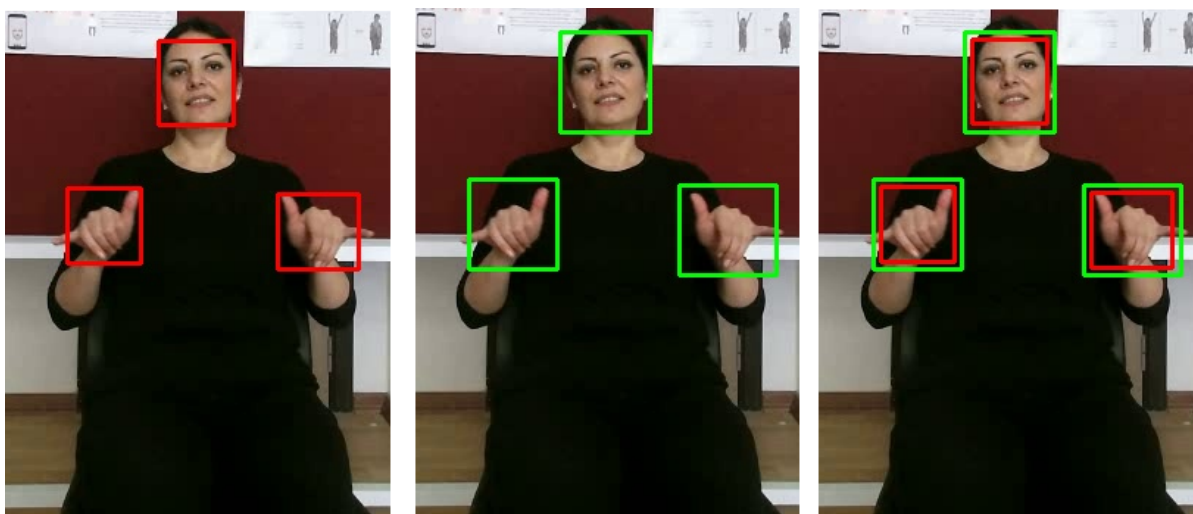
Fonte: Autor.

Em relação à técnica de detecção de objetos, executada para cada uma das imagens, foi empregado um limiar de 55% de confiança para aceitação das inferências, sendo que, dentre elas, a face e as duas mãos com maior confiança eram selecionadas, minimizando as chances de utilizar segmentos ruidosos na composição do conjunto de dados. Além disso, a área das caixas delimitadoras geradas pelo modelo foi aumentada em 10%, para evitar que parte dos dedos do intérprete saiam dos limites do segmento das mãos, conforme ilustrado na Figura 4.13.

A resolução das imagens submetidas ao detector foi padronizada em 512x512 *pixels*, fazendo com que seja possível utilizar os mesmos critérios de distância euclidiana para ambos os conjuntos de dados, ao passo que os canais das mãos e do rosto foram redimensionados para 100x100 *pixels* após serem segmentados. Ademais, a fim de evitar distorções nas imagens e reduzir o tamanho do plano de fundo dos quadros do conjunto CSL (que originalmente possuem a dimensão de 1280x720 *pixels*), as imagens foram cortadas antes de terem sua resolução reduzida, conforme ilustrado pela Figura 4.14.

Com os três segmentos obtidos, é possível calcular os atributos do triângulo e gerar a imagem da figura geométrica, conforme discutido nas Seções 4.1.4.2 e 4.1.4.3. Entretanto, caso uma das mãos ou a face não seja detectada no quadro do instante t , por sofrer de sobreposição, fugir dos limites da imagem, ou apenas por falha do detector, as detecções realizadas no instante

Figura 4.13: Comparação entre as caixas delimitadoras geradas pelo modelo (em vermelho) em relação ao aumento manual de 10% em seu tamanho (em verde).



Fonte: Autor.

Figura 4.14: Comparação entre um quadro original (à esquerda) e um quadro cortado (à direita) do conjunto CSL.



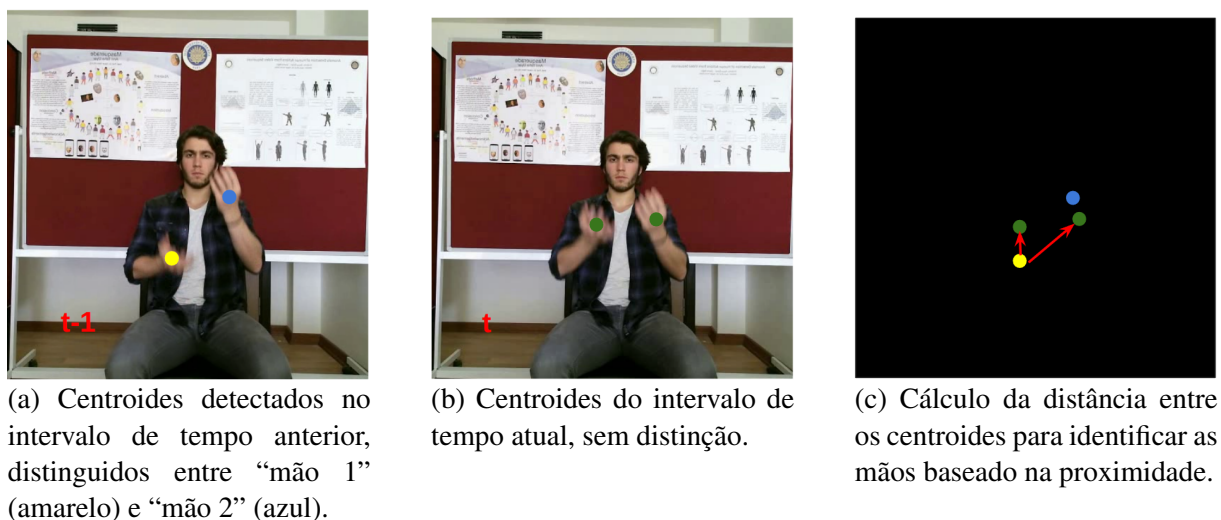
Fonte: Autor.

de tempo anterior são utilizadas em substituição ao que foi perdido para preencher a informação faltante, a menos que a mão não detectada seja a dominante, fazendo com que o quadro seja descartado.

Por conta disso, os segmentos das mãos foram diferenciados entre o segmento da mão dominante e o não dominante, servindo como critério para tomada de decisão mediante a erros do detector de objetos, geração da figura triangular (Seção 4.1.4.3) e também para unir a informação de entrada para o classificador (detalhado na Seção 4.1.5). Embora um mesmo sinal possa ser executado com a mão direita ou esquerda, a mão dominante é a que o intérprete adota como preferível e que costuma ser empregada por ele para executar a maioria dos sinais. Para que

essa diferenciação seja possível, o primeiro passo da abordagem é distinguir os objetos entre “mão 1” e “mão 2”, através da menor distância euclidiana entre os centroides do intervalo de tempo t e $t - 1$. Em outras palavras, após a geração das inferências do detector de objetos no intervalo de tempo t , a distância entre o centroide da “mão 1” (obtido em $t - 1$) e os dois novos centroides das mãos do intervalo atual é computada, sendo que o novo centroide mais próximo ao anterior recebe a classificação de “mão 1”, enquanto o mais distante é a “mão 2”, conforme ilustrado na Figura 4.15. Além disso, no primeiro quadro do vídeo, onde detecções anteriores não estão disponíveis, as mãos são categorizadas arbitrariamente.

Figura 4.15: Processo utilizado para diferenciação das duas mãos



Fonte: Autor.

Feito isso, um vetor de movimento é gerado para cada uma das mãos a partir de seu histórico de posicionamento, de maneira similar ao que foi discutido anteriormente, e a mão com o maior valor de movimento por todos os intervalos de tempo disponíveis é considerada como a dominante. Falhas do modelo em detectar a mão dominante não são toleradas pelo fato de que essa mão agrega muito valor semântico ao sinal sendo executado. Caso a falha seja nos canais que possuem menos alterações ao longo do tempo, como a mão não dominante ou a face, o intervalo de tempo anterior é utilizado em substituição. Ainda assim, caso a falha ocorra no primeiro intervalo de tempo, o mesmo é descartado por não possuir intervalos anteriores como referência. Em acréscimo, a informação de movimento das mãos foi salva e posteriormente agregada ao vetor de atributos do triângulo, visto que através dessa informação é possível saber a direção, magnitude e padrão de movimentação das mãos durante a execução do sinal, podendo auxiliar no processo de classificação.

Por fim, caso o vídeo tenha sido completamente explorado seguindo os critérios supracita-

dos e os 16 quadros alvo ainda não tenham sido amostrados (por falhas na detecção de objetos ou pelo não enquadramento dos critérios estabelecidos, por exemplo) a exploração do vídeo é reiniciada com um espaçamento menor, garantindo que os quadros não selecionados anteriormente se tornem candidatos do subconjunto. Mesmo com essa estratégia, é possível que o mínimo de 16 não seja atingido, principalmente em vídeos de mais curta duração ou ruidosos. Nesses casos, os quadros faltantes são preenchidos por zeros, gerando imagens pretas (no caso dos segmentos das mãos e face) ou um vetor com valores zero, no caso do triângulo. Como validação final, caso o vídeo possua mais de 5 dos 16 quadros preenchidos por zeros, o mesmo é descartado por possuir pouca informação disponível.

Após o pré-processamento, o lote contendo os 180 vídeos é comprimido em um arquivo binário em formato TFRecord (Bisong, 2019) e o processo é repetido até que todo o conjunto de dados seja preparado para o classificador.

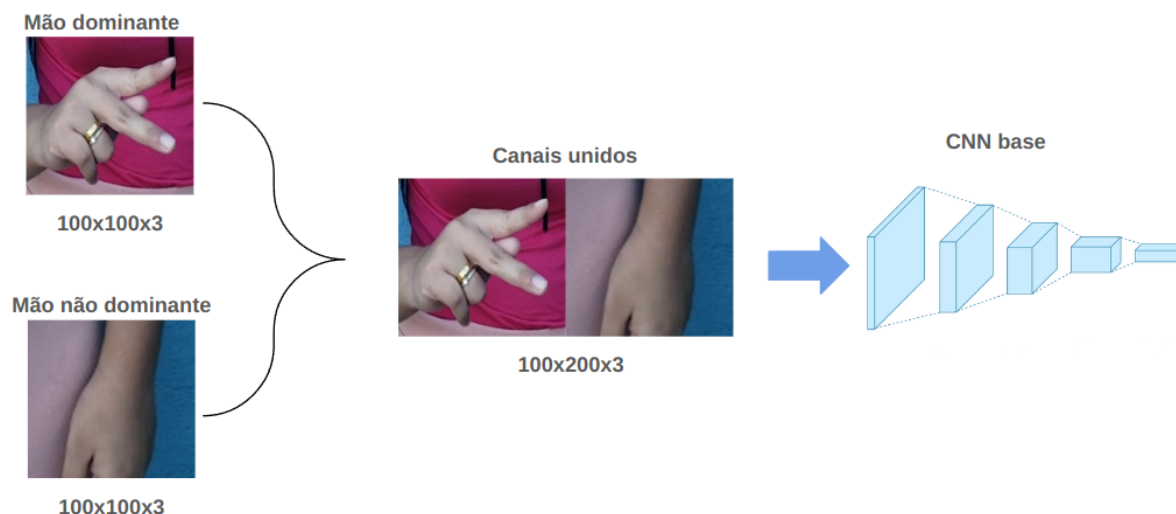
4.1.5 Sistema proposto para classificação dos sinais

A arquitetura do modelo proposto para classificação de sinais conta com diversas etapas e variações, a fim de conseguir processar a informação dos quatro canais distintos. Primeiro, pelo fato das imagens de entrada possuírem baixa resolução (100×100 pixels) e visando o mínimo de custo computacional, a arquitetura EfficientNetB0 (Tan e Le, 2019) foi empregada como extrator de características, por possuir uma das melhores compensações entre o número de parâmetros e a acurácia.

Além disso, como a mão esquerda e direita apresentam informações similares (visto que o sinal pode ser executado por qualquer uma das mãos), as imagens foram unidas horizontalmente, alterando sua dimensão de $100 \times 100 \times 3$ para $100 \times 200 \times 3$ pixels e, dessa forma, utilizando um único canal de CNN para extrair as suas características. Ademais, a informação da mão dominante foi utilizada para padronizar a junção das imagens, mantendo a mesma sempre de um único lado, conforme ilustrado na Figura 4.16.

A imagem da face, por sua vez, foi submetida diretamente à CNN base para extração de características. Enquanto o último canal, contendo os atributos do triângulo, não recebeu nenhum tipo de alteração significativa durante o treinamento, visto que os seus atributos foram todos calculados no pré-processamento do conjunto de dados, sendo apenas acrescido das informações de movimento das mãos computadas para definir a mão dominante (discutido na Seção 4.1.4.5), resultando em um vetor com 13 atributos. A Figura 4.17 ilustra a arquitetura do

Figura 4.16: Unificação das imagens das mãos em um único canal.



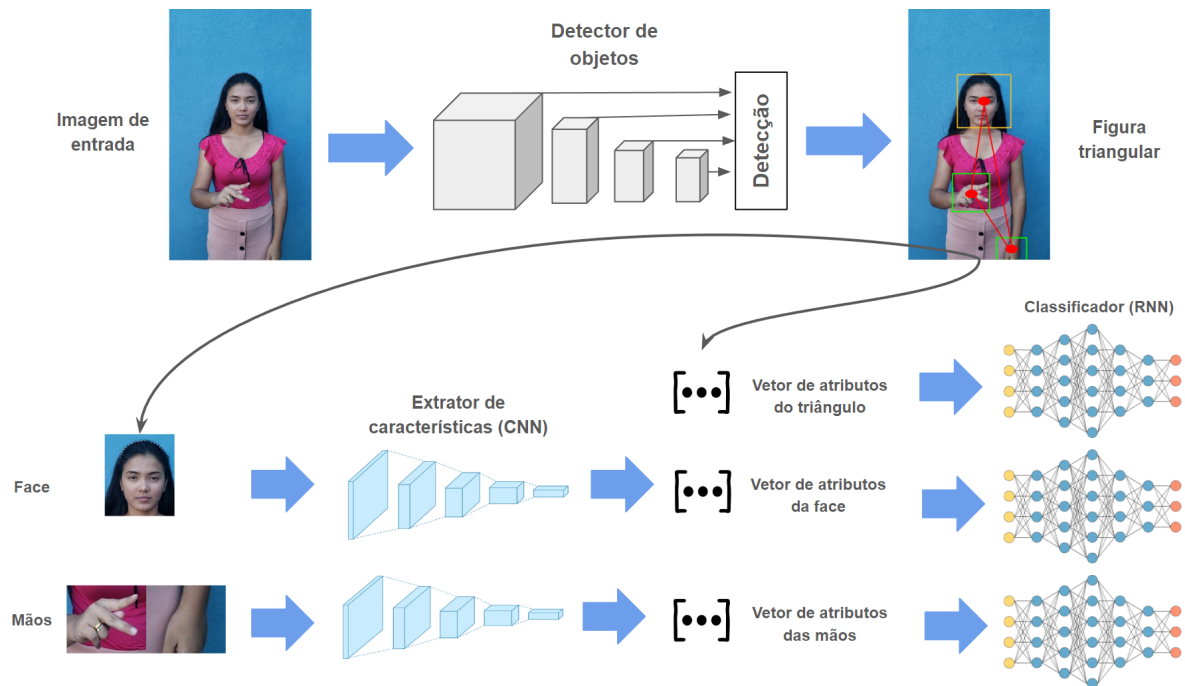
Fonte: Autor.

sistema proposto.

Conforme observado, as RNNs foram empregadas como classificador para o reconhecimento dos padrões temporais de cada um dos canais, compostos pelos vetores de atributos das mãos, face e do triângulo. Ainda assim, a escolha do tipo de célula recorrente, normalização, número de neurônios e quantidade de camadas ocultas que compõem os modelos preditivos é muito dependente do domínio do problema (Géron, 2017) e ocorreu de maneira empírica através de tentativa e erro, com a intenção de encontrar a arquitetura de rede neural com melhor resultado para a tarefa. Além disso, existem inúmeras possíveis abordagens para fusão desses três canais a fim de melhor explorar a informação disponível para a classificação, conforme descrito na Seção 2.3.10. Baseado nisso, a biblioteca Keras Tuner (O'Malley et al., 2019) foi empregada com o propósito de facilitar a busca pela melhor combinação de hiperparâmetros.

Para o funcionamento adequado da ferramenta, o primeiro passo necessário é criar um espaço de busca com valores mínimos e máximos que sejam razoáveis para cada hiperparâmetro, considerando o contexto do problema, além da definição dos possíveis tipos de neurônios, métodos de fusão, células recorrentes e entre outros parâmetros arquiteturais a serem testados. No trabalho em questão, três diferentes redes recorrentes foram criadas, uma para cada canal de informação, sendo que células do tipo LSTM ou GRU foram empregadas, podendo ou não ser bidirecionais e contendo ou não uma camada de atenção. Além disso, os intervalos do número de camadas, neurônios e probabilidade de *Dropout* variaram para cada rede neural, conforme exibido na Tabela 7.

Figura 4.17: Arquitetura proposta para o reconhecimento de sinais.



Fonte: Autor.

Tabela 7: Variação nos intervalos de hiperparâmetros para cada canal de RNN.

	Face	Mãos	Triângulo
Mínimo de camadas	1	1	1
Máximo de camadas	3	5	3
Mínimo de neurônios	64	32	64
Máximo de neurônios	448	512	320
Mínimo de <i>Dropout</i>	5%	5%	5%
Máximo de <i>Dropout</i>	40%	40%	40%

Fonte: Autor.

Ademais, três abordagens para a fusão das características extraídas pelos canais de RNN foram testadas para gerar a classificação do sinal executado. Na primeira, sendo uma fusão antecipada, os três vetores de características foram concatenados em um único vetor de maior dimensionalidade, e o mesmo foi submetido a um classificador Softmax, composto por uma única camada de rede neural Perceptron com a função de ativação supracitada. A segunda abordagem, por outro lado, foi baseada na fusão tardia, onde cada vetor é classificado individualmente, e a média das probabilidades é utilizada como forma de fusão dos resultados dos classificadores Softmax. Por fim, uma abordagem de meta-aprendizado foi adotada, unindo os três vetores de características e submetendo-os a uma nova rede MLP, a qual, por sua vez, possui de uma a duas camadas com 64 a 512 neurônios e com uma taxa de *Dropout* que varia entre

5% a 40%.

Considerando as configurações supracitadas, o algoritmo Hyperband (Li et al., 2018a) foi utilizado no Keras Tuner para o ajuste dos hiperparâmetros, visando o balanceamento entre a exploração do espaço de parâmetros e o tempo de execução da busca. Além disso, cada rede neural recorrente foi otimizada individualmente, garantindo que os resultados obtidos fossem os melhores possíveis para processar a informação do canal específico, desconsiderando a influência dos demais canais. Por fim, o máximo de 15 épocas foi estabelecido para o treinamento dos modelos durante a busca das melhores combinações, enquanto a métrica de sucesso foi a acurácia no conjunto de validação. Após a convergência do algoritmo de busca de hiperparâmetros, diversos modelos foram treinados para cada um dos canais de informação, conforme detalhado na Seção 4.2. Ademais, é válido ressaltar que a otimização de hiperparâmetros foi realizada apenas para o conjunto de dados AUTSL, enquanto a base CSL foi utilizada apenas para comprovar a validade das arquiteturas resultantes do processo de otimização.

4.1.6 Classificação dos sinais com a figura do triângulo

A figura do triângulo, por outro lado, teve seus atributos extraídos a partir da arquitetura MobileNetV2 (Sandler et al., 2018), sendo ainda mais simples e com menos parâmetros, motivado pelo fato de que a imagem possui poucos padrões a serem discriminados, composta apenas por algumas cores e formas geométricas. Após a obtenção do vetor de características, uma rede neural recorrente também foi otimizada para este canal por meio do Keras Tuner, conforme os hiperparâmetros apresentados na Tabela 8, porém, o mesmo método de fusão definido pelos critérios explicados na Seção 4.1.5 foi utilizado. Por fim, outra série de modelos foram treinados considerando a combinação com a informação da figura do triângulo, conforme explicado na Seção 4.2, servindo como comparativo direto ao método proposto por esta pesquisa.

Tabela 8: Variação nos intervalos de hiperparâmetros para cada canal de rede neural recorrente.

Fig. Triângulo	
Mínimo de camadas	1
Máximo de camadas	4
Mínimo de neurônios	32
Máximo de neurônios	512
Mínimo de <i>Dropout</i>	5%
Máximo de <i>Dropout</i>	40%

Fonte: Autor.

4.2 Metodologia experimental

Esta seção tem por objetivo detalhar os experimentos realizados para cada uma das contribuições esperadas pelo trabalho, listadas na Seção 1.4, especificando a metodologia para condução e avaliação dos experimentos.

Dito isto, no que tange a primeira proposta desta pesquisa, o experimento realizado girou em torno do treinamento dos diferentes modelos de detecção de objetos, seguindo as mudanças arquiteturais descritas na Seção 4.1.4.1 e utilizando o conjunto de dados criado para o contexto da língua de sinais. Além disso, todos os modelos foram treinados utilizando uma taxa de aprendizado inicial de 0,001 e um *batch size* de 16 por 80000 iterações, onde cada iteração representa a passagem de um *batch* de dados, com exceção do Faster-RCNN (por limitações de memória) que foi treinado com um *batch size* de 8 por 160000 iterações, fazendo com que todos os detectores sejam submetidos à mesma quantidade de imagens.

Como forma de avaliação do primeiro experimento, três métricas foram levadas em consideração, sendo elas o tempo médio de inferência em milissegundos por imagem, e os percentuais de mAP considerando os limiares de 50% e 75% de IOU, sendo chamados de mAP@50 e mAP@75, respectivamente. As métricas de mAP foram obtidas ao final do treinamento no conjunto de teste, evidenciando a taxa de sucesso dos detectores em intérpretes e cenários não observados anteriormente. Ainda assim, limiares mais rígidos (como 95%) não foram considerados, visto que a tarefa em questão não exige tamanha precisão na localização dos objetos. Por fim, um total de 200 vídeos do conjunto de teste da base AUTSL foram selecionados e o tempo de inferência para cada um dos quadros foi mensurado, utilizando tanto o processador como a placa de vídeo (especificados na Seção 4.1.1), gerando a média do tempo de inferência para cada um dos detectores de acordo ao *hardware* empregado.

O segundo experimento, por outro lado, visa investigar o impacto gerado pelo vetor de atributos do triângulo na acurácia de classificação, evidenciando a capacidade da técnica em representar o ponto de contato e movimento do sinal. Para isso, após o uso do Keras Tuner para encontrar o melhor conjunto de hiperparâmetros, diversos modelos foram treinados utilizando diferentes combinações dos canais existentes, desde a adoção de uma única entrada até a união de todas as entradas simultaneamente, resultando em 7 classificadores distintos.

Ademais, cada treinamento foi conduzido por um total de 40 épocas, um *batch size* de 30, uma taxa de aprendizado inicial de 0,001 e a função de otimização Adam, com exceção dos canais individuais das mãos e da face, que foram treinados em diversas etapas para garantir o

fine tuning adequado da CNN, assim como observado no trabalho de (Sarhan et al., 2022). A primeira etapa seguiu os mesmos parâmetros supracitados para o treinamento, e teve como objetivo o aquecimento inicial do modelo (conforme discutido na Seção 2.3.7), mantendo congelado todos os pesos da CNN que foram transferidos a partir do treinamento da base ImageNet e ajustando apenas a rede recorrente. As etapas posteriores, por sua vez, tiveram o objetivo de adaptar os parâmetros do extrator de características para o domínio do problema, contando com uma taxa de aprendizado que variou de 0,00001 até 0,001 através de um decaimento triangular e com o otimizador SGD, por apresentar melhores resultados do que o Adam para o *fine tuning* dos modelos. Ao total, 4 novas etapas de treinamento foram conduzidas, descongelando as camadas 221, 162, 119 e 75 da EfficientDet, sendo que as etapas possuíram as durações de 40, 35, 35 e 30 épocas.

O conjunto de dados CSL seguiu uma abordagem de treinamento similar à supracitada, alterando apenas a quantidade de etapas do *fine tuning* da CNN base de 4 para apenas 2, descongelando as camadas 221 e 162 com durações de 30 épocas. Tal redução foi motivada pela transferência de aprendizado entre os parâmetros dos modelos treinados no conjunto AUTSL, facilitando a convergência dos modelos por se tratar de um domínio similar. Por fim, as CNNs ajustadas nos canais individuais foram utilizadas no treinamento dos canais combinados.

Além disso, outros 4 classificadores foram gerados para este experimento, contando com a combinação da figura do triângulo com os demais canais de informação e servindo como comparativo entre a técnica proposta por esta pesquisa e a presente na literatura. O treinamento seguiu os mesmos parâmetros dos demais canais supracitados, porém, o ajuste da CNN foi realizado através de uma única etapa após o aquecimento do modelo, descongelando a camada 143 da MobileNet, uma vez que descongelar mais camadas gerou a degradação dos resultados.

Por fim, como forma de avaliar os resultados do segundo experimento, os valores de acurácia na base de teste foram obtidos para cada um dos modelos, além do tempo de inferência no processador e na placa de vídeo e a quantidade de parâmetros. O tempo de inferência em milissegundos foi obtido através da média do tempo de resposta do modelo para cada um dos vídeos (com 16 quadros) do conjunto de teste da base AUTSL, e junto à quantidade de parâmetros representam a complexidade e custo computacional do classificador.

4.3 Considerações finais do Capítulo

Conforme discorrido, o presente trabalho visa abordar algumas das lacunas presentes na área de reconhecimento de palavras em língua de sinais, como a exploração eficiente dos parâmetros linguísticos a fim de gerar classificadores menos complexos computacionalmente. Para que isso seja possível, propõe-se uma etapa de pré-processamento baseada em detectores de objetos simplificados, que deverão extrair as dicas visuais mais importantes da imagem (mãos e cabeça) e utilizá-las como entrada para o modelo. Além disso, a representação da referência espacial e do movimento das mãos do intérprete através de um simples vetor de atributos tem um grande potencial para reduzir a quantidade de informação a ser processada e simplificar o sistema, abstraindo detalhes pouco relevantes da imagem como o plano de fundo e demais partes do corpo do indivíduo. Por fim, a técnica presente na literatura que mais se assemelha à proposta desta pesquisa foi reproduzida a fim de gerar um comparativo entre as abordagens. Dito isso, o Capítulo 5 exibe os resultados obtidos por esta pesquisa, exaltando algumas das contribuições e objetivos atingidos pela mesma.

5 Resultados e Discussões

Este capítulo tem como objetivo expor os resultados obtidos por esta pesquisa a partir dos experimentos realizados, descritos em detalhes no Capítulo 4.2. Dito isso, a Seção 5.1 apresenta os resultados do treinamento dos detectores de objetos para as dicas visuais, enquanto a Seção 5.2 é dedicada à apresentação do desempenho obtido pelos classificadores utilizando diferentes canais de informação para o problema de reconhecimento de palavras em língua de sinais.

5.1 Resultados da detecção de objetos

O primeiro experimento realizado é também um pré-requisito para os demais, visto que os detectores de objetos são empregados na etapa de pré-processamento do conjunto de dados. Dito isso, os resultados obtidos a partir do treinamento dos detectores podem ser vistos na Tabela 9.

Tabela 9: Resultado do treinamento dos detectores de objetos.

Arquitetura	Tempo de inferência na CPU (ms)	Tempo de inferência na GPU (ms)	mAP@50	mAP@75
SSD640	53,2	11,6	98,5	95,0
SSD320	15,7	9,9	92,1	73,1
EfficientDet D0	67,8	16,1	96,7	85,8
YoloV7	123,9	7,4	98,6	95,7
Faster RCNN	281,0	26,3	99,0	96,2
CenterNet	40,0	7,9	99,0	96,7

Fonte: Autor.

Conforme esperado, todos os detectores atingem um desempenho bastante satisfatório na tarefa de classificar e localizar as dicas visuais, considerando a simplicidade da tarefa, ficando acima dos 85% de mAP, sendo o SSD320 a única exceção por se tratar de um modelo com baixa resolução de entrada e desempenho inferior para objetos pequenos na imagem. Em relação ao EfficientDet, ainda que seja uma arquitetura mais moderna, possua uma CNN base com mais parâmetros e camadas e que faça uso de pirâmides de características bidirecionais, seus resultados foram inferiores ao SSD640, evidenciando que modelos complexos podem não ser fundamentais para obter os melhores resultados em problemas mais simples.

O YoloV7, por sua vez, foi superior ao SSD por uma pequena margem, mostrando a efetividade das novas versões da arquitetura em questão. Entretanto, não foi capaz de superar o Faster RCNN, o qual também foi superior a todos os detectores supracitados em ambos os limiares de IOU, visto que a complexidade deste modelo é dada pelas múltiplas propostas e caixas âncoras geradas durante as inferências, garantindo maiores taxas de acerto, mesmo para objetos de menor proporção. Por fim, a abordagem da CenterNet se mostrou a mais bem sucedida para a detecção das dicas visuais, possivelmente pelo fato de que as cenas são pouco populadas e os objetos não costumam ficar muito próximos uns dos outros, garantindo uma taxa mais alta de sucesso na estimativa dos centroides e de suas caixas a partir dos mapas de calor gerados pelo modelo.

Ademais, conforme explicado na Seção 4.1.4, os detectores em questão foram simplificados para apresentar o mínimo de custo computacional possível enquanto mantêm o desempenho próximo à versão original. Como forma de elucidar o trabalho de otimização desenvolvido, a Tabela 10 apresenta o tempo de inferência dos detectores sem as mudanças propostas.

Tabela 10: Desempenho obtido pelos detectores de objetos não simplificados.

Arquitetura	Tempo de inferência na CPU (ms)	Tempo de inferência na GPU (ms)
SSD640	108,0	44,1
SSD320	32,7	25,7
EfficientDet D0	124,5	53,4
Faster RCNN	811,5	79,1
YoloV7	211,1	7,6

Fonte: Autor.

Como é possível notar, a maioria dos modelos apresentou uma redução significativa de tempo de inferência mediante às simplificações, começando pelas arquiteturas mais complexas, como o Faster RCNN e o EfficientDet, reduzindo o tempo na geração das previsões na GPU em 66,75% e 69,8%, respectivamente, até as arquiteturas baseadas no SSD, com reduções de 73,69% e 61,4%, considerando as versões com 640x640 e 320x320 *pixels* de imagem de entrada, respectivamente. Em relação ao YoloV7, as operações de convolução que compõem a arquitetura foram cuidadosamente otimizadas para a computação em GPU, gerando uma diferença notável entre os dois componentes de *hardware*. Dessa forma, mesmo a versão otimizada

do modelo, que contou com uma redução de mais de 61% no número de parâmetros, não apresentou uma diferença significativa no tempo de inferência na GPU. Ainda assim, ao considerar um dispositivo de *hardware* mais simples, como uma CPU, é possível observar um tempo de inferência 41,3% menor em relação ao modelo original. Além disso, o menor número de parâmetros também reduz consideravelmente os requisitos de memória do modelo, viabilizando a adoção do mesmo em dispositivos mais limitados. Por fim, o CenterNet se mostrou uma das arquiteturas mais eficientes para a tarefa, visto que o mesmo conseguiu obter a melhor compensação entre tempo de inferência e mAP sem necessitar de grandes simplificações em seu modo de operação.

5.2 Resultados dos atributos *handcrafted*

Em relação ao segundo experimento, o primeiro passo foi a otimização dos hiperparâmetros dos canais de informação a fim de encontrar as melhores arquiteturas para extração de características, gerando redes recorrentes com as especificações exibidas na Tabela 11. Conforme observado, a maior parte dos modelos obtiveram melhor convergência com menos de três camadas e com a adoção de células do tipo GRU e camadas de atenção, evidenciando a efetividade das técnicas para o contexto do problema. Além disso, a abordagem de fusão antecipada foi a escolhida através do Keras Tuner para realizar a combinação dos canais, pois a mesma se mostrou mais efetiva para a tarefa. Uma vez que as arquiteturas foram definidas, é possível treinar os diferentes modelos para classificação dos sinais, alcançando os resultados exibidos na Tabela 12.

Tabela 11: Arquiteturas geradas para cada uma das redes recorrentes propostas.

Hiperparâmetros	Mãos	Face	Triângulo	Fig. Triângulo
Bidirecional	Sim	Não	Não	Sim
Atenção	Sim	Sim	Sim	Sim
Célula Recorrente	GRU	GRU	GRU	GRU
Dropout	25%	5%	10%	25%
Neurônios da camada 1	288	384	320	288
Neurônios da camada 2	-	320	192	416
Neurônios da camada 3	-	192	-	-

Fonte: Autor.

Em relação aos canais individuais, as mãos possuem a maior quantidade de parâmetros linguísticos, conforme discutido no Capítulo 1.1, gerando melhores resultados de acurácia para

Tabela 12: Comparação dos resultados dos modelos no conjunto de dados AUTSL.

Canal utilizado	Acurácia de teste	Inferência na GPU (ms)	Inferência na CPU (ms)	Parâmetros
Mãos	71,83%	42	110	6.893.525
Triângulo	46,85%	28	28	661.490
Figura do triângulo	54,96%	37	69	7.641.074
Face	24,49%	38	69	6.986.453
Mãos + Face	68,88%	54	153	13.879.752
Mãos + Triângulo	79,79%	44	118	7.554.789
Mãos + Figura do triângulo	76,06%	53	154	14.534.373
Face + Triângulo	54,69%	41	70	7.647.717
Face + Figura do triângulo	55,74%	52	114	14.627.301
Mãos + Face + Triângulo	80,72%	55	153	14.541.016
Mãos + Face + Figura do triângulo	74,72%	65	199	21.520.600

Fonte: Autor.

discriminação dos sinais, enquanto a face, por si só, não é capaz de distinguir os sinais efetivamente, por contar apenas com as marcações não manuais. O canal do triângulo, por sua vez, mesmo sendo composto por apenas 13 valores numéricos, consegue classificar corretamente quase metade das amostras de teste, destacando o sucesso dos atributos *handcrafted* em representar o movimento e o ponto de contato do sinal. Ainda assim, a figura do triângulo foi individualmente superior ao simples vetor de atributos, acrescentando a acurácia de teste em 8,11%.

No que tange à união dos canais, entretanto, é possível verificar com maior exatidão como cada um dos atributos contribuiu para a discriminação dos sinais, uma vez que canais com características similares apresentam pouco ou nenhum ganho de acurácia quando concatenados. Baseado nisso, fica evidente que as características das mãos e dos atributos do triângulo são complementares na representação dos parâmetros linguísticos, gerando um aumento de aproximadamente 8% de acurácia. Embora a união do triângulo com a face também tenha gerado ganhos nos resultados, é possível notar uma intersecção entre os atributos do canal das mãos e da face, provavelmente pelo fato de que, em alguns sinais, as mãos ficam próximas ou sobrepõem o rosto do intérprete, compondo os quadros do canal da face e influenciando nas características

extraídas pela CNN, conforme exibido na Figura 5.1. Por sua vez, unir os três canais de dados (mãos, face e atributos do triângulo) gerou um aumento de menos de 1%, demonstrando a falta de impacto da informação proveniente da face, não só pela intersecção dos atributos supracitada, mas também pelo fato de que o conjunto de dados em questão possui poucos sinais com expressão facial acentuada.

Figura 5.1: Intersecção visual entre os canais da face e das mãos.



Fonte: Autor.

Ao contrário do que foi observado no canal individual da figura do triângulo, o padrão de melhoria dos resultados não foi mantido na união com os demais canais, tornando notável a menor efetividade da imagem da figura geométrica em representar os parâmetros linguísticos precisamente, além da dificuldade de se concatenar atributos extraídos por modelos mais complexos de aprendizado profundo. Em acréscimo, também é perceptível o aumento no número de parâmetros e no tempo de inferência dos modelos com a união deste canal, mesmo com a adoção da MobileNetV2 (2.257.984 milhões de parâmetros) em relação à EfficientNetB0 (4.049.571 milhões de parâmetros) para extrair os atributos da figura geométrica. Vale ressaltar ainda que o treinamento inicial da figura foi realizado com o modelo EfficientNetB0, porém, a acurácia do canal individual foi similar à MobileNet, justificando a escolha do modelo mais simples e eficiente. Dito isso, embora a figura do triângulo tenha resultado em um acréscimo de 4,23% de acurácia ao ser unida com o canal das mãos, o custo computacional extra pode não justificar a adoção de tal abordagem, ainda mais considerando a superioridade do uso dos

atributos *handcrafted*, atingindo uma acurácia 3,73% maior. Além disso, a diferença no número de parâmetros e tempo de inferência de cada um dos canais deixa claro que a exploração eficiente dos parâmetros linguísticos possibilita a obtenção de bons resultados sem a necessidade de modelos muito complexos, sendo de extrema importância para a área de reconhecimento de sinais.

Baseado nisso, os resultados no conjunto AUTSL validam a técnica desenvolvida por esta pesquisa para a representação do ponto de contato e do movimento do sinal a partir de um vetor de atributos extraídos manualmente, atingindo mais de 80% de acurácia em um conjunto de dados desafiador com diversos intérpretes, planos de fundo e condições de iluminação. Ainda que outros trabalhos no estado da arte tenham obtido melhores resultados (conforme mostrado no Capítulo 3.2), a maior parte deles utilizam arquiteturas complexas, compostas por métodos de extração de pose corporal e pela união das previsões de diversos modelos distintos, enquanto a técnica desenvolvida por este trabalho gerou um aumento de 7,96% de acurácia com menos de 670 mil parâmetros e acréscimos de apenas 2 e 8 milissegundos no tempo de inferência na GPU e CPU, respectivamente. Além disso, os 80,72% de acurácia obtidos na pesquisa são superiores aos 60,37% utilizados como linha de base pelos criadores do conjunto AUTSL (Sincan e Keles, 2020), reforçando o potencial e validade do sistema proposto.

Como finalização do segundo experimento, entretanto, todos os modelos supracitados foram retreinados no conjunto de dados CSL, a fim de verificar o desempenho do sistema proposto para o problema de reconhecimento de sinais, independentemente da língua nativa ou da base de dados. A princípio, como forma de demonstrar os ganhos gerados pela transferência de aprendizado entre problemas de domínio similar, o canal individual das mãos teve dois treinamentos iniciais, em que o primeiro contou com os parâmetros transferidos a partir do conjunto ImageNet, enquanto o segundo recebeu os parâmetros aprendidos na base AUTSL, resultando em uma acurácia de 68,25% e 75,85%, respectivamente. Dessa forma, fica evidente que a transferência de aprendizado entre canais similares é eficaz para este tipo de problema, facilitando a convergência dos modelos para outros idiomas. Motivado pelas melhorias expressivas, os canais da face e da figura do triângulo também contaram com a transferência a partir da base AUTSL, e os resultados de cada um dos classificadores podem ser vistos na Tabela 13.

Conforme observado, os canais individuais da face, dos atributos do triângulo e da figura do triângulo apresentaram um desempenho similar ao que foi mostrado no conjunto AUTSL, sendo o maior destaque para o canal das mãos, com uma acurácia aproximadamente 11% maior.

Tabela 13: Comparação dos resultados dos modelos no conjunto de dados CSL.

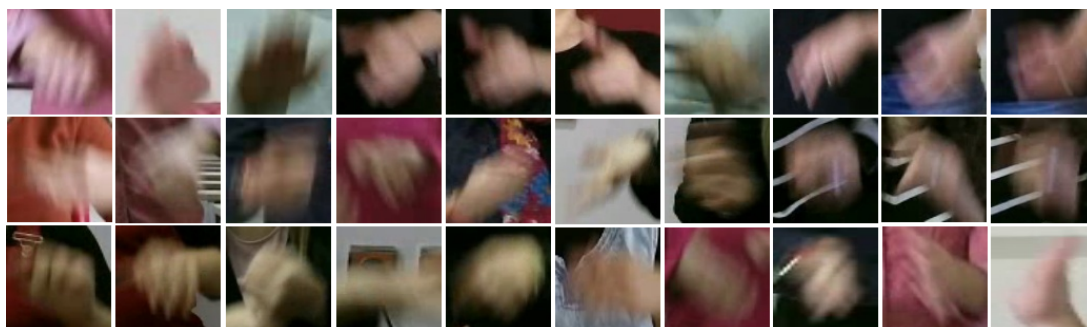
Canal utilizado	Acurácia de teste	Inferência na GPU (ms)	Inferência na CPU (ms)	Parâmetros
Mãos	82,79%	42	109	7.051.623
Triângulo	51,00%	28	28	714.372
Figura do triângulo	59,94%	38	67	7.869.316
Face	21,77%	40	71	7.039.335
Mãos + Face	80,03%	55	151	14.090.458
Mãos + Triângulo	82,82%	45	117	7.765.495
Mãos + Figura do triângulo	83,07%	55	162	14.920.439
Face + Triângulo	54,68%	42	78	7.753.207
Face + Figura do triângulo	62,90%	50	110	14.908.151
Mãos + Face + Triângulo	82,29%	56	156	14.804.330
Mãos + Face + Figura do triângulo	82,15%	68	202	21.959.274

Fonte: Autor.

Embora seja necessário um estudo mais aprofundado para determinar a razão exata para o aumento de acurácia com a informação das mãos, dois fatores podem ter contribuído para estes resultados, sendo a execução mais lenta dos sinais e a maior resolução dos vídeos. Ao comparar o tempo médio de duração dos vídeos, o conjunto CSL apresenta um aumento de 1,12 segundos, tanto pela presença de sinais mais longos quanto pela execução mais lenta por parte dos intérpretes, atenuando o efeito de borrão de movimento, o qual é acentuado no conjunto AUTSL, conforme exibido na Figura 5.2. Além disso, os vídeos do conjunto CSL possuem originalmente uma resolução superior ao AUTSL, fazendo com que mais detalhes das mãos sejam preservados após a segmentação e redimensionamento do canal. Dito isso, é possível concluir que conjuntos de dados gravados com maior resolução de imagem e com os devidos cuidados e técnicas para redução de borrão de movimento podem se beneficiar ainda mais da exploração das dicas visuais.

A união dos canais, por outro lado, não foi efetiva no conjunto CSL, atingindo valores de acurácia similares ou inferiores mesmo com a agregação de mais informação para o classificador. Conforme discutido anteriormente, é esperado que o canal da face gere poucas melhorias nos resultados devido à intersecção com as mãos e pela ausência de sinais dependentes das

Figura 5.2: Efeito do borrão de movimento no canal das mãos do conjunto AUTSL.



Fonte: Autor.

marcações não manuais, que também é o caso na base CSL. Além disso, o comportamento da figura do triângulo também foi similar ao visto anteriormente, não sendo muito efetivo mediante à união com outros canais. Ainda assim, mesmo a utilização dos atributos do triângulo não foi capaz de melhorar a discriminação entre os sinais.

Ao analisar minuciosamente os erros produzidos pelo classificador, é possível notar alguns dos fatores que podem ter contribuído para estes resultados. Primeiro, destaca-se a existência de sinais muito similares, equiparando-se a uma condição de polissemia e sendo responsável pela maior parte das inferências incorretas. A Figura 5.3, por exemplo, ilustra a similaridade entre os sinais de “MASCULINO” e “JOALHERIA”, diferindo-se apenas por um leve movimento realizado ao segurar uma das orelhas, confundindo as previsões do modelo entre as duas classes.

Figura 5.3: Sinal de “MASCULINO” (em cima) e “JOALHERIA” (em baixo) na base de dados CSL.



Fonte: Autor.

De igual modo, a Figura 5.4 apresenta outro exemplo desse tipo, onde é possível notar que os sinais de “BOLINHO FRITO” e “SORVETE” são distinguidos apenas pela repetição do sutil movimento próximo à boca do intérprete. Em casos como esses, a discriminação entre os sinais é complexa, mesmo com o uso dos atributos do triângulo, visto que os padrões de movimento e

o ponto de contato são similares. Um fator sutil que pode ajudar na diferenciação das palavras é o movimento em profundidade, presente quando as mãos do intérprete se mantêm nas mesmas coordenadas espaciais mas se deslocam para frente e para trás. Contudo, os atributos *hand-crafted* extraídos a partir do triângulo não são capazes de capturar essa informação de maneira adequada.

Figura 5.4: Sinal de “BOLINHO FRITO” (em cima) e “SORVETE” (em baixo) na base de dados CSL.



Fonte: Autor.

Em acréscimo, outro detalhe que pode influenciar nas previsões incorretas do modelo é a presença de sinais mais longos, compostos por múltiplos gestos subsequentes, como é o caso das 50 classes com maior quantidade de erros, por exemplo, onde o tempo médio de execução é de 3,57 segundos, muito superior à média do próprio conjunto CSL, que é de 3,17 segundos, aumentando a dificuldade de representar toda a informação disponível em apenas 16 quadros.

Além disso, também existem erros provenientes de falhas do extrator de características, onde o movimento e ponto de contato são idênticos, mas o formato das mãos se diferem, e do detector de objetos, em que algumas roupas estampadas confundem as previsões do mesmo. Para ambos os casos, novas arquiteturas de modelos podem ser testadas a fim de obter melhores resultados. Finalmente, também é possível considerar que o Keras Tuner, utilizado para otimização dos hiperparâmetros das RNNs, gerou arquiteturas ótimas apenas para o conjunto AUTSL, sendo necessário novas execuções da ferramenta para a base CSL.

Por fim, o número de parâmetros e o tempo de inferência dos novos modelos são praticamente idênticos ao conjunto AUTSL, visto que a única alteração arquitetural se deu no aumento de neurônios da camada Softmax para representar a maior quantidade de classes disponíveis. Baseado nisso, embora mais de 80% dos sinais tenham sido corretamente classificados, a maior parte dos erros gerados no conjunto CSL são provenientes de situações onde os atributos do

triângulo não são capazes de serem discriminativos, como sinais com movimentos semelhantes. Ainda assim, diversas melhorias podem ser realizadas em trabalhos futuros para tornar a abordagem proposta por esta pesquisa ainda mais acurada e eficiente, conforme detalhado a seguir no Capítulo 6.

6 Conclusões

A língua de sinais é uma ferramenta fundamental de comunicação para os deficientes auditivos, permitindo sua integração com a sociedade. Ainda que possua regras semânticas complexas e ambiguidades, assim como na língua falada, um conjunto de apenas 6 parâmetros formam a estrutura de todos os sinais das línguas de sinais ao redor do mundo, facilitando a sistematização da mesma. Baseado nisso, é notório que muitos trabalhos recentes têm surgido com o intuito de realizar o reconhecimento e tradução das palavras em língua de sinais para língua nativa, graças aos avanços das técnicas e arquiteturas modernas de aprendizado profundo (vistas no Capítulo 2) e ao poder computacional disponível. Ainda assim, um longo caminho precisa ser percorrido até que se alcance um sistema com capacidade prática de apoio ao deficiente auditivo em suas atividades cotidianas.

Dito isso, as principais pesquisas e técnicas presentes na literatura utilizadas para o pré-processamento, extração de características e classificação de sinais foram levantadas no Capítulo 3, evidenciando que apenas uma parcela ínfima de trabalhos se preocupam com a exploração eficiente dos parâmetros linguísticos a fim de reduzir a complexidade das soluções desenvolvidas e facilitar sua adoção fora do ambiente laboratorial. Portanto, este trabalho apresentou, no Capítulo 4, uma proposta de arquitetura para o reconhecimento de sinais que visa, de maneira eficiente e com o menor custo computacional, extrair e explorar as dicas visuais por meio de detectores de objetos e atributos *handcrafted*, realizando a classificação das palavras em conjuntos de dados distintos.

Dessa forma, os experimentos propostos foram realizados e detalhados no Capítulo 5 a fim de tentar validar as hipóteses e questionamentos levantados por esta pesquisa. Assim, o primeiro questionamento realizado foi: É possível obter êxito na extração de dicas visuais com baixo custo computacional? Conforme discutido, as modificações dos modelos de detecção de objetos permitiram gerar reduções significativas em seus tempos de inferência, chegando a até 73,69% e atingindo um máximo de 135 FPS (*Frames Per Second*) na GPU (YoloV7) e 63 FPS, na CPU (SSD320). Dito isso, é possível concluir que a modificação dos modelos existentes na literatura é uma abordagem válida, pois permite adequar o nível de complexidade das arquiteturas ao problema sendo tratado, gerando melhor compensação entre custo computacional e acurácia. Além disso, algumas arquiteturas apresentam melhor desempenho na tarefa do que outras, devido às especificidades do funcionamento de cada detector, fazendo com que seja necessária a condução de experimentos similares aos que foram demonstrados por esta pesquisa a

fim de encontrar o melhor modelo.

Assim sendo, além de garantir que a etapa de pré-processamento seja realizada com o mínimo de esforço computacional, futuras pesquisas podem se basear nos resultados obtidos para criar modelos ainda mais simplificados para desempenhar a localização e segmentação da face e das mãos do intérprete, consolidando este tipo de técnica para o contexto do problema. Além disso, um novo conjunto de dados de larga escala foi criado especificamente para as condições comumente observadas na língua de sinais, contando com mais de 477 mil faces e 954 mil mãos anotadas, podendo ser utilizado como base para o treinamento de novos detectores de mãos e faces de domínio similar.

O segundo questionamento, por sua vez, é relacionado à hipótese de que é possível representar o ponto de contato e o movimento dos sinais através de um vetor de atributos *handcrafted*. De acordo aos resultados obtidos, é possível notar que os atributos extraídos a partir da referência espacial das dicas visuais mostraram uma contribuição relevante para a classificação, gerando um aumento de 7,96% na acurácia do conjunto AUTSL com o mínimo de impacto na performance do modelo, destacando que, para o reconhecimento de sinais, entender e representar os parâmetros que dão semântica aos gestos é tão importante quanto utilizar modelos cada vez mais complexos. No conjunto CSL, por sua vez, embora a união dos canais e o uso dos atributos do triângulo não tenham gerado melhorias nos resultados, uma análise minuciosa destacou que os sinais incorretos não seriam facilmente discriminados apenas pela informação proposta de posição espacial e movimento, relevando um viés no conjunto de dados em questão. Assim, espera-se que futuras pesquisas possam utilizar os atributos supracitados para outros conjuntos de dados e também para a área de reconhecimento contínuo de sinais.

Como melhorias para trabalhos futuros, recomenda-se: i) treinar outras arquiteturas de detecção de objetos existentes na literatura, ou até mesmo a criação de novos modelos voltados para este tema, maximizando a eficiência da tarefa; ii) anotar novas imagens para compor e aumentar a variedade do conjunto de dados criado nesta pesquisa; iii) utilizar técnicas de extração de pontos-chaves (*keypoints*) da face (Guerra et al., 2018), garantindo um mapeamento mais preciso do rosto e das expressões faciais dos intérpretes e eliminando a intersecção com as mãos; iv) estudar de maneira mais profunda as propriedades das figuras bidimensionais para tentar propor atributos do triângulo ainda mais discriminativos.

Por fim, todo o código fonte, conjunto de dados e modelos desenvolvidos por esta pesquisa se encontram devidamente documentados e disponíveis publicamente, além de possuir um ar-

tigo publicado na ICDIP (*International Conference of Digital Image Processing*) [Carneiro et al. \(2021\)](#), aumentando o nível de contribuição com a comunidade científica. Embora o trabalho proposto também não tenha se estendido para fora do ambiente laboratorial, o sistema criado pode ser utilizado como base para que trabalhos futuros da área consigam mitigar algumas das dificuldades em implantar uma solução acessível e funcional, facilitando efetivamente o cotidiano dos deficientes auditivos.

Referências Bibliográficas

- ABADI, M.; BARHAM, P.; CHEN, J.; CHEN, Z.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; IRVING, G.; ISARD, M.; ET AL. Tensorflow: a system for large-scale machine learning. In: *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Savannah, GA, USA, 2016, p. 265–283.
- AGARAP, A. F. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.
- AGRAWAL, T.; UROLAGIN, S. 2-Way Arabic Sign Language Translator Using CNNLSTM Architecture and NLP. In: *Proceedings of the 2020 2nd International Conference on Big Data Engineering and Technology*, Association for Computing Machinery, 2020, p. 96—101.
- ALAWNEH, L.; MOHSEN, B.; AL-ZINATI, M.; SHATNAWI, A.; AL-AYYOUB, M. A comparison of unidirectional and bidirectional lstm networks for human activity recognition. In: *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, IEEE, 2020, p. 1–6.
- DE AMORIM, C. C.; MACÊDO, D.; ZANCHETTIN, C. Spatial-temporal graph convolutional networks for sign language recognition. In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*, Cham: Springer International Publishing, 2019, p. 646–657.
- BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In: *3rd International Conference on Learning Representations, ICLR*, San Diego, CA, USA, 2015.
- BAMBACH, S.; LEE, S.; CRANDALL, D. J.; YU, C. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2015, p. 1949–1957.
- BELISSEN, V.; GOUIFFÈS, M.; BRAFFORT, A. Automatic recognition of Sign Language structures in RGB videos: the detection of pointing and lexical signs. *Hal preprint hal-02146368f*, 2019.
- BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, v. 5, n. 2, p. 157–166, 1994.

BÉNYI, Á. 87.47 a heron-type formula for the triangle. *The Mathematical Gazette*, v. 87, n. 509, p. 324–326, 2003.

BISONG, E. Tensorflow 2.0 and keras. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Springer, p. 347–399, 2019.

BOUKDIR, A.; BENADDY, M.; ELLAHYANI, A.; MESLOUHI, O. E.; KARDOUCHI, M. Isolated Video-Based Arabic Sign Language Recognition Using Convolutional and Recursive Neural Networks. *Arabian Journal for Science and Engineering*, p. 1–13, 2021.

BRADSKI, G. The openCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, v. 25, n. 11, p. 120–123, 2000.

CAMGÖZ, N. C.; KINDIROĞLU, A. A.; KARABÜKLÜ, S.; KELEPIR, M.; ÖZSOY, A. S.; AKARUN, L. BosphorusSign: A Turkish Sign Language Recognition Corpus in Health and Finance Domains. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, Portorož, Slovenia: European Language Resources Association (ELRA), 2016, p. 1383–1388.

CAO, Z.; HIDALGO, G.; SIMON, T.; WEI, S.-E.; SHEIKH, Y. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 43, n. 1, p. 172–186, 2019.

CAPOVILLA, F. C.; RAPHAEL, W. D.; MAURICIO, A. C. L. *Novo Deit-Libras dicionário enciclopédico ilustrado trilingue da Língua de Sinais Brasileira (Libras) baseado em Linguística e Neurociências Cognitivas*. INEP, 2800 p., 2009.

CARNEIRO, A. C.; SILVA, L. B.; SALVADEO, D. P. Efficient sign language recognition system and dataset creation method based on deep learning and image processing. In: *Thirteenth International Conference on Digital Image Processing (ICDIP 2021)*, International Society for Optics and Photonics, 2021, p. 1187803.

CARREIRA, J.; ZISSERMAN, A. Quo vadis, action recognition? a new model and the kinetics dataset. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, p. 6299–6308.

DE CASTRO, G. Z.; GUERRA, R. R.; DE ASSIS, M. M.; REZENDE, T. M.; DE ALMEIDA, G. T.; ALMEIDA, S. G.; DE CASTRO, C. L.; GUIMARAES, F. G. Desenvolvimento de uma

base de dados de sinais de Libras para aprendizado de máquina: Estudo de caso com CNN 3D. In: *Anais do 14th Simpósio Brasileiro de Automação Inteligente*, 2019, p. 2116–2121.

CHAI, X.; WANG, H.; CHEN, X. The devisign large vocabulary of chinese sign language database and baseline evaluations. *Technical report VIPL-TR-14-SLR-001. Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS*, 2014.

CHO, K.; VAN MERRIËNBOER, B.; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, 2014, p. 1724–1734.

CHOLLET, F.; ET AL. keras, 2015. Disponível em: <<https://keras.io/>>. Acessado em: 5 de jan. de 2023.

CHUNG, J.; GULCEHRE, C.; CHO, K.; BENGIO, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

CRUZ, A. R. D. S. Uma estratégia para reconhecimento de sinais de Língua Brasileira de Sinais utilizando aprendizado profundo. 2020. 78 f. Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas, Amazonas, Manaus. 2020.

CUBUK, E. D.; ZOPH, B.; MANE, D.; VASUDEVAN, V.; LE, Q. V. Autoaugment: Learning augmentation strategies from data. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, p. 113–123.

DAS, N.; OHN-BAR, E.; TRIVEDI, M. M. On performance evaluation of driver hand detection algorithms: Challenges, dataset, and metrics. In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, 2015, p. 2953–2958.

DE COSTER, M.; VAN HERREWEGHE, M.; DAMBRE, J. Sign language recognition with transformer networks. In: *12th International Conference on Language Resources and Evaluation*, European Language Resources Association (ELRA), 2020, p. 6018–6024.

DE COSTER, M.; VAN HERREWEGHE, M.; DAMBRE, J. Isolated Sign Recognition from RGB Video using Pose Flow and Self-Attention. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, p. 3441–3450.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, p. 248–255.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, 2019, p. 4171–4186.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, n. 7, 2011.

EBLING, S.; CAMGÖZ, N. C.; BRAEM, P. B.; TISSI, K.; SIDLER-MISEREZ, S.; STOLL, S.; HADFIELD, S.; HAUG, T.; BOWDEN, R.; TORNAY, S.; ET AL. SMILE Swiss German sign language dataset. In: *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC) 2018*, The European Language Resources Association (ELRA), 2018.

ELMAN, J. L. Finding structure in time. *Cognitive Science*, v. 14, n. 2, p. 179–211, 1990.

EVERINGHAM, M.; VAN GOOL, L.; WILLIAMS, C. K.; WINN, J.; ZISSERMAN, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, v. 88, n. 2, p. 303–338, 2010.

FANG, H.-S.; XIE, S.; TAI, Y.-W.; LU, C. Rmpe: Regional multi-person pose estimation. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2017, p. 2334–2343.

FEDERAÇÃO MUNDIAL DOS SURDOS. Position Paper regarding the United Nations Convention on the Rights of People with Disabilities, 2003. Disponível em: <<https://www.un.org/esa/socdev/enable/rights/contrib-wfd.htm>>. Acessado em: 25 de abr. de 2021.

FORSTER, J.; SCHMIDT, C.; HOYOUX, T.; KOLLER, O.; ZELLE, U.; PIATER, J. H.; NEY, H. RWTH-PHOENIX-Weather: A Large Vocabulary Sign Language Recognition and Translation Corpus. In: *International Conference on Language Resources and Evaluation*, 2012, p. 3785–3789.

GE, R.; KAKADE, S. M.; KIDAMBI, R.; NETRAPALLI, P. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. *Advances in Neural Information Processing Systems*, v. 32, 2019.

GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. 1 ed. O'Reilly Media, 574 p., 2017.

GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2015, p. 1440–1448.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, 2010, p. 249–256.

GÖKÇE, Ç.; ÖZDEMİR, O.; KINDIROĞLU, A. A.; AKARUN, L. Score-level multi cue fusion for sign language recognition. In: *European Conference on Computer Vision*, Springer International Publishing, 2020, p. 294–309.

GONZALEZ, R. C.; WOODS, R. C. *Processamento digital de imagens*, v. 3. Pearson Education, 624 p., 2009.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*, v. 1. MIT Press Cambridge, 800 p., 2016.

GOYAL, P.; DOLLÁR, P.; GIRSHICK, R.; NOORDHUIS, P.; WESOLOWSKI, L.; KYROLA, A.; TULLOCH, A.; JIA, Y.; HE, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

GRAVES, A.; SCHMIDHUBER, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, v. 18, n. 5-6, p. 602–610, 2005.

GRUBER, I.; KRNOUL, Z.; HRUZ, M.; KANIS, J.; BOHACEK, M. Mutual Support of Data Modalities in the Task of Sign Language Recognition. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, p. 3424–3433.

- GUERRA, R. R.; REZENDE, T. M.; GUIMARAES, F. G.; ALMEIDA, S. G. M. Facial expression analysis in brazilian sign language for sign recognition. In: *Anais do XV Encontro Nacional de Inteligência Artificial e Computacional*, SBC, 2018, p. 216–227.
- GURNEY, K. *An introduction to neural networks*. 1 ed. CRC Press, 248 p., 1997.
- HAN, J.; SHAO, L.; XU, D.; SHOTTON, J. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics*, v. 43, n. 5, p. 1318–1334, 2013.
- HAN, X.; LU, F.; TIAN, G. Efficient 3D CNNs with knowledge transfer for sign language recognition. *Multimedia Tools and Applications*, v. 81, n. 7, p. 10071–10090, 2022a.
- HAN, X.; LU, F.; YIN, J.; TIAN, G.; LIU, J. Sign Language Recognition Based on R (2+ 1) D With Spatial–Temporal–Channel Attention. *IEEE Transactions on Human-Machine Systems*, v. 52, n. 4, p. 687–698, 2022b.
- HARRIS, C. R.; MILLMAN, K. J.; VAN DER WALT, S. J.; GOMMERS, R.; VIRTANEN, P.; COURNAPEAU, D.; WIESER, E.; TAYLOR, J.; BERG, S.; SMITH, N. J.; ET AL. Array programming with NumPy. *Nature*, v. 585, n. 7825, p. 357–362, 2020.
- HE, K.; GKIOXARI, G.; DOLLÁR, P.; GIRSHICK, R. Mask r-cnn. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2017, p. 2961–2969.
- 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*, 2016, p. 770–778.
- HINTON, G.; SRIVASTAVA, N.; SWERSKY, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. 2012.
- HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997.
- HOSAIN, A. A.; SANTHALINGAM, P. S.; PATHAK, P.; RANGWALA, H.; KOSECKA, J. Hand Pose Guided 3D Pooling for Word-Level Sign Language Recognition. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, p. 3429–3439.

HU, H.; ZHOU, W.; PU, J.; LI, H. Global-local enhancement network for NMF-aware sign language recognition. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, v. 17, n. 3, p. 1–19, 2021.

HUANG, J.; RATHOD, V.; SUN, C.; ZHU, M.; KORATTIKARA, A.; FATHI, A.; FISCHER, I.; WOJNA, Z.; SONG, Y.; GUADARRAMA, S.; ET AL. Speed/accuracy trade-offs for modern convolutional object detectors. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, p. 7310–7311.

HUANG, J.; ZHOU, W.; ZHANG, Q.; LI, H.; LI, W. Video-Based Sign Language Recognition Without Temporal Segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 32, n. 1, 2018.

IBGE. Estatísticas de Gênero, 2010. Disponível em: <<https://www.ibge.gov.br/apps/snig/v1/?loc=0&cat=-1,-2,-3,128&ind=4648>>. Acessado em: 25 de abr. de 2021.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2015, p. 448–456.

JADERBERG, M.; SIMONYAN, K.; ZISSERMAN, A.; ET AL. Spatial transformer networks. *Advances in Neural Information Processing Systems*, v. 28, 2015.

JIA, Y.; SHELHAMER, E.; DONAHUE, J.; KARAYEV, S.; LONG, J.; GIRSHICK, R.; GUADARRAMA, S.; DARRELL, T. Caffe: Convolutional architecture for fast feature embedding. In: *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014, p. 675–678.

JIN, S.; MA, X.; HAN, Z.; WU, Y.; YANG, W.; LIU, W.; QIAN, C.; OUYANG, W. Towards multi-person pose tracking: Bottom-up and top-down methods. In: *ICCV PoseTrack Workshop*, 2017, p. 7.

JOZE, H. R. V.; KOLLER, O. MS-ASL: A Large-Scale Data Set and Benchmark for Understanding American Sign Language. In: *Proceedings of the British Machine Vision Conference (BMVC)*, 2019, p. 41.1–41.16.

KAY, W.; CARREIRA, J.; SIMONYAN, K.; ZHANG, B.; HILLIER, C.; VIJAYANARASIMHAN, S.; VIOLA, F.; GREEN, T.; BACK, T.; NATSEV, P.; ET AL. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

- KETKAR, N.; MOOLAYIL, J.; KETKAR, N.; MOOLAYIL, J. Introduction to pytorch. *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch*, p. 27–91, 2021.
- KINDIROGLU, A. A.; OZDEMIR, O.; AKARUN, L. Temporal accumulative features for sign language recognition. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, IEEE Computer Society, 2019, p. 1288–1297.
- KING, D. E. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, v. 10, p. 1755–1758, 2009.
- KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. In: *3rd International Conference on Learning Representations, ICLR*, San Diego, CA, USA, 2015.
- KRATIMENOS, A.; PAVLAKOS, G.; MARAGOS, P. Independent Sign Language Recognition with 3d Body, Hands, and Face Reconstruction. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, p. 4270–4274.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, v. 25, p. 1097–1105, 2012.
- LI, D.; RODRIGUEZ, C.; YU, X.; LI, H. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, p. 1459–1469.
- LI, L.; JAMIESON, K.; DESALVO, G.; ROSTAMIZADEH, A.; TALWALKAR, A. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, v. 18, n. 185, p. 1–52, 2018a.
- LI, Y.; HUANG, H.; XIE, Q.; YAO, L.; CHEN, Q. Research on a surface defect detection algorithm based on MobileNet-SSD. *Applied Sciences*, v. 8, n. 9, p. 1678, 2018b.
- LIAO, Y.; XIONG, P.; MIN, W.; MIN, W.; LU, J. Dynamic sign language recognition based on video sequence with BLSTM-3D residual networks. *IEEE Access*, v. 7, p. 38044–38054, 2019.
- LIN, M.; CHEN, Q.; YAN, S. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

LIN, T. LabelImg, 2015. Disponível em: <<https://github.com/heartexlabs/labelImg>>. Acessado em: 5 de jan. de 2023.

LIN, T.-Y.; DOLLÁR, P.; GIRSHICK, R.; HE, K.; HARIHARAN, B.; BELONGIE, S. Feature pyramid networks for object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, p. 2117–2125.

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: *European Conference on Computer Vision*, Springer International Publishing, 2014, p. 740–755.

LIU, T.; ZHOU, W.; LI, H. Sign language recognition with long short-term memory. In: *IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016a, p. 2871–2875.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; BERG, A. C. Ssd: Single shot multibox detector. In: *European Conference on Computer Vision*, Springer International Publishing, 2016b, p. 21–37.

MACLIN, R.; OPITZ, D. An empirical evaluation of bagging and boosting. In: *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97*, Providence, Rhode Island, USA, 1997, p. 546–551.

MAGALHÃES, G. I. Reconhecimento de gestos da língua brasileira de sinais através de redes neurais. 2018. 126 f. Dissertação (Mestrado em Ciência da Computação) - Instituto Tecnológico de Aeronáutica, São Paulo. 2018.

MARTINEZ-ALPISTE, I.; GOLCARENARENJI, G.; WANG, Q.; ALCARAZ-CALERO, J. M. Smartphone-based real-time object recognition architecture for portable and constrained systems. *Journal of Real-Time Image Processing*, v. 19, n. 1, p. 103–115, 2022.

MARUYAMA, M.; GHOSE, S.; INOUE, K.; ROY, P. P.; IWAMURA, M.; YOSHIOKA, M. Word-level sign language recognition with Multi-stream Neural Networks Focusing on Local Regions. *The Institute of Electronics, Information and Communication Engineers (IEICE) Technical Report.*, v. 120, n. 409, p. 53–58, 2021.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, v. 5, n. 4, p. 115–133, 1943.

- MCKINNEY, W.; ET AL. pandas: a foundational Python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, v. 14, n. 9, p. 1–9, 2011.
- MEKALA, P.; GAO, Y.; FAN, J.; DAVARI, A. Real-time sign language recognition based on neural network architecture. In: *2011 IEEE 43rd Southeastern Symposium on System Theory*, IEEE, 2011, p. 195–199.
- MENDONÇA, D. M.; MILITÃO, T. Pares polissêmicos em libras. *Revista eletrônica do ISAT (Instituto Superior Anísio Teixeira)*, v. 10, p. 45–57, 2018.
- MENG, L.; LI, R. An Attention-Enhanced Multi-Scale and Dual Sign Language Recognition Network Based on a Graph Convolution Network. *Sensors*, v. 21, n. 4, p. 1120, 2021.
- MOHAIMENIANPOUR, S.; VAUGHAN, R. Hands and faces, fast: mono-camera user detection robust enough to directly control a UAV in flight. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, p. 5224–5231.
- MURPHY, K. P. *Machine learning: a probabilistic perspective*. 1 ed. MIT press, 1104 p., 2012.
- NEIDLE, C.; THANGALI, A.; SCLAROFF, S. Challenges in development of the american sign language lexicon video dataset (asllvd) corpus. In: *5th Workshop on the Representation and Processing of Sign Languages: Interactions Between Corpus and Lexicon, LREC*, European Language Resources Association (ELRA), 2012, p. 143–150.
- NOBLE, W. S. What is a support vector machine? *Nature Biotechnology*, v. 24, n. 12, p. 1565–1567, 2006.
- NWANKPA, C.; IJOMAH, W.; GACHAGAN, A.; MARSHALL, S. Activation functions: comparison of trends in practice and research for deep learning. In: *2nd International Conference on Computational Sciences and Technology, (INCCST)*, 2021, p. 124 – 133.
- O'MALLEY, T.; BURSZEIN, E.; LONG, J.; CHOLLET, F.; JIN, H.; INVERNIZZI, L.; ET AL. KerasTuner, 2019. Disponível em: <<https://github.com/keras-team/keras-tuner>>. Acessado em: 02 de nov. de 2022.
- ORGANIZAÇÃO MUNDIAL DA SAÚDE (OMS). Deafness and hearing loss, 2021. Disponível em: <<https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss>>. Acessado em: 25 de abr. de 2021.

- ÖZDEMİR, O.; KINDIROĞLU, A. A.; CAMGÖZ, N. C.; AKARUN, L. BosphorusSign22k Sign Language Recognition Dataset. In: *Proceedings of the LREC2020 9th Workshop on the Representation and Processing of Sign Languages: Sign Language Resources in the Service of the Language Community, Technological Challenges and Application Perspectives*, Marseille, France: European Language Resources Association (ELRA), 2020, p. 181–188.
- PAPINENI, K.; ROUKOS, S.; WARD, T.; ZHU, W.-J. Bleu: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002, p. 311–318.
- PAVLAKOS, G.; CHOUTAS, V.; GHORBANI, N.; BOLKART, T.; OSMAN, A. A.; TZIONAS, D.; BLACK, M. J. Expressive body capture: 3d hands, face, and body from a single image. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, p. 10975–10985.
- POLYAK, B. T. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, v. 4, n. 5, p. 1–17, 1964.
- POTTER, L. E.; ARAULLO, J.; CARTER, L. The leap motion controller: a view on sign language. In: *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, 2013, p. 175–178.
- PREETHAM, C.; RAMAKRISHNAN, G.; KUMAR, S.; TAMSE, A.; KRISHNAPURA, N. Hand talk-implementation of a gesture recognizing glove. *Proceedings of Texas Instruments India Educators' Conference, TIIEC 2013*, p. 328–331, 2013.
- PU, J.; ZHOU, W.; LI, H. Sign language recognition with multi-modal features. In: *Pacific Rim Conference on Multimedia*, Springer, 2016, p. 252–261.
- 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*, 2016, p. 779–788.
- REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, v. 28, p. 91–99, 2015.

- ROBUSTO, C. C. The cosine-haversine formula. *The American Mathematical Monthly*, v. 64, n. 1, p. 38–40, 1957.
- RONCHETTI, F.; QUIROGA, F.; ESTREBOU, C. A.; LANZARINI, L. C.; ROSETE, A. LSA64: an Argentinian sign language dataset. In: *XXII Congreso Argentino de Ciencias de la Computación (CACIC)*, 2016, p. 794–803.
- ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386, 1958.
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986.
- RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATY, A.; KHOSLA, A.; BERNSTEIN, M.; ET AL. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, v. 115, n. 3, p. 211–252, 2015.
- SAGI, O.; ROKACH, L. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, v. 8, n. 4, p. e1249, 2018.
- 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*, 2018, p. 4510–4520.
- SARHAN, N.; LAURI, M.; FRINTROP, S. Multi-phase Fine-Tuning: A New Fine-Tuning Approach for Sign Language Recognition. *KI-Künstliche Intelligenz*, v. 36, n. 1, p. 91–98, 2022.
- SCHUSTER, M.; PALIWAL, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, v. 45, n. 11, p. 2673–2681, 1997.
- SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. In: *3th International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- SINAGA, K. P.; YANG, M.-S. Unsupervised K-means clustering algorithm. *IEEE Access*, v. 8, p. 80716–80727, 2020.

SINCAN, O. M.; JUNIOR, J.; JACQUES, C.; ESCALERA, S.; KELES, H. Y. Chalearn LAP large scale signer independent isolated sign language recognition challenge: Design, results and future research. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, p. 3472–3481.

SINCAN, O. M.; KELES, H. Y. Autsl: A large scale multi-modal turkish sign language dataset and baseline methods. *IEEE Access*, v. 8, p. 181340–181355, 2020.

SINCAN, O. M.; KELES, H. Y. Using Motion History Images with 3D Convolutional Networks in Isolated Sign Language Recognition. *IEEE Access*, v. 10, p. 18608–18618, 2022.

SMITH, L. N. Cyclical learning rates for training neural networks. In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2017, p. 464–472.

SNOEK, C. G.; WORRING, M.; SMEULDERS, A. W. Early versus late fusion in semantic video analysis. In: *Proceedings of the 13th Annual ACM International Conference on Multimedia*, 2005, p. 399–402.

SRIDHAR, A.; GANESAN, R. G.; KUMAR, P.; KHAPRA, M. Include: A large scale dataset for indian sign language recognition. In: *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, p. 1366–1375.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, v. 15, n. 1, p. 1929–1958, 2014.

SUHARJITO; GUNAWAN, H.; THIRACITTA, N.; NUGROHO, A. Sign Language Recognition Using Modified Convolutional Neural Network Model. In: *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, 2018, p. 1–5.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, p. 1–9.

TALUKDAR, A. K.; BHUYAN, M. Vision-Based Continuous Sign Language Spotting using Gaussian Hidden Markov Model. *IEEE Sensors Letters*, v. 6, n. 7, p. 1–4, 2022.

TAN, M.; LE, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In: *International Conference on Machine Learning*, PMLR, 2019, p. 6105–6114.

TAN, M.; PANG, R.; LE, Q. V. Efficientdet: Scalable and efficient object detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, p. 10781–10790.

TRAN, D.; BOURDEV, L.; FERGUS, R.; TORRESANI, L.; PALURI, M. Learning spatio-temporal features with 3d convolutional networks. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2015, p. 4489–4497.

TRAN, D.; WANG, H.; TORRESANI, L.; RAY, J.; LECUN, Y.; PALURI, M. A closer look at spatiotemporal convolutions for action recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, p. 6450–6459.

TU, Z. E. A. Multi-stream CNN: Learning representations based on human-related regions for action recognition. *Pattern Recognition*, v. 79, p. 32–43, 2018.

TUNGA, A.; NUTHALAPATI, S. V.; WACHS, J. P. Pose-based Sign Language Recognition using GCN and BERT. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, p. 31–40.

VAN ROSSUM, G.; ET AL. Python Programming Language. In: *USENIX Annual Technical Conference*, Santa Clara, CA, 2007, p. 1–36.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. *Advances in Neural Information Processing Systems*, v. 30, 2017.

VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; HABERLAND, M.; REDDY, T.; COURNAPEAU, D.; BUROVSKI, E.; PETERSON, P.; WECKESSER, W.; BRIGHT, J.; ET AL. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, v. 17, n. 3, p. 261–272, 2020.

WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

WANG, F.; DU, Y.; WANG, G.; ZENG, Z.; ZHAO, L. (2+ 1) D-SLR: an efficient network for video sign language recognition. *Neural Computing and Applications*, p. 1–11, 2021.

WERBOS, P. Beyond regression: New tools for prediction and analysis in the behavioral sciences. Tese (Doutorado em Matemática Aplicada) - Harvard University, Cambridge, Massachusetts, 1974.

XAVIER, A. N.; BARBOSA, P. A. Diferentes pronúncias em uma língua não sonora? Um estudo da variação na produção de sinais da libras. *DELTA Documentação de Estudos em Linguística Teórica e Aplicada*, v. 30, n. 2, p. 371–413, 2014.

XIAO, Q.; CHANG, X.; ZHANG, X.; LIU, X. Multi-information spatial-temporal LSTM fusion continuous sign language neural machine translation. *IEEE Access*, v. 8, p. 216718–216728, 2020.

XIAO, S.; FANG, Y.; NI, L. Multi-modal Sign Language Recognition with Enhanced Spatiotemporal Representation. In: *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021, p. 1–8.

XU, B.; WANG, N.; CHEN, T.; LI, M. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

YAN, S.; XIONG, Y.; LIN, D. Spatial temporal graph convolutional networks for skeleton-based action recognition. In: *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, p. 7444–7452.

ZAMANLOOY, B.; MIRHASSANI, M. Efficient VLSI implementation of neural networks with hyperbolic tangent activation function. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, v. 22, n. 1, p. 39–48, 2013.

ZHANG, S.; ZHANG, Q. Sign language recognition based on global-local attention. *Journal of Visual Communication and Image Representation*, v. 80, p. 103280, 2021.

ZHOU, X.; WANG, D.; KRÄHENBÜHL, P. Objects as Points. In: *arXiv preprint arXiv:1904.07850*, 2019.