FACULDADE DE TECNOLOGIA DE ADAMANTINA
TECNOLOGIA EM CIÊNCIA DE DADOS

ANDERSON LINS
LUIS HENRIQUE TURRA RAMOS

USO DE REDES NEURAIS CONVOLUCIONAIS PARA O RECONHECIMENTO DE LIBRAS

Adamantina – SP

2023

FACULDADE DE TECNOLOGIA DE ADAMANTINA
TECNOLOGIA EM CIÊNCIA DE DADOS

ANDERSON LINS
LUIS HENRIQUE TURRA RAMOS

USO DE REDES NEURAIS CONVOLUCIONAIS PARA O RECONHECIMENTO DE LIBRAS

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Adamantina, como requisito parcial para obtenção do diploma de Tecnólogo em Ciência de Dados. Sob a orientação do Prof. Dr. Paulo Roberto da Silva Ruiz.

Adamantina – SP

2023

S6778

RAMOS, Luis Henrique Turra.
Uso de redes neurais convolucionais para o reconhecimento de libras
Anderson Lins, Luis Henrique Turra Ramos. – Adamantina: FATEC, 2023.

Orientador: Prof. Dr. Paulo Roberto da Silva Ruiz.
Trabalho de Graduação (Curso de Tecnologia em Ciencia de Dados)
– Faculdade de Tecnologia de Adamantina, 2023.

1. Rede Neurais. 2. Língua de Sinais. 3. Acessibilidade. I. Lins, Anderson II. Ramos, Luis Henrique Turra. III. Ruiz, Prof. Dr. Paulo Roberto da Silva. IV. Faculdade de Tecnologia de Adamantina. V. Título.

ANDERSON LINS
LUIS HENRIQUE TURRA RAMOS

USO DE REDES NEURAIS CONVOLUCIONAIS PARA O RECONHECIMENTO DE LIBRAS

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Adamantina, como requisito parcial para obtenção do diploma de Tecnólogo em Ciência de Dados.

Aprovação em: 23/06/2023.

_____

Prof. Dr. Paulo Roberto da Silva Ruiz

Fatec de Adamantina

Adamantina – SP

Orientador

_____

Prof. Esp. Celso Rodrigo Dias Gualdi

Fatec de Adamantina

Adamantina – SP

Avaliador

_____

Prof. Dr. Ronnie Shida Marinho

Fatec de Adamantina

Adamantina – SP

Avaliador

RESUMO

Este trabalho tem por objetivo avaliar uma arquitetura de Rede Neural Convolucional na tarefa de classificação de sinais do alfabeto em LIBRAS. A base de dados utilizada conta com 15600 imagens divididas em 26 diretórios onde 25 deles contêm um sinal referente a uma letra do alfabeto em LIBRAS, exceto Z, por ser constituído de movimento. O vigésimo sexto diretório refere-se a classe de sinais não identificado. As imagens foram obtidas por meio das bibliotecas *MediaPipe* e *CVZone*, ambas disponíveis em *Python*. Na primeira camada da Rede Neural Convolucional foram adicionados 26 atributos referentes as classes das imagens e em seguida foram adicionadas camadas do *MobileNetV3-Large*, totalizando 263 camadas. Os dados foram divididos em 84,25% para treinamento e 15,75% para validação do modelo. Os resultados mostram que a maior parte das classes apresentaram 100% de predição correta, sendo 18 classes. Já em 7 classes mais de 50% foram preditas corretamente e apenas uma classe apresentou índice de acerto abaixo de 50%. Por fim, o treinamento que gerou o modelo de classificação obteve acurácia de 88,9%, entretanto o modelo foi utilizado para realizar a classificação de um conjunto de dados diferente daquele utilizado no treinamento, mas que sofreu o mesmo pré-processamento, alcançando uma acurácia de 93%.

Palavras-chave: Rede Neurais; Língua de Sinais; Acessibilidade.

ABSTRACT

This work aims to evaluate a Convolutional Neural Network architecture in the task of classifying signs of the alphabet in LIBRAS. The database used has 15600 images divided into 26 directories where 25 of them contain a sign referring to a letter of the alphabet in LIBRAS, except Z, as it consists of movement. The twenty-sixth directory refers to the unidentified signal class. The images were obtained using the MediaPipe and CVZone libraries, both available in Python. In the first layer of the Convolutional Neural Network, 26 attributes referring to the image classes were added and then MobileNetV3-Large layers containing 263 layers were added. Data were divided into 84.25% for training and 15.75% for model validation. The results show that most of the classes presented 100% correct prediction, being 18 classes. In 7 classes more than 50% were correctly predicted and only one class had a hit rate below 50%. Finally, the training that generated the classification model obtained an accuracy of 88.9%, however the model was used to perform the classification of a data set different from that used in the training, but which underwent the same pre-processing, reaching a 93% accuracy.

Keywords: Neural Networks; Sign Language; Accessibility.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS

ADAM *Adaptative Moment Estimation* (Estimativa do Momento Adaptativo)

CPU *Central Processing Unity* (Unidade Central de Processamento)

CVZone *Computer Vision Zone* (Zona de Visão Computacional)

GPU *Graphics Processing Unit* (Unidade de Processamento Gráfico)

HDF5 *Hierarchical Data Format version 5* (Formato dos Dados Hierárquicos – versão 5)

HS H-swish

KDD *Knowledge Discovery in Databases* (Descoberta de Conhecimento em Base de Dados

LIBRAS Língua Brasileira de Sinais

MLP *Multilayer Perceptron* (Perceptron Multicamadas)

NL Não Linearidade

NBN *No Batch Normalization* (Sem Normalização de Lote)

OpenCV *Open Computer Vision* (Visão Computacional Aberta)

RAM *Random Acess Memory* (Memória de Acesso Aleatório)

ReLU *Rectified Linear Unit* (Unidade Linear Retificada)

RNA Rede Neural Artificial

SE *Squeeze-and-Excite*

SiLU *Sigmoid-weighted Linear Unit* (Unidade Linear Sigmoide Ponderada)

WEKA *Waikato Environment for Knowledge Analysis* (Ambiente Waikato para Análise de Conhecimento)

.

SUMÁRIO

1 INTRODUÇÃO

Em sua história, o homem passou por diversas transformações, por exemplo em sua estrutura física, comunicação, organização social, política e econômica. Enquanto um ser biopsicossocial (ENGEL, 1977), o ser humano é formado por aspectos biológicos, psíquicos e sociais. Dessa forma, evidencia-se o desenvolvimento de diversas habilidades no processo de adaptabilidade às condições do contexto ao qual está inserido (VYGOTSKY, 1987). Logo, há o surgimento da comunicação e a construção da linguagem humana, como elementos intrinsicamente relacionados aos aspectos sociais (LABOV, 2008).

A comunicação humana é efetivada pela construção da linguagem (PRETI, 1998). Ao longo da história, os seres humanos, organizados em sociedade, desenvolveram o signo e a significação. O signo é a forma de relacionar um som ou gesto a um dado objeto ou a uma determinada ação. Já a significação constitui-se como o uso social do signo, ou seja, quando se estabelece em toda sociedade a utilização daquele signo para um objeto ou ação. Sendo assim, a base da comunicação humana é a atribuição de significados a determinados signos, sendo essa a característica intrínseca da linguagem (BORDENAVE, 1997).

Conforme Chomsky (1998), a comunicação só existe inserida em uma sociedade, sendo possível pela combinação de signos normatizados para serem utilizados em diferentes contextos. Assim houve a criação da linguagem, a qual remete à combinação de signos e regras para a transmissão de informação. E, dessa maneira, os seres humanos registram sua história, transmitem sua cultura e seus valores ao longo do tempo.

É importante destacar a diferença entre língua e linguagem. Saussure (2008) reforça que a língua é um produto social da linguagem e um conjunto de convenções necessárias, adotadas pelo corpo social o que permite seu exercício pelos indivíduos. Já a linguagem é constituída por códigos que envolvem significação, não precisando necessariamente abranger uma língua (GOLDFELD, 1997).

É fato que existem diversos tipos de línguas, as quais utilizam desde recursos sonoros a símbolos e gestos para representar a comunicação. Nesse contexto, a língua de sinais também é considerada uma forma de comunicação entre indivíduos num contexto social. Ela surgiu com a necessidade natural do ser humano de usar um sistema linguístico para exprimir ideias, sentimentos e ações (QUADROS, 1997).

Nesse sentido, a língua de sinais é uma forma dos surdos participarem da sociedade, comunicando-se por meio da expressão de suas ideias, opiniões e visões de mundo, já que apresentam sérias dificuldades com a língua oral (DIZEU, CAPORALI, 2005). A língua de

sinais é muito importante na vida de uma pessoa surda, pois por meio dela, é possível adquirir a linguagem, conhecimento de mundo e de si mesma (HARRISON, 2000).

A principal característica da língua de sinais está no campo gesto-visual. Assim como as línguas oral-auditivas, as línguas de sinais também possuem características próprias que se constituem a partir de cinco parâmetros que são configuração de mão, movimento, direcionalidade, ponto de articulação e expressões faciais (FELIPE, 2006). Como defendem Dizeu e Caporali (2005), a língua de sinais deve ser respeitada como língua, pois além de estabelecer a comunicação, ela também possui regras que a constitui. Conforme Sacks (1998), elas apresentam sintaxe, gramática e semântica completas, mas obviamente, apresenta um caráter diferente da língua oral.

A língua de sinais se desenvolveu de diferentes formas no mundo, como a língua oral. No Brasil, a Língua Brasileira de Sinais (LIBRAS) reune características próprias da população brasileira. É considerada uma língua de modalidade gestual-visual porque utiliza, como meio de comunicação, movimentos gestuais e expressões faciais que são percebidos pela visão. A formação dos sinais é realizada a partir da combinação do movimento das mãos com um determinado formato incluindo o lugar que pode ser uma parte do corpo ou em um espaço em frente ao corpo. Os parâmetros de LIBRAS são constituídos pelas articulações das mãos podendo ser comparadas aos fonemas (sons) e aos morfemas (pequenas partes que formam as palavras) (RAMOS, 2004).

O alfabeto em LIBRAS é constituído em sua imensa maioria por gestos fixos, exceto a letra “z” que é constituída por movimento. Essa característica permite a utilização de fotografias para o ensino de crianças, por exemplo. Além disso, o aprendizado do alfabeto em LIBRAS é importante para a sua popularização, visto que pessoas que não são surdas-mudas podem aprender essa linguagem. Sendo assim, é possível utilizar aprendizado de máquina nesse processo e realizar classificações automáticas a partir de imagens do alfabeto em LIBRAS. Esse processo pode ser utilizado inclusive em sistemas de traduções automáticas dos sinais a fim de facilitar a comunicação, como realizado no trabalho de Rossit et al. (2022).

O aprendizado de máquina para fins de classificação de imagens obteve avanços significativos nas últimas décadas (ZHANG et al., 2021). Inserem-se nesse contexto as redes neurais artificiais (RNAs) convolucionais profundas (LECUN et al.,1998) com o uso de processamento em placas gráficas, as GPUs (*Graphics Processing Units*). Esse tipo de algoritmo é otimizado para trabalhar com imagens, pois processa linhas, curvas e bordas e em diversas camadas ocultas transforma a imagem em estruturas complexas em busca de

identificar regiões características que proporcionam classes de observações em um sistema de aprendizado supervisionado (PRIDDY, KELLER, 2005).

Diversas arquiteturas de redes neurais convolucionais foram desenvolvidas, obtendo altas acurácias para a classificação de imagens. Essa jornada iniciou com o trabalho pioneiro de Lecun et al. (1998), os quais desenvolveram a rede neural *LeNet-5* para o reconhecimento de dígitos. Já nos trabalhos de He et al. (2015), Howard et al. (2017) e Howard et al. (2019) o número de camadas ocultas aumentou significativamente, da mesma forma, houve um aumento de sua complexidade computacional, o que gerou a necessidade de processamento em GPU. Aliado a isso, a acurácia teve um incremento substancial, alcançando acurácias superiores a 0,90. Por consequência, trabalhos utilizando RNAs convolucionais são encorajados devido as suas elevadas precisão e acurácia.

A comunicação por gestos não é um sistema utilizado exclusivamente por humanos, além disso, a língua de sinais é tão antiga quanto o ser-humano. No entanto, mesmo em tempos pré-históricos, ainda que uma cultura humana tivesse os recursos materiais, os padrões familiares e atitudes perante a vida que possibilitasse que a criança surda sobrevivesse, se formaria naquele ambiente um sistema comunicativo derivado das partes visíveis da paralinguística e muito mais do sistema comportamental daquela cultura (STOKOE JR., 2005).

2 JUSTIFICATIVA

Nos últimos anos, a classificação de imagens utilizando algoritmos de redes neurais artificiais apresentou uma grande evolução em termos de acurácia e precisão, como com o uso de novas arquiteturas de *hardware*, as quais permitiram essa evolução. Ademais, a classificação de sinais de LIBRAS contribui com o estado da arte das aplicações automáticas da língua de sinais, permitindo assim, a adoção de facilidades na conversação entre pessoas surdas e não surdas. Portanto, o desenvolvimento de processamento automático de conversação permite uma maior interação entre as pessoas.

A tarefa de classificação de imagens contribui com a automatização da leitura de sinais em LIBRAS. Essa é a primeira etapa, a qual engloba a classificação dos sinais do alfabeto. Nesse ponto, pesquisas que objetivam testar algoritmos para classificar os sinais são importantes, pois essa é uma etapa primordial para o desenvolvimento de sistemas automáticos de conversação.

O desenvolvimento e uso de algoritmos de RNAs convolucionais é muito recente. Seus resultados são promissores para o universo da classificação de imagens. Estudos nesse sentido, explorando a capacidade desses algoritmos em diversas situações, são relevantes por permitirem avaliar os limites e desafios para o aprimoramento em termos de processamento e arquiteturas utilizadas. É nesse contexto que se insere o presente trabalho.

3 OBJETIVOS

O objetivo geral deste trabalho é avaliar uma arquitetura de RNA convolucional na tarefa de classificação de sinais do alfabeto em LIBRAS, contribuindo com os estudos de automatização do processamento e leitura no contexto da língua de sinais.

A partir deste objetivo geral, foram abordados os seguintes objetivos específicos:

- Apresentar as arquiteturas de RNAs convolucionais.
- Gerar a base de dados com as bibliotecas *MediaPipe* e *CVZone*.
- Implementar as camadas da arquitetura *MobileNetV3-Large.*
- Implementar uma classificação de fotografias de sinais do alfabeto em LIBRAS.
- Contribuir com os desafios da classificação de imagens por meio de RNAs convolucionais.
- Contribuir com o desenvolvimento de sistemas automáticos de tradução de LIBRAS a partir da classificação dos sinais de seu alfabeto.

4 FUNDAMENTAÇÃO TEÓRICA

4.1 Mineração de Dados

A mineração de dados consiste em extrair informação sobre uma base de dados, gerando conhecimento a ser utilizado para a tomada de decisão. Sua maior aplicação dentro das organizações, ocorreu a partir da década de 1980, a partir da necessidade que as empresas apresentavam em transformar dados armazenados em informações relevantes (TAN et al., 2009). Atualmente, diariamente há uma grande geração de dados, os quais precisam ser armazenados, tratados e processados para serem transformados em informações e conhecimentos relevantes de acordo com a finalidade desejada.

Segundo Tan et al. (2009), alguns autores consideram que os processos de mineração de dados e descoberta de conhecimento em banco de dados (*Knowledge Discovery in Databases* - KDD) são diferentes. A definição mais aceita sobre mineração de dados é a que preconiza que a extração de conhecimento de bases de dados seja o processo de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis embutidos nos dados.

Por sua vez, o KDD está preocupado com o desenvolvimento de métodos e técnicas para dar sentido aos dados. O problema básico abordado pelo KDD é o de mapear dados de baixo nível, normalmente muito volumosos, em outras formas que podem ser mais compactas (por exemplo, um breve relatório), mais abstratas (uma descrição ou modelo do processo que gerou os dados) ou mais útil (por exemplo, um modelo preditivo para estimar o valor de casos futuros). Em todos esses processos está a aplicação de métodos específicos de mineração de dados para descoberta e extração de padrões (FAYYAD et. al. 1996).

A Mineração de dados é utilizada em diversas áreas. Uma delas é a área da saúde, conforme relata Fernandes et. al. (2019), destacando que aplicações com algoritmos de aprendizado de máquina são utilizados para análises preditivas em saúde pública e em saúde e segurança no trabalho, apresentando melhor desempenho quando comparada a análises tradicionais. Com isso, os autores concluem que os algoritmos podem ser utilizados para fazer análises exploratórias do conjunto de dados ou para desenvolver modelos de predição para estimar os riscos de adoecimento de trabalhadores baseando-se em históricos prévios de doenças. Outra área em que se aplica mineração de dados é na educação, especificamente para a previsão de evasão escolar. Colpo et al. (2020) realiza estudos no âmbito da previsão da evasão escolar com o objetivo de identificar aspectos contextuais e técnicos a partir dos dados de pesquisas do cenário brasileiro. A maior parte

das pesquisas concentram-se nas regiões nordeste, sul e sudeste, sendo a maioria para explorar a evasão em cursos de graduação. Conforme os autores, as pesquisas apresentam grande variação na quantidade de alunos pesquisados, explorando as vertentes acadêmicas, econômicas e sociais. Os dados foram processados e classificados por meio de árvores de decisão no ambiente WEKA.

4.2 Tipos de aprendizado de máquina

Existem diversos algoritmos de mineração de dados para fins de classificação de dados. Eles podem ser agrupados em três principais tipos: supervisionado, não supervisionado e por reforço.

O aprendizado de máquina supervisionado é o tipo de aprendizado em que as classes ou rótulos de cada dado já são conhecidos (BATISTA, 2003). É um tipo de classificação pautada no treinamento de classificadores a partir das classes previamente estabelecidas. O modelo é construído para realizar a classificação em outros conjuntos de dados com os mesmos atributos do treinamento, indicando a classe de cada dado. Nesse tipo de aprendizado a participação do operador é essencial, pois é ele quem realiza o processo de obtenção de amostras para cada classe, criando o conjunto de treinamento do algoritmo (CERQUEIRA, 2010). O objetivo desse tipo de classificação é construir um modelo capaz de determinar a classe de novos dados não classificados (MONARD, BARANAUSKAS 2003). Assim sendo, inserem-se nesse contexto a classificação e a regressão de dados, sendo dois tipos de problemas de aprendizado supervisionado com saídas categóricas e numéricas, respectivamente.

A aprendizagem não supervisionada consiste no aprendizado onde o conjunto de dados dispõe apenas dos atributos de entrada, ou seja, não há nenhuma informação de sua classe ou rótulo (CERQUEIRA, 2010). O objetivo é construir um modelo que busque por conjuntos de dados com características similares, formando agrupamentos ou *clusters* (BATISTA, 2003). Portanto, o aprendizado não supervisionado depende dos atributos para a realização de agrupamentos. Nesse sentido, os algoritmos buscam dividir o espaço amostral em grupos iniciais e, a partir deles, encontrar similaridades entre os dados para classificá-los nos grupos pré-determinados. De forma iterativa todos os dados são testados e incorporados a cada um dos grupos. Desse modo, é possível agrupar todos os dados e, a partir da análise do operador, os grupos podem se tornar classes dentro do escopo da aplicação da qual os dados originais fazem parte.

No aprendizado por reforço um agente é conectado ao seu ambiente via percepção e ação. Esse tipo de aprendizagem caracteriza-se por diversos momentos de aprendizado,

definidas como experiências, que se estabelecem por tentativa e erro (MAHMUD et al., 2018). A cada iteração o agente recebe alguma informação sobre o estado atual do ambiente, escolhendo uma ação para gerar uma saída. A ação altera o estado do ambiente e esse valor é comunicado ao agente por meio de um sinal de reforço. O agente deve escolher ações que aumentem o valor do sinal de reforço, ele aprende a fazer isso ao longo do tempo através de tentativa e erro (KAELBLING et. al. 1996), sendo um processo de aprendizagem contínuo, onde suas interações com o ambiente ocorrem em intervalos de tempo discretos (SUTTON, BARTO, 1998). Por essas características, o aprendizado por reforço também é conhecido como um modelo de aprendizado semi-supervisionado no contexto do aprendizado de máquina (KONYUSHKOVA et al., 2020).

Nos últimos anos, o aprendizado por reforço vem sendo aplicado em diversas tarefas, sobretudo em aplicações cotidianas (LI, 2019). Verifica-se a aplicação em controle semafórico, com o intuito de controlar o tempo de fechamento e abertura, conforme o fluxo de veículos e o sincronismo entre os semáforos (WEI et al., 2019; BAZZAN, 2021). As sugestões de melhores rotas no trânsito disponíveis em sistemas de navegação de automóveis e aparelhos celulares, também podem ser obtidos pela aprendizagem por reforço, como em Sharon et al. (2017), Bazzan (2021) e até mesmo aplicado aos serviços de entrega, como abordado por Liu et al. (2020).

Diversos algoritmos de classificação inserem-se no contexto de cada abordagem apresentada. Para classificação de imagens existem diversos algoritmos. Os clássicos algoritmos de árvores de decisão, como o C4.5 (QUINLAN,1986) e Random Forest (BREIMAN, 2001), são representantes da classificação supervisionada, juntamente com as diversas versões de algoritmos de Redes Neurais Artificiais. Além disso algoritmos de regressão, como Árvores de Regressão (QUINLAN, 1992) e Aprendizado Baseado em Instâncias (WEISS et al., 1991), também partilham das características do aprendizado supervisionado. Por sua vez, os algoritmos K-médias (HARTIGAN, 1979) e Redes de Kohonen (KOHONEN,1982), destacam-se como representantes do tipo de classificação não supervisionada. Por fim, algoritmos como o *Q-learning* (WATKINS, 1992) utilizam em sua concepção aspectos da abordagem de aprendizado por reforço.

A seguir, serão apresentadas as características do algoritmo supervisionado utilizado neste trabalho.

4.3 Redes neurais artificiais

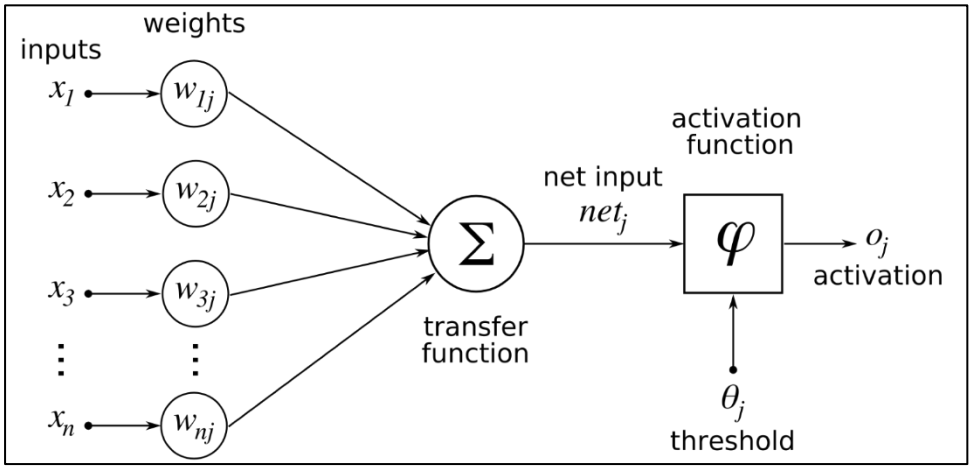
Basicamente, uma rede neural artificial (RNA) consiste em um modelo matemático, o qual busca simular o comportamento de processamento e treinamento de uma rede neural

biológica (PRIDDY, KELLER, 2005). O bloco de processamento é composto pelo neurônio artificial, o qual pode ser considerado uma função matemática que é capaz de realizar regras de processamento simples, sendo elas: multiplicação, soma e ativação a partir de um dado modelo. A entrada da rede é composta por neurônios que representam o espaço dos atributos, onde seus valores são ponderados a partir da multiplicação de um peso. Nas seções intermediárias, que são chamadas de camadas ocultas, os neurônios recebem todas as entradas ponderadas pelos pesos e executa a função de soma. Assim, cada neurônio distribuído em n camadas ocultas realizam a soma das entradas considerando os pesos aplicados. Na camada de saída, a soma de todas as entradas ponderadas anteriormente passa por uma função de ativação para gerar a saída do processamento, a qual corresponde às classes esperadas na aplicação (KOVÁCZ, 2002).

O desenvolvimento das RNAs remonta ao início do século XX, onde pesquisadores desenvolveram modelos matemáticos complexos para demonstrar sua viabilidade. Inicialmente, Mcculloch e Pitts (1943) criaram um modelo baseado em lógica de cálculo com valores binários para simular uma rede neural pela combinação de vários neurônios, o conceito tem como função determinar a ativação de um neurônio com base na soma das entradas recebidas. Essa foi a base para o desenvolvimento do modelo *perceptron*, o qual consiste em uma camada única de apenas um neurônio artificial com a soma dos pesos das entradas e aplicado o limite de ativação para sinal de saída binária (ROSENBLATT, 1958).

A Figura 1 apresenta a estrutura do modelo *perceptron*, composto por uma camada de entrada onde cada neurônio recebe um parâmetro ou atributo que será multiplicado por um peso sináptico. Os resultados do processamento de cada neurônio da camada de entrada serão enviados a um neurônio de uma camada intermediária que recebe o nome função de transferência. Essa função junta todos os resultados e o transfere para uma função de ativação, a qual constitui a camada de saída. A função de ativação tem por objetivo utilizar um limiar para classificar os resultados, conforme as classes de saída do modelo.

Figura 1 - Modelo *perceptron* de uma única camada.



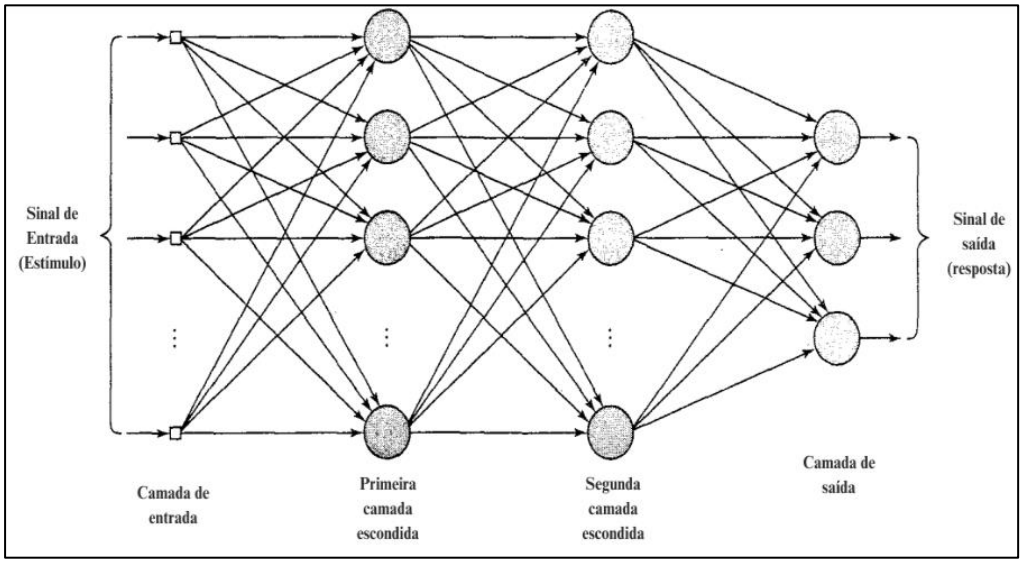
Fonte: (Chrislb (CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=224555) [cited 25 jun. 2023]).

Anos mais tarde, Rumelhart et al. (1986) introduziram o algoritmo *backpropagation*, sendo considerado uma grande revolução no desenvolvimento das redes neurais. Esse algoritmo permite à rede neural aprender a partir de dados classificados, melhorando os pesos durante o treinamento. O algoritmo envia o erro de volta pela rede, da camada de saída passando pelas camadas ocultas até a camada de entrada. Dessa forma, são utilizados gradientes do erro para ajustar os pesos e viés da rede neural, modificando os parâmetros da rede durante o processo de treinamento.

De acordo com Haykin (1994), as redes *perceptron* de uma camada oculta evoluíram para uma estrutura que permite a construção de várias camadas ocultas (escondidas), dando origem às redes *multilayer perceptron* (MLP). Esse tipo de RNA é descrito como um grupo de unidades sensoriais com uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Haykin (1994) destaca que o modelo MLP tem várias características que a destacaram, sendo que em cada um dos neurônios ocorre uma ativação não-linear, além de que a presença de várias camadas ocultas permite a rede extrair progressivamente *features* dos padrões de entrada e exibe alto graus de conectividade determinada pelas sinapses da rede.

É possível verificar por meio da Figura 2 a construção de uma MLP. A novidade desse tipo de rede neural é a adição de mais camadas ocultas. No exemplo, tem-se duas camadas ocultas, as quais recebem as saídas das camadas anteriores e realizam cálculos em cada neurônio e os resultados são enviados para a camada seguinte. Essa estrutura permite a utilização de várias camadas ocultas, a depender da complexidade do processamento a ser realizado.

Figura 2 - Rede *multilayer perceptron* de duas camadas ocultas.



Fonte: (HAYKIN, 1994).

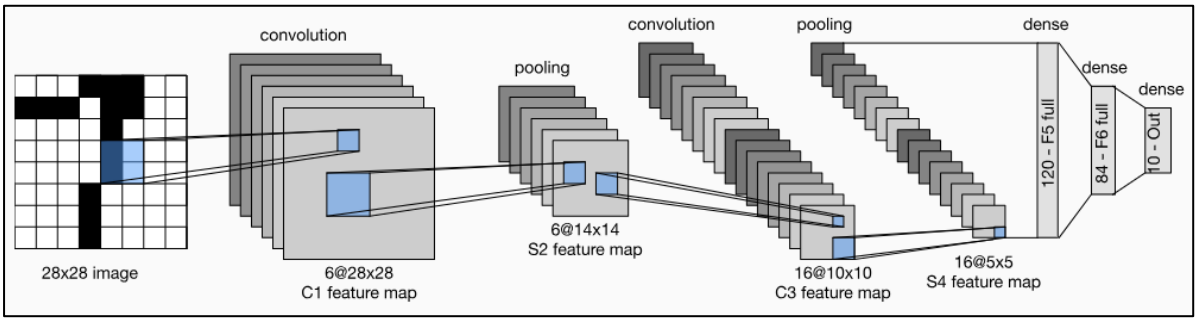
4.3.1 Redes neurais convolucionais

Em seu trabalho Lecun et al. (1998) introduziu as Redes Neurais Convolucionais em uma análise automatizada para reconhecer palavras em documentos manuscritos. Esse processo, chamado de Reconhecimento Óptico de Caracteres, objetiva converter imagens de texto para a classificação de documentos. Nesse trabalho, os autores inserem o conceito de Rede Neural Convolucional, sendo designadas para processamento de dados estruturados em forma de grade, como em imagens e efetivas para entender padrões e hierarquias de características, automatizando a extração de características responsáveis para identificar elementos relevantes que podem descrever os dados.

A Figura 3 apresenta o modelo da Rede Neural Convolucional *LeNet-5* (LECUN et al., 1998) usada para reconhecimento de dígitos. A arquitetura contém sete camadas, sendo três delas convolucionais, duas camadas *pooling* para realizar a redução na quantidade de parâmetros para manter as características mais importantes e duas totalmente conectadas (*fully connection*), cuja função é conectar todos os neurônios da camada seguinte para reduzir a dimensionalidade dos dados, transformando-o de tridimensionais para bidimensionais. Logo, a rede é composta por uma camada de entrada (*input*), sendo a imagem dos caracteres com tamanho 32x32. As camadas C1, C3 e C5 são as convolucionais, as quais utilizam uma operação matemática chamada de convolucão na qual é aplicada um filtro que percorre o dado de entrada para extrair suas principais características. Entre as camadas C1 e C3 têm as camadas *subsampling* ou camadas

pooling, representadas como S2 e S4. As camadas totalmente conectadas são identificadas por F5 e F6. Por fim a camada de saída (Out).

Figura 3 - Arquitetura da LeNet-5.

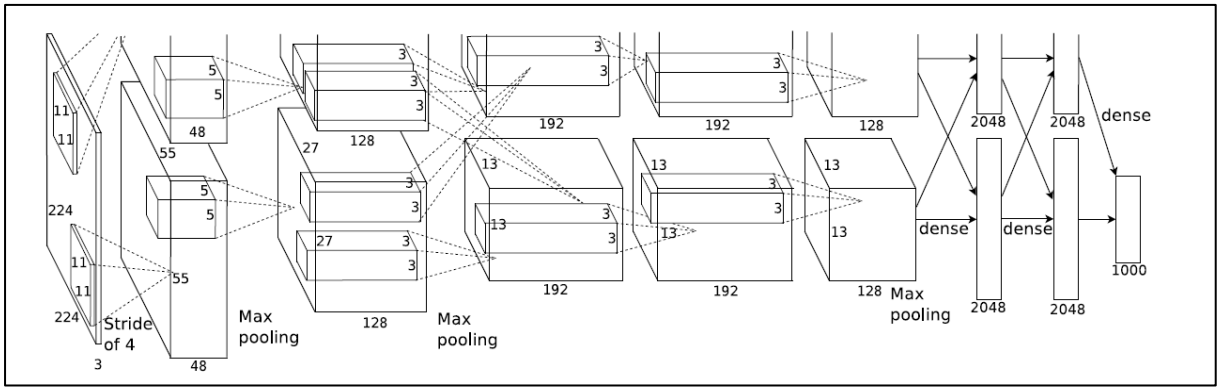


Fonte: (ZHANG et al., 2021).

Outro exemplo de rede convolucional é a *AlexNet*, desenvolvida por Krizhevsky et al. (2012). Considerada uma rede convolucional profunda, tendo como capacidade aprender representações hierárquicas de características mais complexas e a aprendizagem para classificação de imagens em grande escala. A *AlexNet* tem como principal recurso o uso de GPU (*Graphics Processing Unit*), em português Unidade de Processamento Gráfico. O modelo foi treinado usando duas GPUs, dividindo tarefas durante o treinamento e usando paralelismo computacional, fazendo as GPUs lerem e escreverem na memória uma das outras. A arquitetura consiste em oito camadas, sendo que as cinco primeiras são convolucionais e as três restantes são totalmente conectadas e a camada de saída com total de 1000 classes.

Na Figura 4 temos a arquitetura *AlexNet*, sendo possível verificar como as GPUs trabalham na rede neural. Uma GPU trabalha na parte superior das camadas, como visto nas linhas pontilhadas na parte superior da figura. Enquanto a outra GPU, com a parte inferior, como visto nas linhas pontilhadas na parte inferior da figura.

Figura 4 - Arquitetura da *AlexNet*,



Fonte: (KRIZHEVSKY et al., 2012).

Resnet é outro exemplo de rede neural convolucional. Criada por He et al. (2016), ela dispõe de uma arquitetura que introduziu o conceito de conexões residuais, o qual tem como função aprender a mapear a diferença entre a entrada e a saída esperada. Isso permite que a rede ‘pule’ para as camadas mais profundas, enquanto as redes neurais tradicionais com grande quantidade de camadas possuem uma transmissão das informações de maneira linear, passando camada por camada podendo ocorrer perda de informações. Dessa maneira, as redes profundas que contêm grande quantidade de camadas com conexões residuais podem obter melhor desempenho em comparação com as redes neurais que não possuem conexão residual.

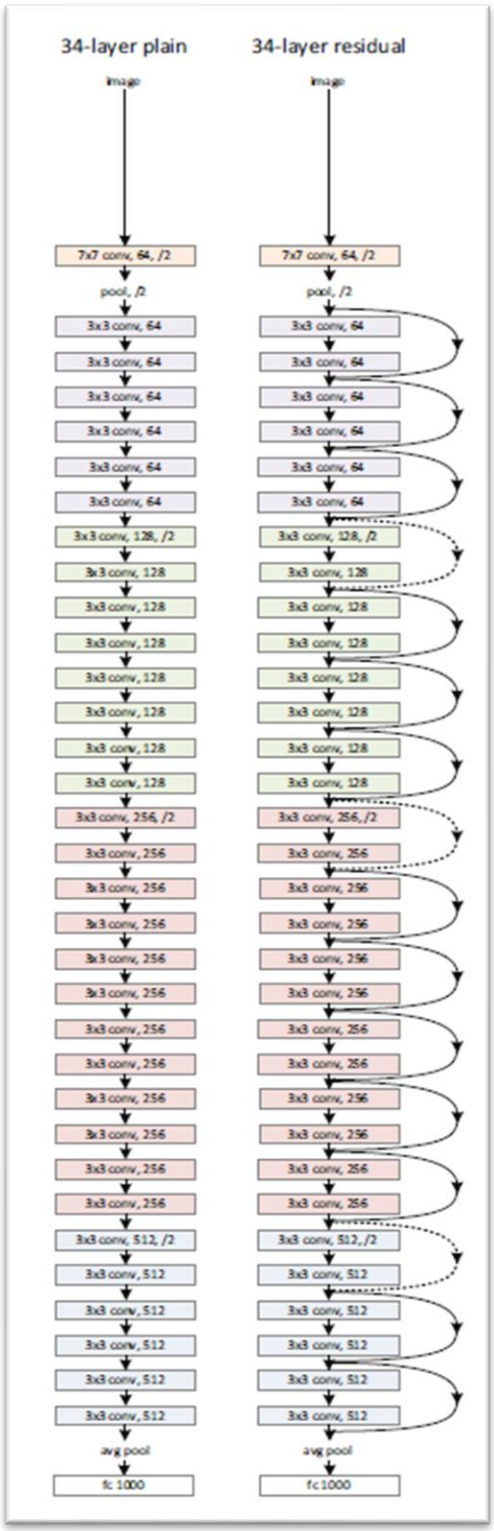
Na Figura 5 temos uma comparação entre as arquiteturas de uma rede neural convolucional, *Resnet-34* sem conexões residuais (34-*layer plain*) e uma arquitetura da *ResNet-34* com conexões residuais (34-*layer residual*). Na figura os retângulos são as camadas convolucionais, as setas mostram as etapas que os dados iram passar, os retângulos com cores diferentes representam etapas na rede, os retângulos bege representam camada de entrada onde temos filtro de 7x7 que seria uma matriz que desliza sobre a imagem e 64 filtros convolucionais que representam todas as matrizes, e o retângulo branco a camada de saída onde tem 1000 filtros convolucionais representando todas as classes.

Após a camada de entrada tem uma camada *pooling* responsável em reduzir a dimensionalidade e na penúltima camada antes da saída, retângulos roxos representam sequência de camadas convolucionais com filtro 3x3, que é aplicado com a função de multiplicar os valores do filtro pelo subconjunto, correspondendo a entrada e somando esses valores para produzir um valor de saída e com 64 filtros convolucionais que tem função de aprender a detectar um conjunto diferente de características nas imagens. Verde são camadas convolucionais com filtro 3x3 com 128 filtros convolucionais, retângulo rosa é filtro 3x3 com 256 filtros convolucionais e retângulo azul é camadas convolucionais com 3x3 filtros com 512 filtros convolucionais.

Em certas camadas como a camada de entrada, nas camadas *pooling* após a de entrada e nas camadas onde ocorre aumento na dimensão tem digito “/2”, que significa *stride* ou em português deslocamento de 2, que nessas camadas o filtro é deslocado 2 *pixels* em cada etapa da convolução resultando em uma redução na resolução espacial da saída em comparação com a entrada. Na figura 5, na coluna 34-*layer residual*(direita), podemos ver setas contornando certas camadas, essas setas são as conexões residuais. Essas conexões podem ser facilmente usadas quando as camadas têm a mesma dimensão, porém quando a dimensão aumenta representada com setas com linhas sublinhadas, os autores

usaram uma técnica chamada de atalho de projeção, onde se aplica uma convolução com filtro 1x1 nos atalhos, dessa maneira eles podem ajustar o número de dimensões para se ajustar com o novo tamanho das camadas (HE et al., 2016).

Figura 5 - Arquitetura da *Resnet-34*.



Fonte: modificada de He et al., (2016).

A arquitetura *MobileNet* foi apresentada por Howard et al. (2017), a qual é baseada em convoluções separáveis em profundidade construindo uma profunda rede neural convolucional. Suas características fazem o modelo ser leve, consequentemente reduzindo seu tempo computacional, exceto a primeira camada que utiliza recursos tradicionais de uma camada convolucional completa.

A Figura 6 apresenta as características da *MobileNet*. A coluna *type/Stride* representa o tipo da camada e o deslocamento, podendo ser: Conv - a camada convolucional; Conv dw - camada convolucional separada em profundidade; s1 e s2 representam o deslocamento do filtro. As três últimas camadas na imagem representam a camada *pooling* para redução da dimensionalidade. A camada FC representa a camada totalmente conectada para conectar todos os neurônios na camada seguinte. Já a *Softmax* representa a camada de saída. A coluna Filter Shape refere-se aos filtros usados nas camadas convolucionais.

Nas linhas estão representados os filtros que possuem três ou quatro dimensões. As camadas com três dimensões são representadas pela altura, largura e canais de entrada do filtro. Já as camadas compostas por 4 dimensões possuem a altura, a largura, o número de canais de entrada e o número de canais de saída do filtro. Por fim, a última coluna representa o tamanho da imagem de entrada para cada camada. As imagens são representadas como entidades tridimensionais, as quais possuem a largura, altura e número de canais de cores. Verifica-se que ao longo do treinamento o tamanho e dimensões das imagens vão progressivamente diminuindo (HOWARD et al. 2017).

Figura 6 - Arquitetura da *MobileNetV1*.

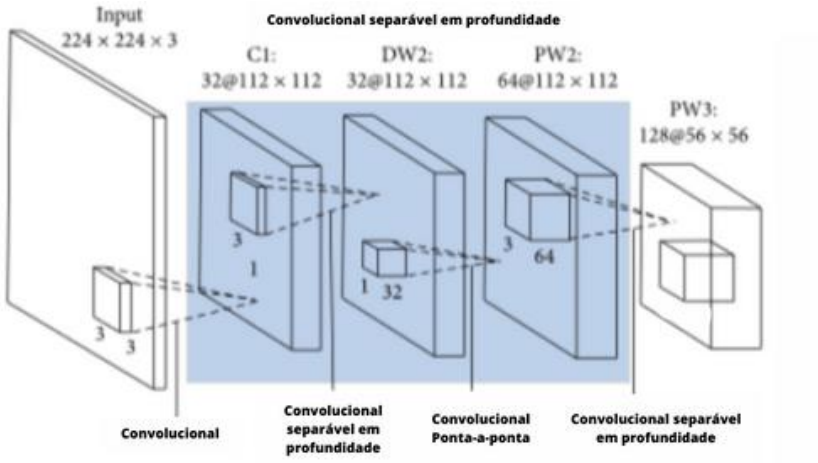
Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Fonte: (HOWARD et al., 2017).

Em seus trabalhos Sifre e Mallat (2014) e Howard et al. (2017) aplicaram a técnica de convoluções separáveis em profundidade. Esse recurso corresponde à fatorização de uma camada convolucional padrão, ou seja, separar as camadas em profundidade e em camada ponta-a-ponta, a qual corresponde a uma convolucão com filtro 1x1. A *MobileNet* utiliza uma camada convolucional separável em profundidade com filtro e na sequência uma convolucional ponta-a-ponta aplicando um filtro 1x1.

Na Figura 7 temos outra visão da arquitetura na *MobileNetV1*, mostrando como funciona a camada convolucional separável em profundidade. No input é representada a imagem em seu tamanho original: 224 x 224 x 3. Na figura é possível ver o filtro convolucional 3 x 3 que percorrerá toda a imagem e o resultado das operações de cada posição do filtro resultará na imagem da próxima camada. Na sequência, observa-se uma seção de três camada em azul, a qual ilustra a técnica convolucional separada em profundidade. Em C1 é representada a camada de entrada com tamanho de 112 x 112 x 32. Nessa camada foi utilizado um filtro 3 x 3 com 1 dimensão. O resultado do processamento resultou na camada DW2 com o mesmo tamanho da antecessora. Nessa camada foi aplicada a convolução ponta-a-ponta com um filtro 1 x 1 com 32 dimensões, resultando na camada PW2. Essa camada corresponde a outra convolução separável em profundida, agora com um filtro 3 x 3 com 64 dimensões (SIFRE, MALLAT, 2014; HOWARD et al., 2017; AJI et al., 2021).

Figura 7 - Convolução separável em profundidade da *MobileNetV1*.



Fonte: modificada de AJI et al., (2021).

Na versão *MobileNetV2*, introduzida por Sandler et al., (2018) foram adicionados dois tipos de blocos convolucionais: o *linear bottleneck* e o residual invertido. O primeiro bloco aplica uma transformação em um espaço dimensional menor, reduzindo custo computacional e número de parâmetros. Ele é estruturado em bloco de camadas que tem como primeira etapa reduzir o número de canais aplicando um filtro 1 x 1. Na segunda etapa aplica uma camada convolucional de profundidade com filtro 3 x 3 para extração das características. Por fim aplicar uma camada convolucional com filtro 1 x 1 para aumentar as dimensões para aquela antes de entrar na primeira etapa do bloco.

Já o bloco residual invertido, que é o inverso do linear *bottleneck*, por aplicar a redução das dimensões no final do bloco. Essa configuração é aplicada devido a redução da dimensão antes da camada convolucional em profundidade, que pode resultar em perda de informação. Assim, no bloco residual invertido é aplicada a redução da dimensionalidade depois, evitando a perda de informação. Ela funciona aplicando convolução com filtro 1 x 1 expandindo a dimensionalidade, a seguir aplica a convolução em profundidade e, por fim a outra convolução para reduzir a dimensionalidade (SANDLER et al., 2018).

MnasNet é uma rede neural Convolucional profunda desenvolvida por Tan et al. (2019) que foi adaptada da *MobileNetV2* de Sandler et al. (2018). Foi introduzido o mecanismo *Squeeze-and-Excite* que, segundo Hu et al. (2018) envolve duas etapas: a *squeeze* e *excite*. A *squeeze* (em português: espremer) refere-se a etapa de captura de informações globais, com a camada global *pooling*.

De acordo com Simonyan e Zisserman (2014) a *squeeze* é responsável pela realização de uma redução espacial em um vetor de uma dimensão. Já o *excite* (em

português: excitar), conforme os mesmos autores, é a etapa onde são recebidos os recursos após o *squeeze*, realizando uma recalibração. Para isso, utiliza o mecanismo de ativação para conectar com uma camada totalmente conectada e uma camada de saída. Dessa maneira, o mecanismo permite à rede neural aprender dados de entrada e recalibrar os recursos de maneira adequada, ajudando no desempenho. Na *MnasNet*, segundo Tan et al. (2019) o mecanismo *Squeeze-and-Excite* consiste em um bloco de camadas, seguindo uma estrutura global *pooling*, totalmente conectada, juntamente com uma função de ativação, totalmente conectada e uma camada de saída.

A *MobileNetV3-Large* é uma rede neural convolucional desenvolvida por Howard et al. (2019) consiste em uma atualização da *MobileNetV1* criada Howard et al. (2017) com a introdução de convoluções separáveis em profundidade. Uma evolução da *MobileNetV2* de Sandler et al. (2018), que introduziu a técnica linear *bottleneck* e com a introdução do mecanismo *Squeeze-and-Excite* desenvolvido por Hu et al. (2018) e utilizado na rede neural profunda da *MnasNet* de TAN et al. (2019).

A Figura 8 apresenta a arquitetura da rede *MobilenetV3-Large*. A coluna *input* denota o tamanho da imagem com largura, altura e dimensionamento. Na coluna *operator* são apresentados os tipos das camadas, sendo que a *conv2d* é uma camada convolucional tradicional, *bneck* é um bloco *bottleneck*, o qual é composto por camadas e operações como: *linear bottleneck*, *batch normalization*, função de ativação e convolução em profundidade. Segundo Ioffe e Szegedy (2015), *Batch normalization* normaliza as ativações de cada camada com base em minilotes (*batches*), isso é feito ao calcular valores de ativação em cada um dos lotes e na sequência normalizar essas ativações reduzindo-as.

Essas ativações são feitas em uma camada antes da função de ativação, assim tornando o modelo menos sensível a inicialização dos pesos e ajuda a reduzir o ajuste excessivo dos dados no treinamento. A coluna *operator* também mostra os filtros como 3x3 e 5x5. No final da arquitetura temos *pool* que é uma camada de redução de dimensionalidade e o NBN que significa *no batch normalization*, ou seja, não foi aplicado *batch normalization*. A coluna *exp size*, é o espaçamento entre as camadas, controlando assim os números de canais que são multiplicados após camadas com filtro 1 x 1 aumentando a captura de informação. Na coluna *#out* é a que se refere ao número de dimensões dos canais de saída dos blocos. *SE* é a coluna que representa se houve o mecanismo *Squeeze-and-Excite*. A coluna *NL* denota o tipo de não linearidade ou função de ativação utilizado *HS* (*h-swish*) ou *RE* (*ReLU*). *HS* corresponde à junção das funções de ativação *ReLU* (*Rectified Linear Unit*) e SiLU (*Sigmoid-weighted Linear unit*). A função de ativação *ReLU* foi introduzida por Glorot et al. (2011), caracterizada por ser uma função não

linear simples, amplamente utilizada devido a sua eficácia. Ela retorna o valor máximo entre zero e o valor de entrada, se o valor for positivo e se o valor for negativo, ele retorna zero. Segundo Hendrycks e Gimpel (2016) a função *SiLU* corresponde à multiplicação da entrada com a função sigmoide de Hinton et al. (2006), para mapear valores reais entre zero e um. Por fim, a coluna *S* refere-se ao *stride*.

Figura 8 - Especificação da *MobileNetV3-Large*.

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

Fonte: (HOWARD et al., 2019).

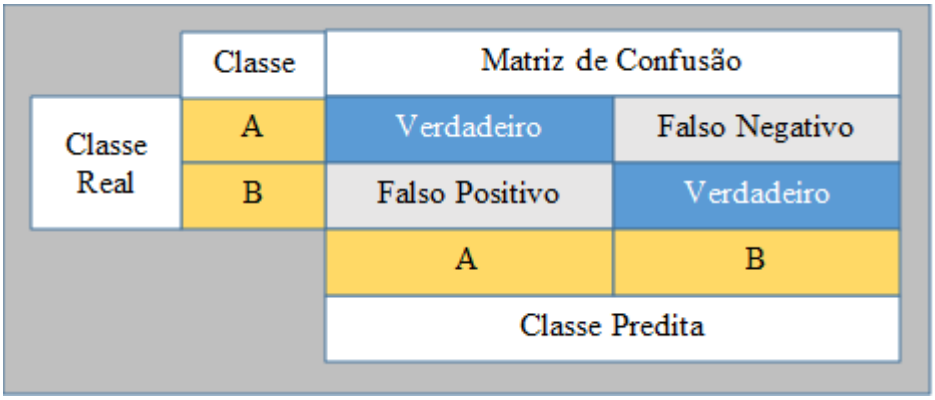
4.4 Métricas de validação

No processo de validação da predição foram utilizadas diferentes métricas, sendo elas: a matriz de confusão e precisão da classificação.

Segundo Sonka e Fitzpatrick (2000), na matriz de confusão os termos utilizados são verdadeiro positivo, falso positivo, verdadeiro negativo e falso negativo. Nesses termos o positivo e negativo referenciam as decisões feitas pelo algoritmo, enquanto verdadeiro e falso referem-se como o algoritmo concorda com a classe correta, o que pode ser verificado na Figura 9. Uma representação em tabela de uma matriz de confusão, onde na primeira coluna refere-se as classes reais sendo no exemplo a classe A e B em amarelo e na vertical na parte inferior da tabela novamente as classes em amarelo que referisse as classes

preditas. Em azul com a palavra verdadeiro temos as predições corretas feitas pelo modelo, e em cinza temos as predições erradas pelo modelo.

Figura 9 - Exemplo de uma matriz de confusão.



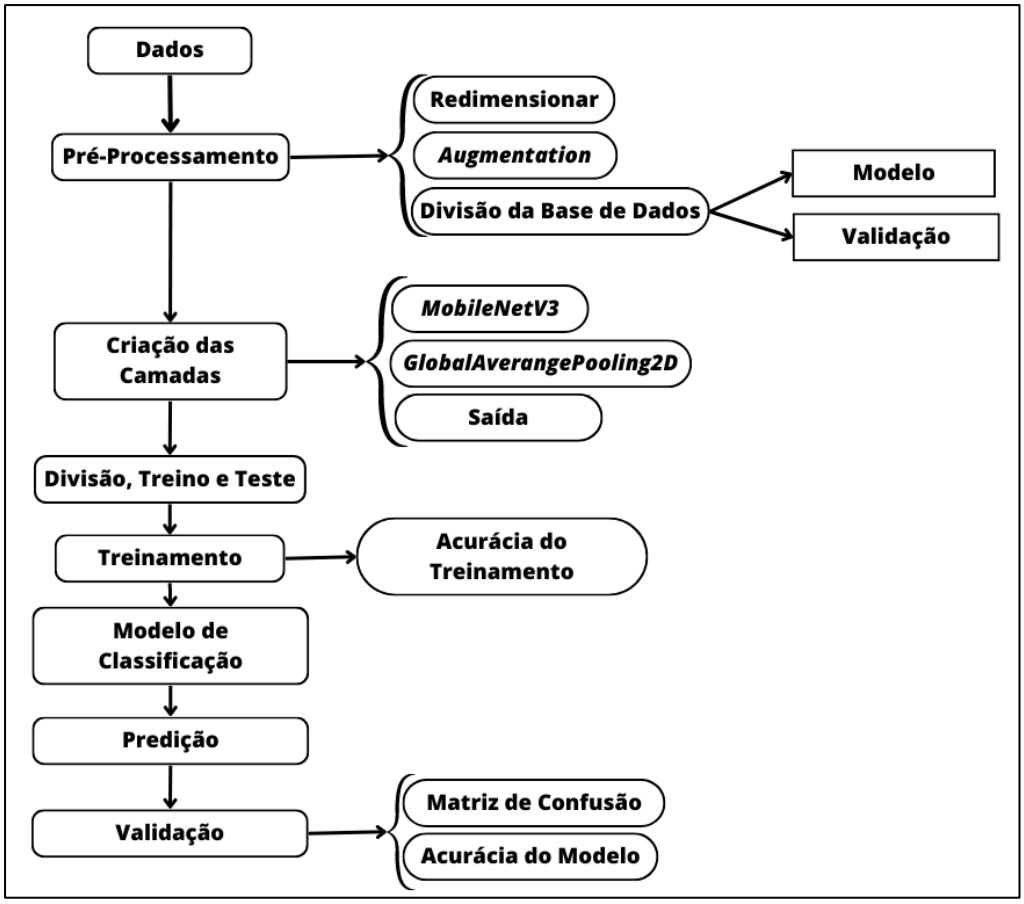
Fonte: Produção própria.

Conforme Witten e Frank (2000), o reporte de classificação representa as principais métricas de uma classificação com base em cada classe. Dessa forma, é possível representar fraquezas funcionais do modelo. A precisão é definida como a razão entre o verdadeiro positivo e a soma dos verdadeiros positivos e falso positivos. O recall é uma medida de completude do classificador para cada classe, sendo a razão entre verdadeiros positivos e a soma dos verdadeiros positivos e falsos negativos. Pontuação F1 é a média harmônica ponderada da precisão e recall, onde as pontuações vão de 0 a 1. Suporte é o número de ocorrências reais da classe no conjunto de dados, que quando há um número desequilibrado entre as classes pode significar problemas estruturais nas pontuações relatadas pelo classificador.

5 METODOLOGIA

Esta seção tem como objetivo apresentar os materiais e métodos utilizados neste trabalho de graduação. A Figura 10 apresenta o fluxograma metodológico desenvolvido. O trabalho inicia com a construção da base de dados, que será contemplado na Seção 5.1. O passo seguinte foi o pré-processamento, o qual contempla as transformações aplicadas às imagens que serão explicadas na Seção 5.2. Por fim, a Seção 5.3 aborda as implementações realizadas para a construção e utilização do modelo de classificação.

Figura 10 - Fluxograma metodológico do trabalho



Fonte: Produção própria.

5.1. Materiais

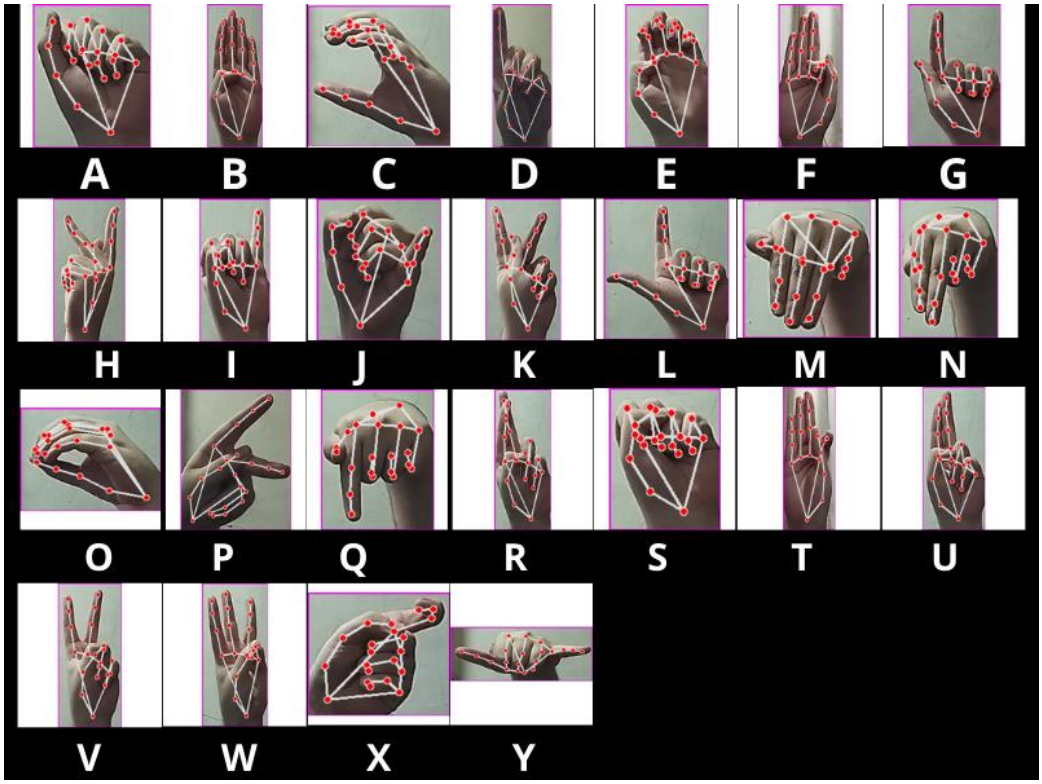
Para o processamento foi utilizado a plataforma computacional remota do *Kaggle notebooks* com as seguintes especificações técnicas: espaço de disco de até 20 GB, CPU (Unidade Central de processamento) com Intel(R) Xeon(R) CPU @ 2,20GHz, 13 GB de

memória RAM (Memoria de Acesso Aleatório) e 1 Nvidia Tesla P100 GPU de 15,9 GB de memória (KAGGLE, 2023).

5.2. Base de dados

A base de dados consiste em 15600 imagens divididas em 26 diretórios, 25 deles contêm um sinal referente a uma letra do alfabeto em libras, exceto Z, por esse ser constituído por movimento. O vigésimo sexto diretório refere-se a classe de sinais não identificado (ZDF). Cada imagem foi configurada em 300 x 300 *pixels* de altura e largura. A Figura 11 apresenta os sinais em LIBRAS referentes a cada uma das letras do português.

Figura 11 - Alfabeto de sinais em libras.



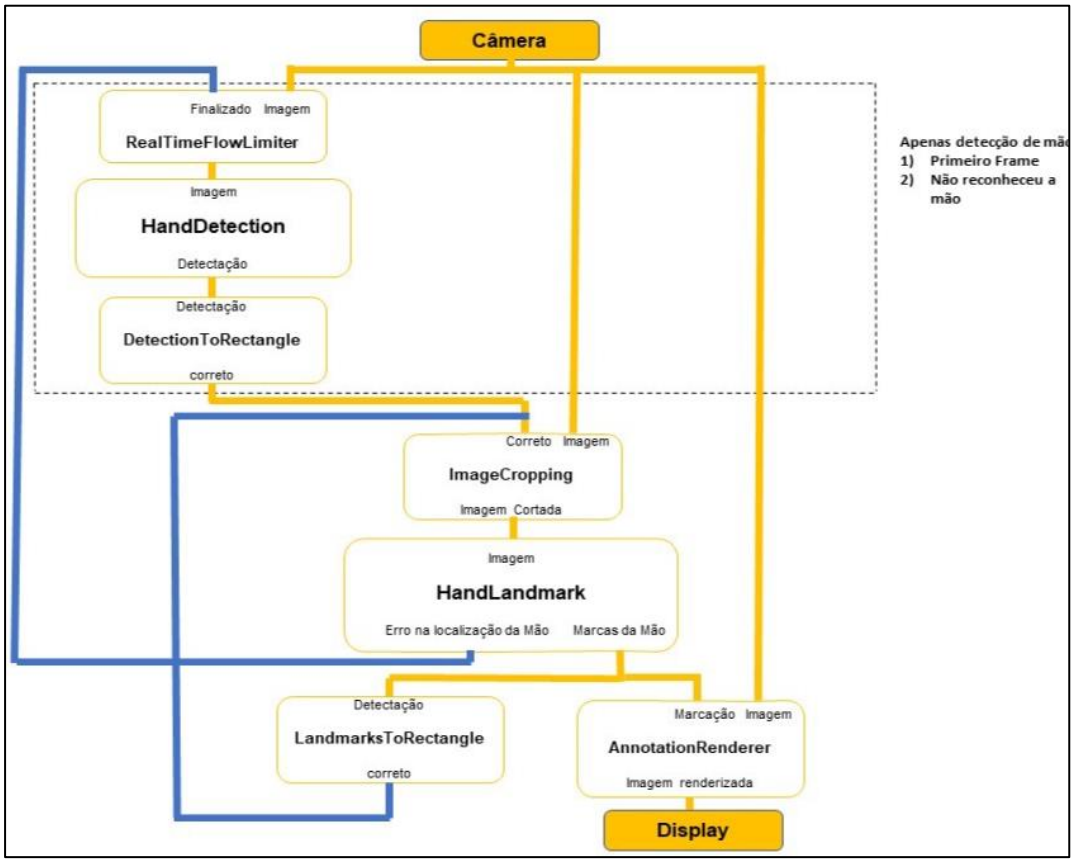
Fonte: Produção própria.

As imagens foram obtidas por meio das bibliotecas *MediaPipe* e *CVZone*. Para realizar a marcação de mão foi utilizada a biblioteca *MediaPipe*, disponível em Python. Segundo Lugaresi et al. (2019), *MediaPipe* é um *framework* designado para implementação de aplicações de aprendizado de máquina sobre dado sensorial. Conforme Zhang et al. (2020) a biblioteca utiliza aprendizado profundo, combinação de detecção de objetos e

regressão para estimar as coordenadas das marcações na mão, identificando a aplicação precisa das coordenadas em 2.5d das marcas da mão, mesmo com visibilidade parcial.

A Figura 12 apresenta as duas etapas de funcionamento do *MediaPipe*. A primeira delas é para detectar a mão, as demais para a aplicação das marcas. A detecção contínua da mão ocorre pela localização do frame atual no vídeo. Já o processo de computação das marcas de mão baseia-se no frame anterior, sendo que uma nova detecção ocorre novamente quando a confiança da detecção cair, devido a movimentação da mão. A etapa de corte da imagem ajuda no processo de marcação e reduz drasticamente a necessidade de usar aumento de dados (*augmentation*) de imagem, o que permite à rede neural se dedicar em aplicar as marcações.

Figura 12 - Pipeline do MediaPipe para detecção da mão e aplicação na marcação.



Fonte: modificado de Zhang et al., (2006).

O *CVZone* é um pacote de visão computacional em Python que oferece recursos avançados para o processamento de imagens e vídeos em tempo real, utilizando como core as bibliotecas *OpenCV* e *MediaPipe* (CVZONE, 2023). Esse pacote foi desenvolvido com o objetivo de fornecer um conjunto de ferramentas úteis para trabalhar com tarefas de visão computacional, como em Strand e Karlsson (2022) para detecção de objetos. Já Guglielmo

et al. (2022) trabalharam com reconhecimento facial e segmentação de imagens. Por sua vez, Khanorkar et al. (2022), aplicou o *CVZone* em classificação de imagens e estimativa de pose. Ambos obtiveram resultados promissores com o uso desse pacote computacional.

5.3. Pré-processamento

O primeiro procedimento realizado no pré-processamento foi o aumento de dados de imagem. Essa técnica é considerada para gerar novas imagens de treinamento a partir das imagens de origem. No aumento de dados podem ser utilizadas, dentre outros recursos, a rotação horizontal, a verticalização, inversão da imagem, aumento de sua largura e altura. São aplicadas no reconhecimento de imagens e apresenta melhora na acurácia. (YUNUS et. al. 2019)

O aumento de dados de imagem, foi utilizada para aplicar rotação em 30 graus. Pelo fato de os sinais em Libras serem específicos a cada palavra, a rotação não deve ser muito grande para não alterar seu significado. Foi utilizada alteração do brilho de forma aleatória entre 0,2 e 1,0, a fim de melhorar a interpretação do modelo em diferentes ambientes. Também foi aplicado zoom na imagem em 0,8 para impor maior precisão na classificação em diferentes distancias. Além disso, foram aplicadas: inversão da imagem horizontalmente; reescala dos pixels para 1/255; mudança do tamanho da imagem para 224 x 224.

Para a realização destes procedimentos foi utilizada a biblioteca *tensorflow*, disponível em Python. Segundo Developers (2022), *tensorflow* é uma biblioteca *open-source* de aprendizado de máquina utilizada para criar e treinar modelos de aprendizado de máquina úteis para área acadêmica e desenvolvedores, tornando-se uma ferramenta popular devido a sua vasta utilização, incluindo: reconhecimento de imagens, de voz, processamento em linguagem natural, análise preditiva, dentre outras aplicações.

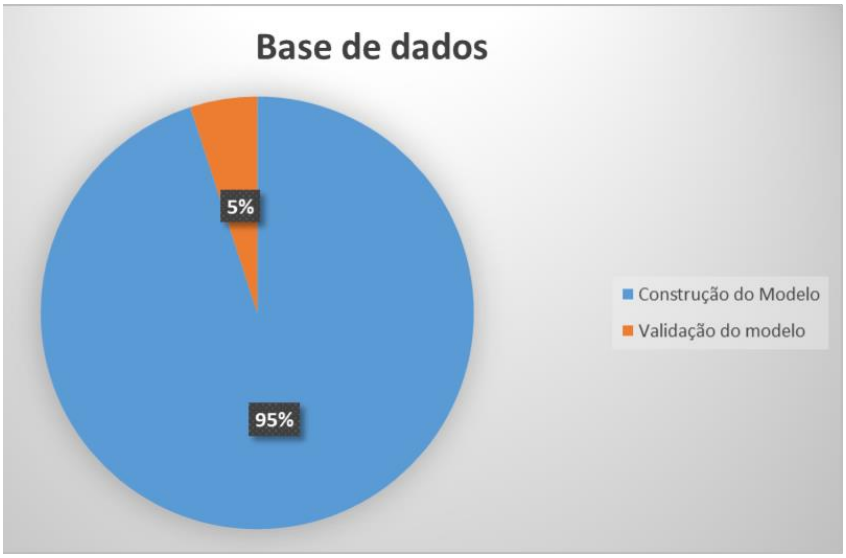
Ainda na etapa de pré-processamento, foram alterados os nomes das classes para valores numéricos sendo: 1,2,3 (...). Além disso, foi realizada uma primeira divisão dos dados, em dados de geração do modelo e dados de validação do modelo. O Quadro 1 apresenta a primeira divisão dos dados, destacando que 14820 imagens foram utilizadas para a construção do modelo e 780 imagens para a validação do modelo de classificação. A Gráfico 1 apresenta a porcentagem da divisão dos dados.

Quadro 1- Quantidade de dados para a construção e validação do modelo.

	Quantidade de imagens
Construção do modelo	14820
Validação do modelo	780

Fonte: Produção própria.

Gráfico 1 - Quantidade de dados para a construção e validação do modelo (%).



Fonte: Produção própria.

5.4. Criação das camadas da RNA

Após a inserção dos dados e de todo o pré-processamento, o próximo passo é a criação das camadas da RNA. A primeira camada corresponde à entrada de dados, correspondentes aos atributos das 26 classes que foram gerados a partir de suas imagens. A primeira camada não corresponde somente às imagens dos símbolos das letras, mas, inclusive das imagens transformadas, ou seja, que foram obtidas no pré-processamento.

A seguir, foram adicionadas as camadas do *MobileNetV3-Large*, contendo 263 camadas, na Figura 13 podemos ver na coluna *Layer(type)* que mostra o tipo das camadas, o primeira é simplificação da *MobileNetV3-Large*, a segunda a camada *global average pooling 2d* introduzida por Lin et al. (2013), essas camadas substituem a camada totalmente conectadas, a camada é aplicada após uma camada convolucional e calcula a média dos valores em cada mapa de característica representando de forma resumida as características

apreendidas pela rede para os neurônios de classificação na última camada e a última camada *Dense* que representa a camada de saída.

Na coluna *output Shape*, representa o tamanho de saída, sendo *None* representado pelo tamanho do lote que vai variar e será determinado pelos dados de entrada, na primeira temos o tamanho 7x7 em largura e altura e 960 é o número de canais de saída, na global *average pooling 2d* somente tem o número lotes não definido e 960 que representa o número de canais de saída e o último onde 26 representa a o número de neurônio da camada de saída onde cada neurônio representa as classes.

Por fim temos *param #* que representa o número de parâmetros, e são valores que pode ser aprendido durante o processo do treino, na figura somente temos parâmetros na *MobileNetV3-Large* e na *Dense*, já que a *global average pooling 2d* é uma camada de redimensionamento.

Figura 13 – resumo das camadas do modelo.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 MobilenetV3large (Functiona  (None, 7, 7, 960)        2996352
 l)

 global_average_pooling2d (G  (None, 960)              0
 lobalAveragePooling2D)

 dense (Dense)               (None, 26)                24986

=================================================================
Total params: 3,021,338
Trainable params: 2,644,554
Non-trainable params: 376,784
_________________________________________________________________
```

Fonte: Produção própria.

5.5. Divisão, treino e teste

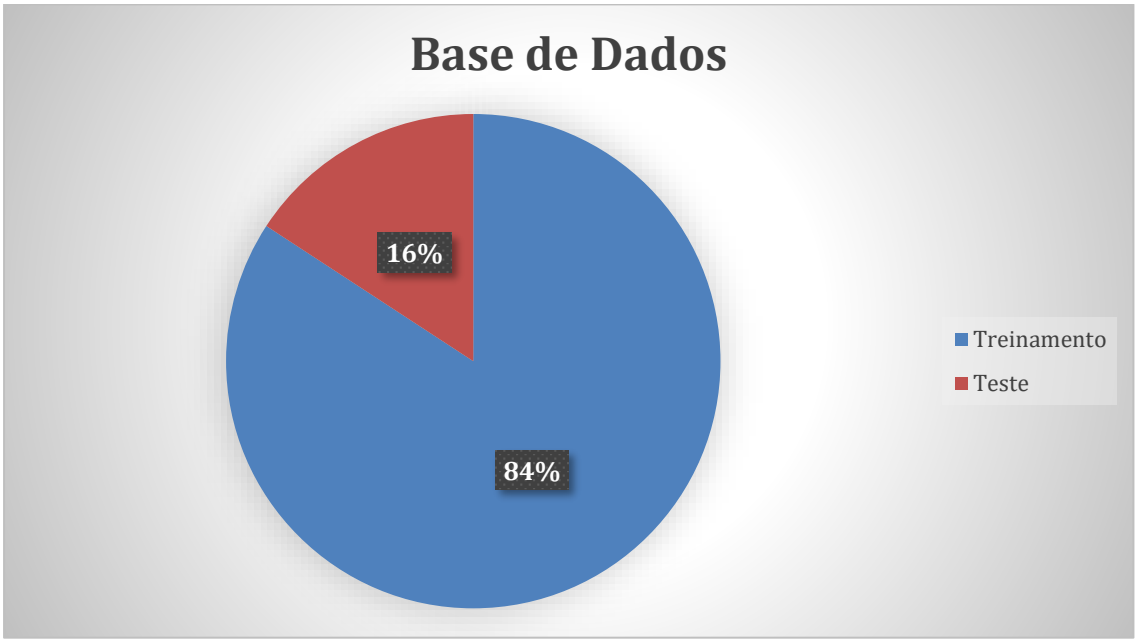
Após a criação das camadas, mais uma vez os dados foram divididos. Daquela primeira divisão, foram utilizados os dados de construção do modelo, os quais foram divididos em dados de treinamento e teste. O Quadro 2 apresenta a divisão, onde é possível verificar que foram utilizadas 12480 imagens para treinamento e 2340 imagens para teste do treinamento. Na Gráfico 2 é possível verificar que os dados de treinamento correspondem a 84,25% e aqueles utilizados no teste correspondem a 15,75%.

Quadro 2 - Divisão de dados do treinamento.

	Quantidade de imagens
Treinamento	12480
Teste	2340

Fonte: Produção própria.

Gráfico 2 - Dados de treinamento e validação do treinamento (%).



Fonte: Produção própria.

5.6. Treinamento

Para compilar o modelo foi utilizado o otimizador *Adam* (*Adaptive Moment Estimation*), introduzido por Kingma e Ba, (2014). Trata-se de um algoritmo de otimização que tem função encontrar parâmetros de um modelo e ajustar a taxa de aprendizado com base na estimativa do erro que é a diferença entra a valor previsto e o valor real. A taxa de aprendizado foi utilizada valor de 0.001 com perda definido em *Categorical Crossentropy*, o qual calcula a perda entre as classes com predições em problemas com duas ou mais classes.

No treinamento foram usadas 50 épocas, que corresponde à interação entre dados de entrada e dados de saída. Além disso, foi utilizado o método *callback ModelCheckpoint* que tem como objetivo salvar o modelo em certos intervalos com o formato do arquivo para

armazenamento de grande quantidade de dados comprimidos HDF5 (*Hierarchical Data Format version 5*) a cada aumento da acurácia.

5.7. Modelo de Classificação e Predição

Após a etapa de treinamento, foi gerado o modelo de classificação, o qual corresponde ao melhor resultado do treinamento. Esse modelo corresponde, portanto, ao classificador que será utilizado para fazer a classificação de novas imagens inseridas no projeto.

A etapa de predição corresponde ao momento que o modelo de classificação será testado com imagens que não foram utilizadas na etapa de treinamento. Portanto, ele será avaliado por meio de novas imagens, que sofreram o mesmo processo de pré-processamento que as imagens utilizadas na fase de treinamento. Para isso, foram utilizadas 780 imagens distribuídas de forma balanceada entre as 26 classes de símbolos.

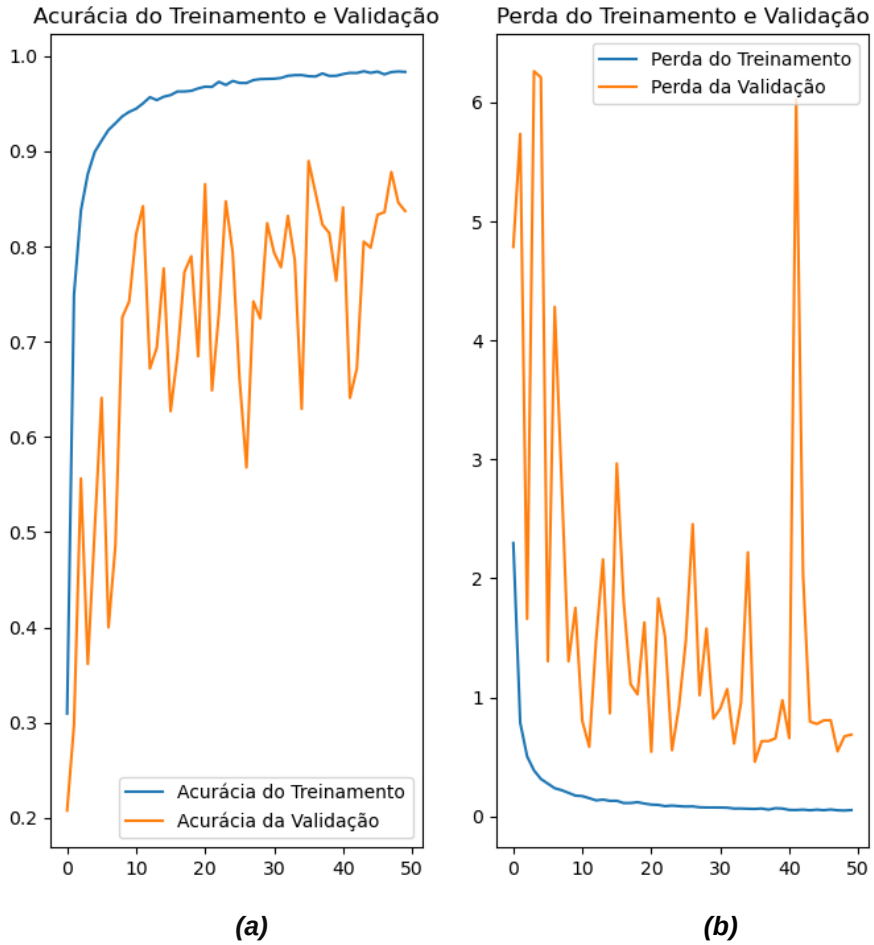
6 RESULTADOS E DISCUSSÃO

As 50 épocas utilizadas no treinamento podem ser verificadas na Figura 14, relacionadas com a acurácia e a perda do treinamento e a validação. Na Figura 14 (a), verifica-se um crescimento da acurácia do treinamento nos dez primeiros épocas, estabilizando-se a precisão acima de 90% nos próximos épocas e com melhor acurácia de treinamento com 98,4% na época 44. Já na acurácia da validação nas primeiras 5 épocas observam-se crescimento da acurácia, mas com variações, levando a variações bruscas. Entre as épocas dez e vinte também houve variação, porém apresentando pequenos crescimentos e a melhor acurácia da validação foi na época 36 com 88,97%.

A Figura 14 (b) apresenta a perda do treinamento e validação. A perda do treinamento estabiliza-se por volta da época 10, apresentando estabilização com valor bem próximo de zero. A perda da validação apresentou grande variação durante todas as épocas. Mas, ao longo do período verifica-se uma perda abaixo de 1, sendo a menor delas na época 36 com 0,45.

Uma explicação para a variação da acurácia e da perda durante o treinamento está descrito em al. (2020). Os autores afirmam que há desvio de variância quando são utilizadas juntas as camadas *dropout* e *batch normalization*. A camada *dropout* de acordo com Srivastava et al., (2014) que é uma técnica para evitar o modelo ajuste excessivamente os dados de treino assim perdendo a capacidade de generalização, essa camada funciona desativando aleatoriamente unidades de ativação de determinadas camadas. Essa junção resulta no deslocamento da variância neural à medida que as informações fluem na inferência que resulta em previsões finais inesperadas, prejudicando o desempenho. O modelo *MobileNetV3-Large* v1,0 utiliza somente *batch normalization*, mas, como verificado, sofre com o mesmo fenômeno.

Figura 14 - Gráfico de desempenho do treinamento.

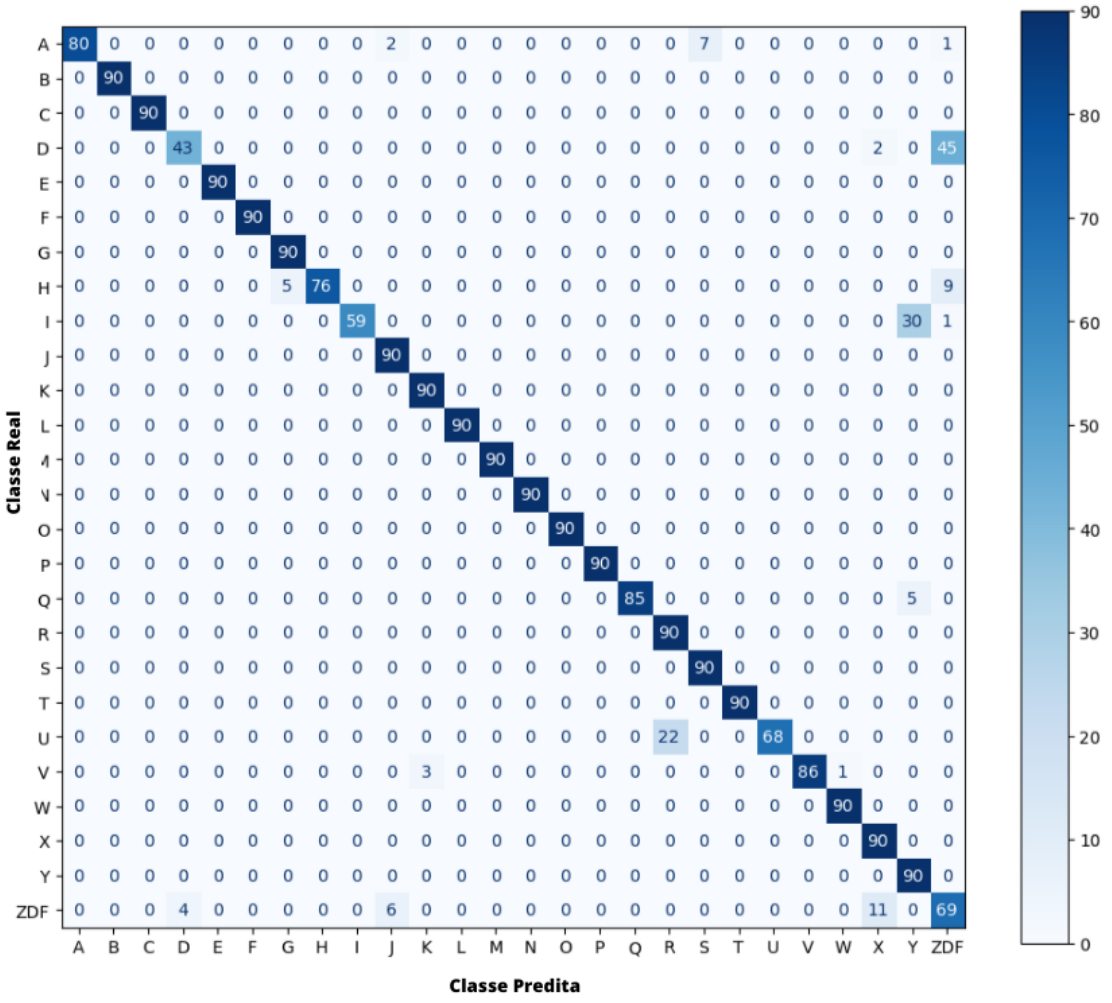


Fonte: Produção própria

Na Figura 15 é apresentada a matriz de confusão dos resultados da validação do modelo de classificação desenvolvido. Nela é possível verificar que a maior parte das classes apresentam 100% de predição correta, sendo em 18 classes. Já em 7 classes mais de 50% foram preditas corretamente. Por sua vez, apenas uma classe com índice de acerto abaixo de 50%.

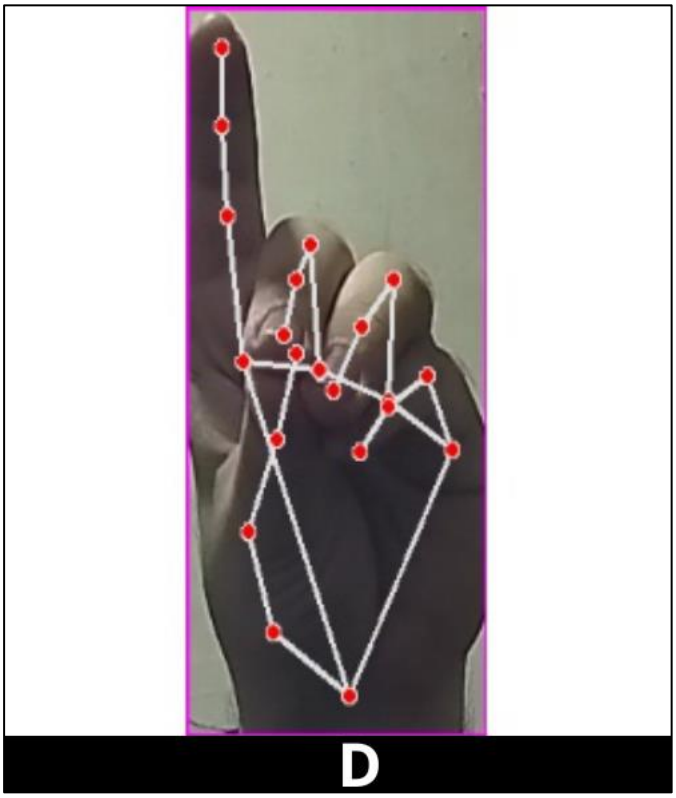
A classe D teve 43 instâncias classificadas corretamente, mas apresentou maior confusão com a classe não definida (ZDF), com 45 instâncias. A classe ZDF contém imagens que remetem a nenhum sinal de libras, sendo este um motivo que justifica a elevada quantidade de erros. Também, em relação à classe D, a questão dos dedos médio, anelar, mínimo e polegar estarem estendidos fazendo com que os pontos de marcador se sobreponham, dificultando uma boa visão da mão (Figura 16).

Figura 15 - Matriz de confusão do treinamento.



Fonte: Produção própria

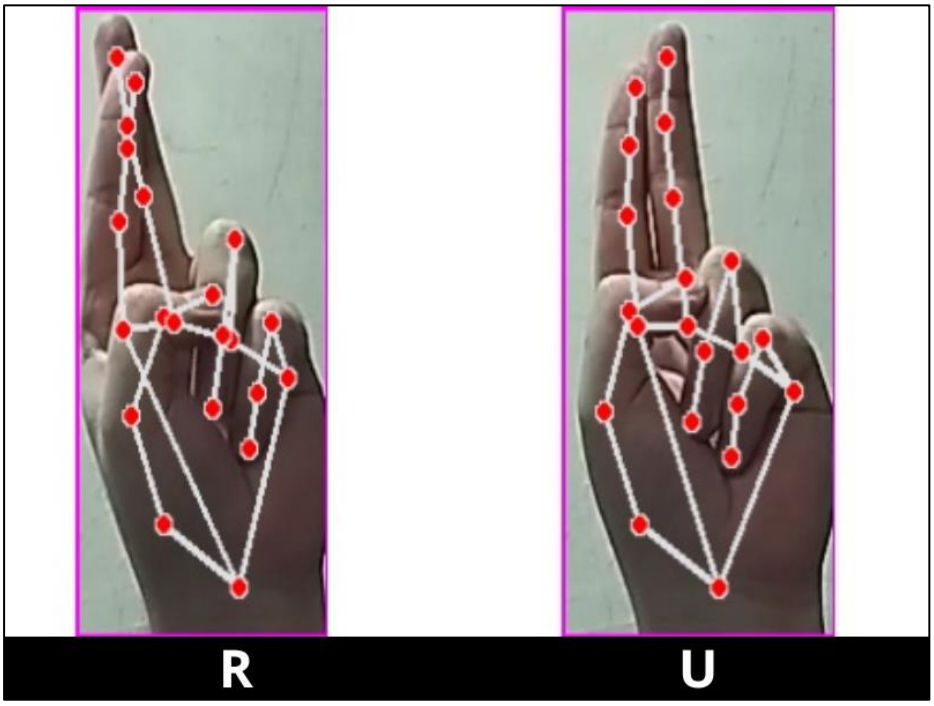
Figura 16 - Imagem da letra D em libras.



Fonte: Produção própria

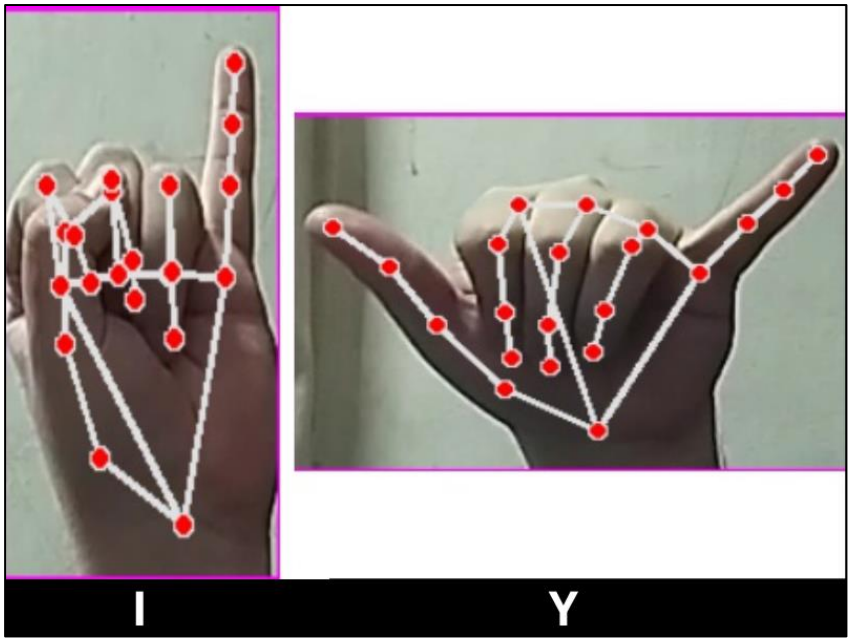
Alguns erros também são justificados pela semelhança entre os sinais. A classe U apresenta 68 instâncias classificadas corretamente e 22 instâncias classificadas erroneamente como classe R. Como é possível verificar na Figura 17 os sinais U e R são muito semelhantes. A única diferença é a flexão do dedo polegar, algo muito próximo. A classe I também apresentou muitos erros, destacando-se as confusões em relação à classe Y, 30 instâncias classificadas erroneamente como tal classe. Uma justificativa para este comportamento pode ser creditada ao dedo mínimo. Por este aparecer estendido em ambos os sinais, podendo ser considerado o fator de semelhança, como apresentado na Figura 18.

Figura 17 - Comparação das letras R e U em libras.



Fonte: Produção própria

Figura 18 - Comparação das letras I e Y em libras.



Fonte: Produção própria

A Figura 19 apresenta o reporte de validação do modelo de classificação onde a pontuação varia entre 0 e 1, contendo precisão, recall e pontuação F1 de cada classe com acurácia de teste do modelo de 93%. Na figura podemos notar as classes as classes Y e a

classes ZDF (não definida), tiveram precisão inferior a 75%, significando que o modelo está cometendo muitos erros na classificação geral, tanto exemplos positivos como negativos. As classes I, U e ZDF, obtiveram recall inferior a 80%, significando que está tendo dificuldade em encontrar exemplos positivos, resultando número alto de falsos positivos. E as classes D, I e ZDF obtiveram pontuação F1 inferior a 80% significando que o modelo está tendo um desequilíbrio entre precisão e recall.

Na Figura 19 é possível notar classes que mantiveram pontuação máxima de 1.0 nas três métricas, que significa desempenho perfeito, as classes são: B, C, E, F, L, M, N, O, P e T. Porém em certas classes como A, H, I, Q, U e V, que tiveram precisão máxima, porém o recall e pontuação f1 inferior a um, indicando que o modelo consegue evitar os falsos positivos. A classe D também pode entrar nessas classes onde mesmo não tendo pontuação máxima, a precisão foi superior ao recall e pontuação F1. Nas classes G, J, K, R, S, W e Y, o recall teve pontuação máxima, enquanto a acurácia e pontuação F1 foi inferior, indicando que o modelo está cometendo erros em exemplos falsos negativos como em falsos positivos.

Ainda na Figura 19, em sua parte inferior é possível verificar a acurácia geral do modelo em 93%, com suporte de 2340, suporte significa o número total de elementos testado no reporte. E a média macro e média dos pesos são médias gerais de cada uma da métricas avaliadas no reporte.

Figura 19 - Reporte de classificação.

Reporte de classificação

	Precisão	Recall	Pontuação F1	Suporte
A	1.00	0.89	0.94	90
B	1.00	1.00	1.00	90
C	1.00	1.00	1.00	90
D	0.91	0.48	0.63	90
E	1.00	1.00	1.00	90
F	1.00	1.00	1.00	90
G	0.95	1.00	0.97	90
H	1.00	0.84	0.92	90
I	1.00	0.66	0.79	90
J	0.92	1.00	0.96	90
K	0.97	1.00	0.98	90
L	1.00	1.00	1.00	90
M	1.00	1.00	1.00	90
N	1.00	1.00	1.00	90
O	1.00	1.00	1.00	90
P	1.00	1.00	1.00	90
Q	1.00	0.94	0.97	90
R	0.80	1.00	0.89	90
S	0.93	1.00	0.96	90
T	1.00	1.00	1.00	90
U	1.00	0.76	0.86	90
V	1.00	0.96	0.98	90
W	0.99	1.00	0.99	90
X	0.87	1.00	0.93	90
Y	0.72	1.00	0.84	90
ZDF	0.55	0.77	0.64	90
Acurácia			0.93	2340
Média Macro	0.95	0.93	0.93	2340
Média dos pesos	0.95	0.93	0.93	2340

Fonte: Produção própria

Conforme o Quadro 3, disponível em Keras (2023), as arquiteturas *MobileNet* contêm baixa quantidade de parâmetros e assim a acurácia tende a ser baixa. Na pesquisa de Howard et al. (2019) o *MobileNetV3-Large* v1.0 contém 5.4M de parâmetros com top-1 de acurácia 75,2%. Contudo a arquitetura *MobileNet* contém baixo volume do modelo, nos dados da tabela, verifica-se que *MobileNet* e *MobileNetV2* tem tamanho do arquivo em 14mb e 16mb respectivamente, e, nesse projeto, o modelo contém cerca de 3M de parâmetros e o arquivo extensão *Hierarchical Data Format version* 5 (HDF5) tem um tamanho de 12mb sendo assim mais viável para classificações em tempo real usando hardware de baixo desempenho.

.

Quadro 3 - Comparação de características das redes neurais profundas.

Modelo	Tamanho (MB)	Top-1 Acurácia	Parâmetros	Profundidade
EfficientNetB7	256	84.3%	66.7M	438
Xception	88	79.0%	22.9M	81
ResNet152V2	232	78.0%	60.4M	307
DenseNet201	80	77.3%	20.2M	402
VGG19	549	71.3%	143.7M	19
MobileNetV2	14	71.3%	3.5M	105
MobileNet	16	70.4%	4.3M	55

Fonte: modificado de Keras (2023).

A validação do treinamento apresentou 88,9% de acurácia enquanto na validação do teste do modelo apresentou 93%. Esta última acurácia indica que o modelo está conseguindo classificar melhor os dados que não faziam parte do treinamento, conforme disposto no Quadro 4 abaixo.

Existem diversas razões que podem explicar essas diferenças nas validações. Uma das possíveis causas é a distribuição das imagens nas duas etapas, podendo ser que as imagens usadas na validação do teste tenham características semelhantes às imagens usadas no treinamento, o que proporcionaria que o modelo apresentasse melhor desempenho no teste. Outra razão está associada ao baixo número de imagens na validação, um conjunto de dados pequeno resulta a estimativa de desempenho do modelo seja menos precisa. Ou seja, se fosse usado um conjunto de dados maior durante o teste, é provável que a acurácia da validação fosse menor.

Quadro 4 - Tabela contendo a acurácia do modelo no Treinamento e Teste.

Etapas de Validação	acurácia
Validação do Treinamento	88,9%
Validação do Teste	93%

Fonte: Produção própria.

Comparando os resultados apresentados com a literatura, observa-se que a utilização de rede neurais mais complexas, juntamente com diferentes técnicas para melhorar as imagens são encorajadas. Mas, mesmo apresentando limitações, este trabalho apresentou resultados coerentes com a literatura. Como exemplo avalia-se o proposto por Cui et al. (1995) onde propuseram um framework que analisa sequência de imagens usando 28 sinais diferentes e o modelo é capaz de aprender eventos sem impor modelo de movimento obteve 96 de acurácia.

Já em Cui e Weng (2000), foi utilizado um modelo para reconhecimento de 28 sinais que reconhece movimento e a forma da mão onde o modelo obteve 93,2 de acurácia. Outro trabalho é de Rastgoo et al. (2021), projeto de modelo de aprendizado usando 4 diferentes bases dados e técnicas de redução de dimensionalidade. O projeto aprende a língua de sinais a partir de vídeos RGB, o modelo estima coordenadas da mão e estima as *features*, estimando sinais em tempo real obteve 91% de acurácia.

O projeto de Zakariah et al. (2022) desenvolveu um modelo combinando várias técnicas para o processamento de imagens, o modelo atingiu 95% de precisão no teste. A publicação de Abdulhussein e Raheem (2020), usando rede neurais profundas obteve 99,3% de precisão usando técnicas de aumento de dados em reconhecimento de sinais estáticos. Outro exemplo é Daroya et al., (2018), onde propôs uma abordagem de classificação em tempo real da língua de sinais usando rede neurais profundas, alcançando 90,3% de precisão usando em imagens coloridas e aumento de dados.

Para melhor visualização as comparações com a litertura estão dispostas no quadro 5.

Quadro 5 - Comparação com trabalhos da literatura.

Autor	Acurácia
Cui et al (1995)	96%
Cui e Weng (2000)	93,2%
Daroya, Peralta e Naval, (2018)	90,3%
Abdulhussein e Raheem (2020)	99.3%
Rastgoo et al. (2021)	91%
Zakariah, et al. (2022)	95%
Este Trabalho	93%

Fonte: Produção própria

7 CONCLUSÃO

Neste trabalho foi implementado um modelo de rede neural convolucional profunda utilizando camadas do *MobileNetV3-Large* para reconhecimento de imagens de sinais do alfabeto de libras. Foi utilizada uma base de dados, de criação própria, contendo 15.600 imagens divididas em 26 classes. Foram utilizados recursos de pré-processamento com técnicas de *augmentation* para aumentar a variabilidade dos dados de entrada. Foram realizados diversos tipos de treinamento com diferentes parâmetros para teste. O treinamento que gerou o modelo de classificação obteve acurácia de 88,9%.

O modelo de classificação foi utilizado para realizar a classificação de um conjunto de dados diferente daquele utilizado no treinamento, mas que sofreu o mesmo pré-processamento. A acurácia alcançada pelo modelo foi de 93%. As principais vantagens desse modelo é a precisão da rede neural *MobileNetV3-Large*, a qual apresentou baixo número de parâmetros, podendo ser construído e processado apresentando menor tempo e custo computacional.

A plataforma *Kaggle* utilizada para o treinamento e teste do modelo apresentou uma boa performance de processamento. A conta gratuita permite o processamento de uma quantidade significativa de dados. Além disso, os recursos disponibilizados contam com processamento em CPU e GPU, sendo suficientes para o processamento dos dados do trabalho.

A principal limitação do trabalho é quanto ao uso do alfabeto de LIBRAS, já que os sinais são representados em palavras e não é soletrado. Destacam-se como possíveis avanços do projeto o uso ou criação de uma base de dados contendo sinais de palavras completas para identificação. Dessa forma contemplará os desafios de processamento nesse contexto em sua totalidade.

A partir dos resultados deste trabalho, são sugeridos os seguintes trabalhos futuros:

- Desenvolvimento de um aplicativo móvel que utilize a rede neural para reconhecer e traduzir gestos de libras em tempo real;
- Treinamento de rede neural para reconhecer gestos e expressões faciais utilizados na comunicação em língua de sinais;
- Implementação de um mecanismo de aprendizado ativo na rede neural para que ele aprenda a se adaptar a novos gestos e sinais;
- Sistema de reconhecimento de gestos de corpo inteiro para ampliar a comunicação além do alfabeto de sinais;

- Exploração do uso de técnicas de transferência de aprendizado para aproveitar o reconhecimento prévio adquirido por redes neurais profundas treinadas em reconhecimento do alfabeto de sinais e aplicá-las em tarefas relacionadas, como gestos corporais e expressões faciais;
- Aplicação da tecnologia de realidade aumentada para criar uma experiência imersiva de aprendizado de línguas de sinais usando a rede neural como parte integrando do aprendizado;
- Sistema de tradução automática entre diferentes línguas de sinais onde a rede neural é responsável por compreender e gerar os gestos corretos para cada idioma;
- Criação de um conjunto de dados maior e mais diversificado para treinar a rede neural;

REFERÊNCIAS

ABDULHUSSEIN, A. A.; RAHEEM, F. A**.** Hand Gesture Recognition of Static Letters American Sign Language (ASL) Using Deep Learning. **Engineering and Technology Journal**, v. 38, n. 6A, p. 926–937, jun. 2020.

AJI, S. et al. Deep Transfer Learning for Automated Artillery Crater Classification. **Thai Journal of Mathematics**, v. 19, n. 3, p. 1068–1081, jun. 2021.

BATISTA, G. E. de A. P. et al **Pré-processamento de dados em aprendizado de máquina supervisionado**. 2003. Tese de Doutorado. Universidade de São Paulo.

BAZZAN, A. L.C. Contribuições de aprendizado por reforço em escolha de rota e controle semafórico. **Estudos Avançados,** 35, p95-110, 2021.

BORDENAVE, J. E. D. **O que é comunicação**. São Paulo: Brasiliense, 1997.

BREIMAN, L. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5-32, 2001.

CERQUEIRA, P. H. R. **Um estudo sobre reconhecimento de padrões:** um aprendizado supervisionado com classificador bayesiano. 2010. Tese de Doutorado. Universidade de São Paulo.

CHOMSKY**,** N. **Linguagem e mente**. Brasília: UnB, 1998.

COLPO, M. P. et al. Mineração de dados educacionais na previsão de evasão: uma RSL sob a perspectiva do congresso brasileiro de informática na educação. In: **ANAIS DO XXXI SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇAO.** SBC, 2020. p. 1102-1111.

CUI, Y.; SWETS, D. L.; WENG, J. J. Learning-based hand sign recognition using SHOSLIF-M. In: **Proceedings of IEEE International Conference on Computer Vision. IEEE**, 1995. p. 631-636.

CUI, Y.; WENG, J. Appearance-Based Hand Sign Recognition from IntensityImage Sequences. **Computer Vision and Image Understanding**, v. 78, n. 2, p.157–176, maio 2000.

CVZONE. **Cvzone: software**, 2023. Disponível em: https://github.com/cvzone/cvzone. Acesso em: 09 maio 2023.

DAROYA, R.; PERALTA, D.; NAVAL, P. Alphabet Sign Language Image Classification Using Deep Learning. In: **TENCON 2018 - 2018 IEEE REGION 10 CONFERENCE**, out. 2018.

DEVELOPERS, TensorFlow. TensorFlow. **Zenodo**, 2022. Disponível em: https://ui.adsabs.harvard.edu/abs/2021zndo...4758419D/abstract. Acesso em 07 jun. 2023.

DIZEU, L. C. T. de B.; CAPORALI, S. A. A língua de sinais constituindo o surdocomo sujeito. **Educação & Sociedade**, v. 26, p. 583-597, 2005. Disponível em: <

ENGEL, G. L. **The need for a new medical model:** a challenge for biomedicine. Science, v. 196, n. 4286, p. 129-136, 1977.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI magazine**, v. 17, n. 3, p. 37-37, 1996.

FELIPE, T. A. Os processos de formação de palavra na Libras. **ETD Educação Temática Digita**l, v. 7, n. 02, p. 200-212, 2006.

FERNANDES, F. T.; CHIAVEGATTO FILHO, A. D. P. Perspectivas do uso de mineração de dados e aprendizado de máquina em saúde e segurança no trabalho. **Revista Brasileira de Saúde Ocupacional**, v. 44, 2019.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep Sparse Rectifier Neural Networks. In: **INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND STATISTICS.** JMLR Workshop and Conference Proceedings, 2011. p. 315-323.

GOLDFELD, M. **A criança surda:** linguagem e cognição numa perspectiva sócio-interacionista. São Paulo: Plexus, 1997.

GUGLIELMO, G. et al. Blink To Win: Blink Patterns of Video Game Players Are Connected to Expertise. in: **FDG '22: PROCEEDINGS OF THE 17TH INTERNATIONAL CONFERENCEON THE FOUNDATIONS OF DIGITAL GAMES**, 5 set. 2022.

HARRISON, K.M.P. O momento do diagnóstico de surdez e as possibilidadesde encaminhamento. In: **LACERDA, C.B.F.; NAKAMURA, H.; LIMA, M.C. (Org.). Fonoaudiologia: surdez e abordagem bilíngue**. São Paulo: Plexus, 2000.

HARTIGAN, J. A. et al. A k-means clustering algorithm**. Applied statistics**, v. 28, n.1, p. 100-108, 1979.

HAYKIN, S. **Neural Networks:** A Comprehensive Foundation. 2. ed. India: PrenticeHall, 1994. p. 842.

HE, K. et al. Deep Residual Learning for Image Recognition. In: **PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION.** 2016. p. 770-778.

HENDRYCKS, D.; GIMPEL, K. Gaussian Error Linear Units (GELUs). **arXiv preprint arXiv:1606.08415**. 2016.

HINTON, G. E.; OSINDERO, S.; TEH, Y. W. A Fast-Learning Algorithm for DeepBelief Nets**. Neural Computation**, v. 18, n. 7, p. 1527–1554, jul. 2006.

HOWARD, A. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. **arXiv preprint arXiv:1704.04861**, 2017.

HOWARD, A. et al. Searching for MobileNetV3. In: **Proceedings of the IEEE/CVF international conference on computer vision**. 2019. p. 1314-1324.

HU, J. et al. Squeeze-and-Excitation Networks. **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2018. p. 7132-7141.

IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: **INTERNATIONAL CONFERENCE ON MACHINE LEARNING**. pmlr, 2015.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. **Journal of artificial intelligence research**, v. 4, p. 237-285, 1996.

KAGGLE. **Notebooks Documentation**. 2023. Disponível em: <https://www.kaggle.com/docs/notebooks>. Acesso em: 12 jun. 2023.

KERAS, **Keras documentation:** Keras Applications. Disponível em: <https://keras.io/api/applications/>. Acesso em: 07 jun. 2023.

KHANORKAR, A. et al. Background removal of video in realtime. **3c TIC: cuadernos de desarrollo aplicados a las TIC**, v. 11, n. 2, p. 195–206, 2022.

KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. eprint **arXiv:1412.6980**, 22 dec. 2014.

KOHONEN, T. Self-organizing formation of topologically correct feature maps. **Biological Cybernetics**, v. 43, p. 59-69, 1982.

KONYUSHKOVA, K.; ZOLNA, K.; AYTAR, Y.; NOVIKOV, A.; REED, S.; CABI, S.; de FREITAS, N. Semi-supervised reward learning for offline reinforcementlearning. **ArXiv preprint arXiv:2012.06899**, 2020.

KOVÁCS, Z. L. **Redes neurais artificiais**. Editora Livraria da Fisica, 2002.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet classification withdeep convolutional neural networks. **Communications of the ACM**, v. 60, n. 6, p84–90, 24 maio 2012.

LABOV, W. **Padrões sociolinguísticos**. Trad.: Marcos Bagno, Maria Marta Pereira Scherre, Caroline Rodrigues Cardoso. São Paulo: Parábola, 2008.

LECUN, Y. et al. **Gradient-based learning applied to document recognition**. In: **PROCEEDINGS OF THE IEEE**, v. 86, n. 11, p. 2278–2324, 1998. Disponível em: < https://ieeexplore.ieee.org/abstract/document/726791 >. Acesso em: 07 jun. 2023.

LI, X.; CHEN, S.; YANG, J. Understanding the Disharmony between Weight Normalization Family and Weight Decay. In: **PROCEEDINGS OF THE AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE,** v. 34, n. 04, p. 4715–4722, 3 abr. 2020.

LI, Y. Reinforcement learning applications. **arXiv preprint arXiv:1908.06973**, 2019.

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

LIU, S.; et al. Integrating Dijkstra's algorithm into deep inverse reinforcement learning for food delivery route planning. **Transportation Research Part E: Logistics and Transportation Review** 142, 2020.

LUGARESI, C. et al. MediaPipe: A Framework for Building Perception Pipelines. **arXiv:1906.08172**, 14 jun. 2019.

MAHMUD, M.; KAISER, M. S.; HUSSAIN, A.; VASSANELLI, S. Applications of Deep Learning and Reinforcement Learning to Biological Data. in: **IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS,** vol. 29, no. 6, pp. 2063-2079, June 2018.

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,dez. 1943.

MONARD, M. C.; BARANAUSKAS, J. A. Sistemas inteligentes - Fundamentos e aplicações. Chapter Conceitos sobre Aprendizado de Máquina. **Rezende (2003)**, v. 15, p. 89-114, 2003.

PRETI, D. (Org.). **Estudos de língua falada:** variações e confrontos. São Paulo: Humanitas, 1998.

PRIDDY, K. L.; KELLER, P. E. **Artificial neural networks:** an introduction. SPIE press, 2005.

QUADROS, R.M. **Educação de surdos**:a aquisição da linguagem. Porto Alegre: Artes Médicas, 1997.

QUINLAN, J. R. Induction of decision trees. **Machine Learning**, v.1, n. 1, p. 81 – 106, 1986.

QUINLAN, J. R. Learning with continuous classes. In: **5TH AUSTRALIAN JOINT CONFERENCEON ARTIFICIAL INTELLIGENCE**. Vol. 92. 1992.

RAMOS, C. R. LIBRAS: A língua de sinais dos surdos brasileiros. 2004. **Disponível para download na página da Editora Arara Azul: http://www.editora-arara-azul.com.br/pdf/artigo2.pdf**, 2004. Acesso em: 10 jun.
2023.

RASTGOO, R.; KIANI, K.; ESCALERA, S. Real-time isolated hand sign language recognition using deep networks and SVD. **Journal of Ambient Intelligence and Humanized Computing**, v. 13, n. 1, p. 591–611, 16 fev. 2021.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychol Rev**. 65 nov. 1958.

ROSSIT, B. N; DOS SANTOS OLIVEIRA, E.; VASCONCELLOS, J. G. M. Talkbox: tradução de libras com Machine Learning. 2022. Disponível em: < https://repositorio.maua.br/handle/MAUA/414>. Acesso em: 10 jun. 2023.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representationsby back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, out. 1986.

SACKS, O. **Vendo vozes**: uma viagem ao mundo dos surdos. São Paulo: Cia das Letras, 1998.

SANDLER, M. et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2018. p. 4510-4520.

SAUSSURE, F. de. **Curso de Linguística Geral.** Tradução de Carlos Alberto Faraco e Cristóvão Tezza. 53ª ed. São Paulo: Cultrix, 2008.

SHARON, G. et al. Real-time Adaptive Tolling Scheme for Optimized Social Welfare in Traffic Networks. In: **PROCEEDINGS OF THE 16TH INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS-2017)**, 16., 2017, São Paulo. p.828-36.

SIFRE, L.; MALLAT. Rigid-Motion Scattering For Image Classification. **arXiv preprint arXiv:1403.1687**, 2014.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large- Scale Image Recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SONKA, M.; FITZPATRICK, J. M. **Handbook of medical imaging: volume 2. medical image processing and analysis**. 1. ed. San Jose: Spie Press, 2000. v. 2p.567–605.

SRIVASTAVA, N. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal of Machine Learning Research**, v. 15, n. 56, p. 1929–1958, 2014.

STOKOE Jr.; William C. Sign language structure: An outline of the visual communication systems of the American deaf. **Journal of deaf studies and deaf education**, v. 10, n. 1, p. 3-37, 2005.

STRAND, F.; KARLSSON, J. Visual Detection of Personal Protective Equipment& Safety Gear on industry Workers**. arXiv preprint arXiv:2212.04794**. 9 dec. 2022.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning:** An Introduction. Cambridge, MA, USA: MIT Press, 1998.

TAN, M. et al. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. 2019. p. 2820-2828.

TAN, P.N.; STEINBACH, M.; KUMAR, V. **Introdução ao datamining**: mineracão de dados. Ciencia Moderna, São Paulo, 2009.

VYGOTSKY, L.S. **Pensamento e linguagem**. São Paulo:Martins Fontes; 1987. Disponível em: < http://www.institutoelo.org.br/site/files/publications/5157a7235ffccfd9ca905e359020c413.pdf>. Acesso em: 10 jun. 2023.

WATKINS, C. JCH; DAYAN, P. Q-learning. **Machine learning**, v. 8, p. 279-292, 1992. Disponível em: < https://link.springer.com/article/10.1007/BF00992698>. Acesso em: 10 jun. 2023.

WEI, Hua et al. A survey on traffic signal control methods. **ArXiv preprint**

arXiv:1904.08117, 2019.

WEISS, S. M.; CASIMIR A. K. **Computer systems that learn:** classification and prediction methods from statistics, neural nets, machine learning, and expert systems. Morgan Kaufmann Publishers Inc., 1991.

WITTEN, I. H.; FRANK, E. **Data mining**: Practical machine learning tools and techniques with JAVA implementations. San Francisco: Morgan Kaufmann, 2000. 371p.

YUNUS, R. et al. A Framework to Estimate the Nutritional Value of Food in Real Time Using Deep Learning Techniques. **IEEE Access**, v. 7, p. 2643–2652, 2019.

ZAKARIAH, M. et al. Sign Language Recognition for Arabic Alphabets UsingTransfer Learning Technique. **Computational Intelligence and Neuroscience**, v.2022, p. 1–15, 22 abr. 2022.

ZHANG, A.; LIPTON, Z. C.; Li, M.; SMOLA, A. J. Dive into deep learning. **arXiv preprint arXiv:2106.11342**, 2021.

ZHANG, F. et al. MediaPipe Hands: On-device Real-time Hand Tracking. **arXiv:2006.10214**, 17 jun. 2020.