INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MATO GROSSO DO SUL CAMPUS CORUMBÁ

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

WILLIAN MARQUES DE CARVALHO JUNIOR

RECONHECIMENTO DE GESTOS ESTÁTICOS DO ALFABETO DE LIBRAS: UMA ABORDAGEM COMPARATIVA ENVOLVENDO TÉCNICAS DE APRENDIZADO DE MÁQUINA E APRENDIZADO PROFUNDO

CORUMBÁ - MS
2021

WILLIAN MARQUES DE CARVALHO JUNIOR

RECONHECIMENTO DE GESTOS ESTÁTICOS DO ALFABETO DE LIBRAS: UMA ABORDAGEM COMPARATIVA ENVOLVENDO TÉCNICAS DE APRENDIZADO DE MÁQUINA E APRENDIZADO PROFUNDO

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso do Sul – Câmpus Corumbá – como um dos requisitos para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Heitor Scalco Neto

CORUMBÁ - MS
2021

AGRADECIMENTOS

Agradeço, antes de tudo, a Deus.

Aos meus pais, William de Carvalho Prado e Edna Marques de Alvarenga Carvalho, por acreditarem em mim, pelo incentivo e apoio aos estudos sempre.

Aos meus irmãos Luis Henrique e Guilherme Otávio, pelo companheirismo e camaradagem.

A instituição de ensino IFMS pela oportunidade.

Ao meu orientador, Heitor Scalco Neto, pelas inúmeras horas de contribuição, paciência e confiança.

Aos meus amigos de graduação, pela amizade e apoio.

Aos professores Diego Saqui e Anderson Pereira, pela contribuição e conhecimentos transmitidos durante a Iniciação Científica.

A todos os professores que de alguma forma contribuíram para a minha formação.

RESUMO

A Libras, língua de sinais usada por surdos e deficientes auditivos brasileiros, foi reconhecida oficialmente em 2002, e desde então, diversas tecnologias têm surgido com o intuito de auxiliar o aprendizado e cotidiano desta comunidade. O uso do aprendizado de máquina (*machine learning*) e visão computacional (*computer vision*) para o desenvolvimento de sistemas inteligentes, como reconhecimento de gestos, tem-se mostrado eficaz. Este trabalho apresenta um comparativo com base no alfabeto de Libras, entre as seguintes técnicas de inteligência computacional (IC): redes neurais densas (DNN), redes neurais convolucionais (CNN) e florestas aleatórias (RF). As técnicas de IC são aplicadas e comparadas, com o intuito de avaliar os métodos para a tarefa de reconhecimento de gestos estáticos do alfabeto de Libras. Os resultados alcançados indicaram uma melhor eficácia de classificação utilizando a técnica de redes neurais convolucionais, a qual obteve cerca de 99,07% de acerto no conjunto de treino e 98,50% de acerto no conjunto de teste, além disto, apresentou baixo índice de falsos positivos e falsos negativos. Ademais, o desempenho da técnica por florestas aleatórias provou ser uma alternativa, com taxas de acerto nos conjuntos de treino e teste de respectivamente, 97,62% e 96,44%.

Palavras-chave: Libras, Aprendizado de Máquina, Visão Computacional, Redes Neurais Densas, Redes Neurais Convolucionais, Florestas Aleatórias.

ABSTRACT

Libras, a sign language used by deaf and hearing impaired Brazilians, was officially recognized in 2002, and since then, several technologies have emerged with the aim of helping the learning and daily life of this community. The use of machine learning (ML) and computer vision (CV) for the development of intelligent systems, such as gesture recognition, has been shown to be effective. This work presents a comparison, based on the Libras alphabet, between the following computational intelligence (CI) techniques: dense neural networks (DNN), convolutional neural networks (CNN) and random forests (RF). CI techniques are applied and compared, with the aim of evaluating the methods for the static gesture recognition task in the Libras alphabet. The results achieved indicated a better classification efficiency using the technique of convolutional neural networks, which obtained about 99.07% of accuracy in the training set and 98.50% of accuracy in the test set. low rate of false positives and false negatives. Furthermore, the performance of the technique by random forests proved to be an alternative, with hit rates in the training and test sets of, respectively, 97.62% and 96.44%.

Keywords: Libras, Machine Learning, Computer Vision, Dense Neural Networks, Convolutional Neural Networks, Random Forests.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Segundo o último censo realizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE), em 2010, cerca de 9,7 milhões de pessoas possuem alguma deficiência auditiva, o que representa mais de 5% da população do país. Deste indicativo, sensivelmente 7,6% são totalmente surdas, representando aproximadamente 3,4 milhões de pessoas (SIDRA, 2010). Um importante instrumento de comunicação para surdos é a língua brasileira de sinais (Libras). De acordo com Carvalho (2007), Libras é a língua de sinais usada por deficientes auditivos brasileiros e foi criada em parceria com o Instituto Nacional de Educação de Surdos (INES), desenvolvida a partir de uma mistura entre a língua francesa de sinais e gestos já utilizados no Brasil.

A comunicação por Libras foi reconhecida oficialmente em 2002 e, segundo a lei 10.436/2002, Libras é entendida como uma forma oficial de comunicação e expressão, com estrutura gramatical própria, constituindo um sistema linguístico de transmissão de ideias e fatos, oriundos de comunidades de pessoas surdas do Brasil (BRASIL, 2002). Embora existam leis e medidas que garantam o uso de Libras, em diversos contextos, ainda existe uma barreira de acessibilidade para a comunidade surda: a falta de ouvintes que compreendem Libras. Essa barreira é oriunda da falta de intérpretes em locais como: comércios, academias, escolas e outros.

Sendo assim, diversas tecnologias têm surgido para auxiliar a comunidade surda, como o *VLibras* (VLIBRAS, 2016) e o *Hand Talk* (HANDTALK, 2012). O VLibras é constituído por um conjunto de ferramentas utilizadas na tradução automática do português para libras (VLIBRAS, 2016). O *Hand Talk* é um aplicativo que traduz frases de português para libras. A proposta destes é ajudar na comunicação entre as pessoas e fazer com que os usuários conheçam um pouco mais sobre essa língua. Por meio de áudio ou do texto digitado, o serviço consegue, com ajuda de um avatar, transformar o áudio ou texto em sinais (HANDTALK, 2012).

Duas relevantes linhas de pesquisa que envolvem o reconhecimento de gestos e tecnologias são: visão computacional e aprendizado de máquina. A visão computacional é definida como a ciência das máquinas que enxergam e, segundo Prince (2012), tem o objetivo de extrair informações úteis de imagens, permitindo tomar decisões sobre o ambiente e cenas por meio de imagens e vídeos (LINDA, 2001). O aprendizado de máquina é um campo oriundo da inteligência computacional, que envolve a construção de algoritmos que permitem a máquina aprender sem que sejam programados explicitamente (SIMON, 2013).

Diversos trabalhos discorrem a respeito do reconhecimento de gestos em imagens, sobretudo, com foco no alfabeto manual das línguas de sinais. Passos e Comunello (2019) apresentaram 90,38% de acurácia no reconhecimento de gestos do alfabeto americano, utilizando arquiteturas de redes neurais convolucionais pré-treinadas. Dias et al. (2020) apresentaram 96,19% de acurácia no reconhecimento de gestos de Libras utilizando redes neurais densas, porém com extração de recursos tradicionais. Pessoa et al. (2016) apresentaram 84% de acurácia no reconhecimento

de gestos de Libras utilizando a técnica por máquina de vetores de suporte (SVM) e extração tradicional. Porém, os estudos desenvolvidos nesses trabalhos não abordam um comparativo entre técnicas de aprendizado de máquina utilizando extração de recursos automático. Além disso, não foi encontrado a aplicação da técnica de Florestas Aleatórias combinado a filtros de convolução para o reconhecimento de sinais. Desta forma, abre-se a oportunidade para o estudo do comparativo dessas técnicas, inclusive em ambiente real, tema abordado neste trabalho.

2 OBJETIVOS

2.1 Objetivo Geral

Analisar as técnicas de redes neurais convolucionais (CNN), redes neurais densas (DNN) e florestas aleatórias na tarefa de classificação de gestos do alfabeto manual.

2.2 Objetivos Específicos

- Reconhecer padrões de Libras por meio da técnica de Redes Neurais Convolucionais.
- Reconhecer padrões de Libras por meio da técnica de Redes Neurais Densas.
- Reconhecer padrões de Libras por meio da técnica de Florestas Aleatórias.
- Testar o reconhecimento de Libras em ambiente real, integrando com uma *webcam*.
- Comparar as técnicas aplicadas, a fim de estabelecer a melhor técnica para o reconhecimento gestos estáticos de Libras.

3 REFERENCIAL TEÓRICO

Esta seção apresenta o referencial teórico sobre os conceitos e as técnicas utilizadas para a metodologia deste trabalho. A estrutura do texto está organizada da seguinte forma: nas primeiras seções são apresentados aspectos sobre a língua brasileira de sinais, tais como o histórico e a sua importância no cotidiano brasileiro. Em seguida, a partir da seção 3.2, são apresentados os campos e as aplicações da visão computacional. A partir da seção 3.3 discute-se os principais marcos e conceitos de aprendizado de máquina, incluindo as técnicas de redes neurais artificiais (RNA), redes neurais profundas (DNN - *Deep Neural Networks*), Redes Neurais Convolucionais (CNN - *Convolutional Neural Network*) e, por fim, a técnica de florestas aleatórias (RF - *Random Forests*).

3.1 Língua brasileira de sinais

O contexto histórico da língua de sinais envolve a jornada dos surdos no Brasil. Até o século XV, os surdos eram mundialmente considerados como ineducáveis. A partir do século XVI, com mudanças nessa visão acontecendo na Europa, essa ideia foi deixada de lado. Teve início a luta pela educação dos surdos, na qual ficou marcada a atuação de um surdo francês, chamado Eduard Huet. Em 1857, Huet veio ao Brasil a convite de D. Pedro II para fundar a primeira escola para surdos do país, chamada na época de Imperial Instituto de Surdos Mudos. Com o passar do tempo, o termo "surdo-mudo" saiu de uso por ser incorreto, mas a escola seguiu forte e funciona até hoje, com o nome de Instituto Nacional de Educação de Surdos (INES) (BOGAS, 2020).

Sendo assim, com a influência francesa de Huet, a língua brasileira de sinais (Libras) foi então criada, junto com o INES, a partir de uma mistura entre a Língua Francesa de Sinais e de gestos já utilizados pelos surdos brasileiros. Com o decorrer dos anos, foi ganhando espaço, porém sofreu uma grande derrota em 1880. Um congresso sobre surdez, em Milão, proibiu o uso das línguas de sinais no mundo, acreditando que a leitura labial era a melhor forma de comunicação para os surdos. Isso consequentemente atrasou a difusão da língua no país. Com a persistência do uso e uma crescente busca por legitimidade da língua de sinais, a Libras voltou a ser aceita. A luta pelo reconhecimento da língua, no entanto, não parou e em 1993 um projeto de lei buscava regulamentar o idioma no país. Quase dez anos depois, em 2002, a Libras foi finalmente reconhecida como uma língua oficial do Brasil (BOGAS, 2020).

Dessa maneira, desde seu reconhecimento oficial, a Libras vem ganhando cada vez mais atenção, pode-se destacar alguns marcos:

- **2004**: Lei que determina o uso de recursos visuais e legendas nas propagandas oficiais do governo (BRASIL, 2004);

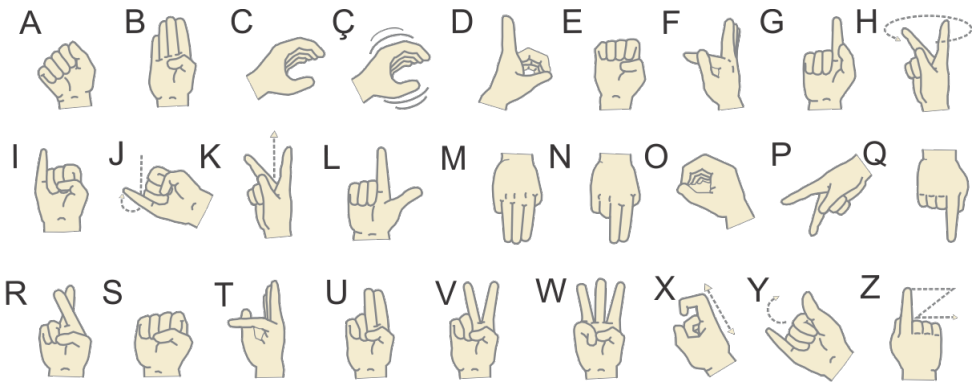
- **2008**: Instituído o Dia Nacional do Surdo, comemorado em 26 de Setembro, considerado o mês dos surdos (BRASIL, 2008);
- **2010**: Foi regulamentada a profissão de Tradutor e Intérprete de Libras (BRASIL, 2010);
- **2015**: Publicação da Lei Brasileira de Inclusão (ou Estatuto da Pessoa com Deficiência), que trata da acessibilidade em áreas como educação, saúde, lazer, cultura, trabalho etc (BRASIL, 2015);
- **2016**: Anatel publica resolução com as regras para o atendimento das pessoas com deficiência por parte das empresas de telecomunicações (ANATEL, 2016).

A Libras, assim como outras línguas de sinais, possui uma gramática própria, e se apresenta estruturada em todos os níveis, como as línguas orais: fonológico, morfológico, sintático e semântico (GESSER, 2009). Portanto o uso de tecnologias para o reconhecimento de gestos do mesmo não é algo trivial, sendo assim, neste trabalho a ênfase se estabelece no alfabeto manual.

3.1.1 Alfabeto Manual

Segundo PINHEIRO (2010), o alfabeto manual, também denominado datilologia, é um sistema de configurações da mão, que representa as letras do alfabeto da língua oral escrita. No Brasil, o alfabeto manual é composto de 27 formatos, dentre eles gestos estáticos e dinâmicos, sendo que cada um corresponde a uma letra do alfabeto do português brasileiro. O alfabeto manual é um recurso originário das línguas orais escritas, sendo assim, não desempenha nenhum papel na construção da língua brasileira de sinais. Por essa razão, é comumente usado pela comunidade surda em casos onde não exista um sinal específico para a representação de algo, ou ainda para soletrar nomes próprios ou de lugares (GESSER, 2009). A Figura 1 apresenta o alfabeto manual da língua portuguesa brasileira.

Figura 1 – Alfabeto manual da língua portuguesa brasileira



Fonte: (GODOI; LIMA; ANDRADE, 2016)

3.2 Visão Computacional

A visão humana pode ser considerada o sentido mais poderoso na captura de informações, oriundas do ambiente que nos cerca. Essa capacidade humana em processar e interpretar imensas quantidades de dados visuais, impulsionou o desenvolvimento de técnicas para compreender e simular a visão humana, dessa forma surgiu o campo da visão computacional (VC) (MOLZ, 2001). Segundo Ballard e Brown (1982), visão computacional é a ciência que estuda e desenvolve tecnologias, que permitem as máquinas enxergar e extrair características do meio, através de imagens capturadas por câmeras, sensores, entre outros dispositivos.

Uma tecnologia frequentemente utilizada em conjunto com a visão computacional é o aprendizado de máquina (AM). A utilização de métodos de aprendizado de máquina em sistemas de visão computacional, permite o desenvolvimento de sistemas de reconhecimento de padrões (RP) (SEBE et al., 2005). O reconhecimento de padrões é uma disciplina científica onde o principal objetivo é a classificação de objetos (THEODORIDIS; KOUTROUMBAS, 2008).

3.3 Aprendizado de Máquina

O termo aprendizado de máquina (ou *Machine Learning*), foi apresentado pela primeira vez em 1959, por Arthur Samuel, o qual foi pioneiro na área de aprendizado máquina voltado para jogos de computador. Ele definiu aprendizado de máquina como o campo de estudo que envolve a construção de algoritmos que permitem a máquina aprender sem que sejam programados explicitamente (SIMON, 2013).

Embora houvessem estudos elaborados a cerca do tema, o sentido de aprendizado ainda era muito vago. Assim Mitchell et al. (1997) apresentaram um conceito formal para aprendizado: "um programa de computador aprende a partir de uma experiência E, com respeito a uma classe de tarefas T e com medida de desempenho D, se seu desempenho D na tarefa T melhora com a experiência E". Logo, podemos destacar os três pilares da aprendizagem de máquina: a fonte de experiência, a tarefa a ser realizada e a medida de desempenho a ser otimizada.

3.3.1 Classe de tarefas

Em suma, a tarefa representa o processo no qual busca-se automatizar. Existem diversos algoritmos de aprendizado de máquina, no qual permitem que diferentes tipos de tarefas sejam resolvidas, como:

- **Classificação**: Neste tipo de tarefa, basicamente o programa recebe dados de entrada, na qual deve especificar a quais rótulos (classes) esses dados pertencem (GOODFELLOW et al., 2016). Um exemplo de classificação é o reconhecimento de objetos, na qual normalmente tem-se como entrada imagens que fornecem características com base nos valores de pixels e profundidade.

- **Regressão**: Neste tipo de tarefa, o programa recebe dados de entrada, e deve prever algum valor numérico. Muito semelhante a classificação, porém o formato da saída é diferente (GOODFELLOW et al., 2016). Este tipo de tarefa é bastante usado para realizar previsões, por exemplo da bolsa de valores.

- **Tradução Automática**: Neste tipo de tarefa, o programa recebe como entrada uma sequência de símbolos em um idioma e visa converter essa sequência em outro idioma (GOODFELLOW et al., 2016). Por exemplo a tradução do inglês para o português, ou Libras para português.

3.3.2 Fonte de experiência

Essencialmente, a experiência indica a fonte de dados em que o algoritmo de aprendizado de máquina é submetido no processo de aprendizagem (GOODFELLOW et al., 2016). Além disso, o tipo de experiência é categorizado em: aprendizado supervisionado e aprendizado não-supersionado (BRAGA et al., 2007).

- **Aprendizado supervisionado**: Algoritmos que abordam o método de aprendizado supervisionado, experimentam um conjunto de dados onde existem características, porém cada amostra está diretamente rotulada, ou seja, cada exemplo possui uma saída previamente estabelecida (GOODFELLOW et al., 2016). Alguns exemplos de algoritmos que abordam esse método são: *Decision Trees, Naive Bayes, k-nearest neighbors (KNN), Perceptron, MultiLayer Perceptron (MLP) e Support Vector Machine (SVM)*.

- **Aprendizado não-supervisionado**: Algoritmos que adotam o método de aprendizado não-supervisionado, experimentam um conjunto de dados contendo muitas características, e através dessas, buscam aprender propriedades úteis da estrutura desse conjunto (GOODFELLOW et al., 2016). Um dos aspectos do aprendizado não-supervisionado é que os conjuntos de dados utilizados não possuem nenhum rótulo (classe). Alguns exemplos de algoritmos que adotam esse método são: *K-Means, DBScan, Single Linkage, Complete Linkage*.

3.3.3 Medida de desempenho

Para avaliar se determinado algoritmo consegue realizar uma tarefa de forma satisfatória, é necessário uma medida quantitativa de seu desempenho. Geralmente essa medida de desempenho é específica para a tarefa a ser realizada. Sendo assim, em situações onde o algoritmo precisa classificar algo, mede-se a acurácia (*accuracy*) do modelo. A acurácia é a proporção de amostras de teste para os quais o modelo produz a saída correta (GOODFELLOW et al., 2016).

3.4 Redes Neurais Artificiais

Dentre as inúmeras máquinas conhecidas, o cérebro humano é uma das mais poderosas e complexas, sendo capaz de realizar o processamento de um grande número de informações em tempo mínimo. Umas das principais estruturas do cérebro são os neurônios, os quais são responsáveis por transmitir e processar as informações. Um aspecto essencial conhecido a respeito do funcionamento do cérebro humano, refere-se que o mesmo, desenvolve suas regras através da experiência adquirida em eventos explorados anteriormente (DEEPLEARNINGBOOK, 2021).

Por meio deste modelo, diversos pesquisadores buscaram simular o processo de aprendizagem por experiência, tendo em vista o desenvolvimento de sistemas inteligentes capazes de realizar tarefas como reconhecimento de padrões, classificação, processamento de imagens, entre outros. Logo, em decorrência destes estudos, surge o modelo do neurônio artificial, e mais tarde, um sistema com vários neurônios interconectados, denominado Rede Neural Artificial (RNA) (DEEPLEARNINGBOOK, 2021).

Inicialmente, em 1943, Warren McCulloch e Walter Pitts, propuseram o primeiro modelo matemático de um neurônio biológico. O modelo consistia em uma rede neural simples, construída através de circuitos elétricos, baseado-se em matemática e algoritmos conhecidos como lógica de limiar (*theshold logic*) (BRAGA et al., 2007). Em consequência deste estudo, as pesquisas sobre as redes neurais foram divididas em duas abordagens: uma voltada para os processos biológicos no cérebro e outra em aplicações à Inteligência Computacional (IC) (DEEPLEARNINGBOOK, 2021).

Anos mais tarde, em 1949, Donald Hebb publicou *The Organization of Behavior*, e apontou pela primeira vez uma formulação clara de uma regra de aprendizagem fisiológica para a modificação sináptica. Em suma, Hebb propôs que a conectividade do cérebro é continuamente modificada conforme um organismo vai aprendendo tarefas funcionais e que agrupamentos neurais são criados por tais modificações. Dessa forma, a eficiência de uma sinapse variável entre dois neurônios é aumentada pela ativação repetida de um neurônio pelo outro neurônio, através daquela sinapse (HAYKIN, 2007).

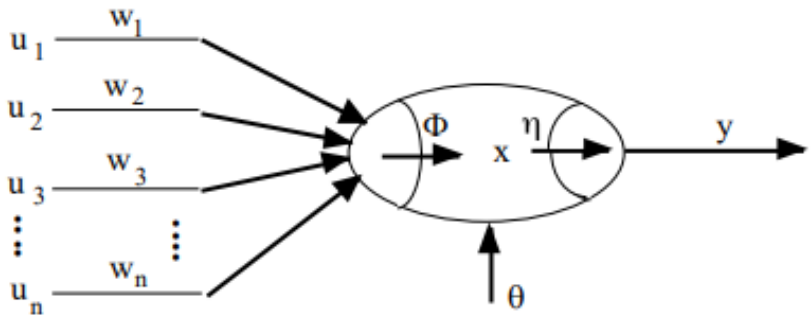
Com os estudos mais sólidos e os avanços de *hardware* computacional na década de 1950, houve a primeira tentativa de reproduzir o modelo da rede neural. Apesar das expectativas, o estudo dirigido por Nathanial Rochester, nos laboratórios de pesquisa da IBM, falharam. Somente em 1956 com o Projeto Dartmouth, ocorreu um impulso tanto em Inteligência Computacional quanto em Redes Neurais, resultando em pesquisas voltadas para o processamento neural (DEEPLEARNINGBOOK, 2021). Finalmente em 1958, o neurobiologista Frank Rosenblatt, inspirado pelos trabalhos anteriores de McCulloch e Pitts, propôs o perceptron, o primeiro modelo de RNA (BRAGA et al., 2007).

3.4.1 *Perceptron*

O neurônio do tipo *perceptron*, proposto por Frank Rosenblatt (1958), consiste na arquitetura de RNA mais básica existente, portanto seu estudo torna-se essencial para compreender o funcionamento de arquiteturas mais complexas (NIELSEN, 2015). Em suma, um perceptron recebe várias entradas binárias e produz uma única saída binária, conforme ilustrado na Figura 2. De forma análoga a um neurônio biológico, um perceptron é composto basicamente por dentritos, corpo da célula e o axônio (BRAGA et al., 2007).

Os dendritos são responsáveis por conduzir os sinais de entrada para o corpo celular. O corpo celular é encarregado de realizar o somatório (processamento) dos dados. E por último o axônio, que por sua vez transmite o sinal resultante do corpo celular para uma terminação (saída) ou ainda, para outros neurônios, neste caso, formando uma rede de neurônios (BARRETO, 2002).

Figura 2 – Neurônio artificial



Fonte: (BARRETO, 2002)

Para calcular a saída, Rosenblatt propôs uma regra simples, que consiste na atribuição de pesos (u) a cada entrada (w), expressando assim a importância das respectivas entradas para a saída (y). Dessa forma, o valor de saída (0 ou 1) é calculado através de uma soma ponderada $\sum_j w_j u_j$ representada por Φ, sendo maior ou menor que um limiar η (*threshold*). O limiar assim como os pesos é um parâmetro ajustável do modelo (NIELSEN, 2015).

A regra do limiar para o modelo pode ser definida como:

$$output = \begin{cases} 0 \; if \; \sum_j \; w_j x_j \leq threshold \\ 1 \; if \; \sum_j \; w_j x_j \geq threshold \end{cases}$$

Deste modo, nota-se que o perceptron é um modelo que toma decisões pesando as evidências, portanto através do ajuste dos pesos e do limiar, podemos obter diferentes modelos de tomada de decisão (NIELSEN, 2015). Para simplificar a regra do modelo, duas alterações são empregues por convenção. Primeiramente, os pesos e as entradas e são definidos como um produto escalar, então $w \cdot x \equiv \sum_j w_j x_j$, dessa forma w e x são vetores contendo os respectivos

elementos. Para completar, o limiar é movido para o outro lado da desigualdade, e substituído pelo que é conhecido de *bias* (viés) do perceptron (NIELSEN, 2015).

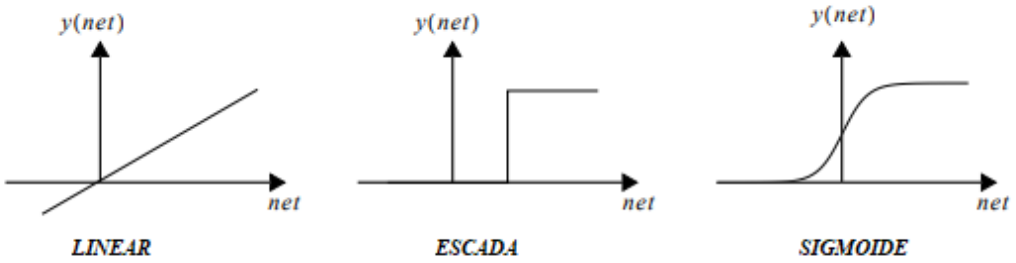
Utilizando o viés em vez do limiar, a regra do perceptron pode ser reescrita:

$$output = \begin{cases} 0 \; if \; w \cdot x + b \leq 0 \\ 1 \; if \; w \cdot x + b \; > 1 \end{cases}$$

O *bias* (viés) é definido como uma medida de tendência do modelo, onde um viés alto, tende a facilitar o perceptron a disparar 1, enquanto um viés mais baixo, dificulta o disparo do mesmo (BRAGA et al., 2007). Conforme os modelos apresentados por McCulloch-Pitts e Frank Rosenblatt, surgiram outros modelos que possibilitaram com que as saídas não fossem somente, 0 ou 1, através do uso de diferentes funções de ativação (BRAGA et al., 2007).

A Figura 3 demonstra algumas das funções de ativação que são aplicadas na saída de uma RNA para modelar os dados de saída. Conforme a Figura 3, nota-se respectivamente a representação gráfica de uma função linear, escada (degrau) e sigmoide. Dentre estas funções, a função sigmoide é a mais utilizada, já que abrange valores de 0 a 1, de modo a aproximar-se de problemas reais (BRAGA et al., 2007).

Figura 3 – Funções de ativação



Fonte: Modificado de Rauber (2005)

Uma das principais características das RNAs refere-se a sua capacidade de generalização (BRAGA et al., 2007). Generalização pode ser definida pela aptidão de uma RNA em produzir saídas apropriadas, para entradas (amostras) que não foram utilizadas durante a fase de treinamento (HAYKIN, 2007). Além disto, RNAs também são capazes de extrair informações (características) que não foram apresentadas de forma explícita, por meio dos exemplos de treinamento (BRAGA et al., 2007).

O aspecto do aprendizado em RNAs é o fator fundamental para que haja êxito na aplicação dessa técnica. Uma RNA tem a capacidade de aprender com exemplos e, ainda, fazer interpolações e extrapolações do que aprendeu. Um algoritmo de aprendizado é definido como um conjunto de procedimentos, com a finalidade de adaptar os parâmetros de uma RNA, para que ela possa aprender uma determinada função. Existem vários algoritmos de aprendizado, cada

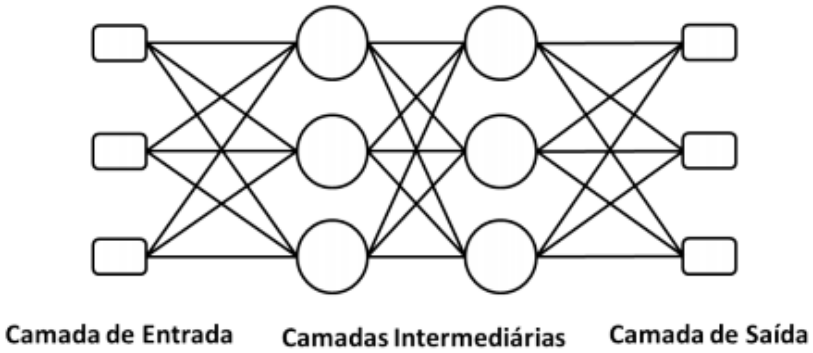
um diferencia-se pelo modo de como o ajuste de pesos é realizado (BRAGA et al., 2007). De acordo com Braga et al. (2007), a etapa de aprendizagem consiste em um processo iterativo de ajuste dos pesos da RNA.

Segundo Bishop et al. (1995), técnicas de IC, como aprendizado de máquina por RNAs, têm apresentado avanços em diversos campos da ciência. Em sua maioria, os resultados originaram-se de aplicações nas quais envolveram o reconhecimento de padrões e que fizeram o uso de diferentes arquiteturas de RNA. Para exemplificar, pode-se citar alguns problemas envolvendo reconhecimento de padrões, onde podemos empregar o uso de RNAs para soluciona-los: reconhecimento de caracteres, reconhecimento de gestos, previsões financeiros, dentre outros (BISHOP et al., 1995).

3.4.2 Redes Neurais de Múltiplas Camadas

Embora com os avanços da época, o modelo do perceptron tinha uma grande limitação quanto a resolução de problemas. De acordo com Braga et al. (2007), RNAs de uma só camada resolvem apenas problemas linearmente separáveis, diferentemente da maioria dos problemas reais que não são linearmente separáveis. Desta forma, o perceptron não é capaz de resolver, por exemplo, o operador XOR. Questão esta que ficou quase duas décadas sem uma solução, num período histórico que ficou conhecido como "Inverno da I.A". Somente em 1986, com o desenvolvimento dos modelos denominados *perceptrons* de múltiplas camadas ou *multilayer perceptron* (MLP), os problemas não-linearmente separáveis puderam ser solucionados (DEEPLEARNINGBOOK, 2021).

Figura 4 – Modelo básico de um MLP



Fonte: Neto (2017)

Uma nomenclatura muito utilizada em RNA é chamada de época. Uma época é definida por um ciclo, onde todos os dados de entrada passam pelas camadas intermediárias e, em seguida, são reajustados (NISSEN et al., 2003). As RNAs do tipo MLP, conforme a Figura 4, são formadas por:

1. **Camada de Entrada**: onde os dados de entrada são apresentados, não ocorrendo nenhum tipo de processamento.

2. **Camadas Intermediárias**: onde ocorrem de fato os processamentos, sendo responsáveis pela precisão dos resultados alcançados pela RNA. Além disto, todo o treinamento é realizado nestas camadas.

3. **Camada de Saída**: onde são apresentados os resultados alcançados pelo treinamento da RNA.

A grande mudança de uma MLP para um *perceptron* está na camada intermediária, onde ao invés de uma única camada, as MLPs possuem duas ou mais camadas com neurônios interconectados. Desse modo, o uso de duas ou mais camadas intermediárias, permite a resolução de problemas linearmente ou não-linearmente separáveis (RESEARCH; DEVELOPMENT; CYBENKO, 1988).

Segundo Hart et al. (2000), existem alguns parâmetros empregados na construção de uma RNA, que influenciam na capacidade do modelo, são eles:

1. **Quantidade de camadas intermediárias**: define a complexidade da fronteira de decisão *(decision boundary)*, ou seja, a linha traçada para separar duas ou mais classes. Este parâmetro está diretamente relacionado a complexidade do problema, que pode ser linear ou não-linear.

2. **Inicialização dos pesos**: define o modo com que os pesos serão inicializados aleatoriamente. Normalmente com valores entre -1 e 1.

3. **Eliminação dos pesos**: técnica que consiste na eliminação de alguns pesos dos neurônios que não influenciam no resultado. Atualmente, a camada de *Dropout* é uma das mais empregadas.

4. **Taxa de aprendizagem**: Define a velocidade de treinamento do modelo, ou seja, o quanto o modelo ajusta os pesos. Apesar de existir alguns valores iniciais recomendados para o treinamento, como por exemplo 0.3, a definição varia de acordo com o problema a ser resolvido. Além disto, valores altos de taxa aprendizado aceleram a duração do treinamento, porém podem dificultar a convergência do modelo.

5. **Taxa de Momentum**: é um fator utilizado para aumentar a velocidade do treinamento do modelo. Este parâmetro fornece uma "prévia" da mudança dos pesos comparados com os pesos atuais. Geralmente a taxa de momentum é definida entre 0,3 a 0,5, mas existem casos em que esta taxa chega até a 0.9 (ATTOH-OKINE, 1999).

6. **Parada do treinamento**: técnica empregada para evitar o *overfitting* (sobreajuste) do modelo, em outras palavras, evitar que o modelo decore os dados de entrada. Consiste

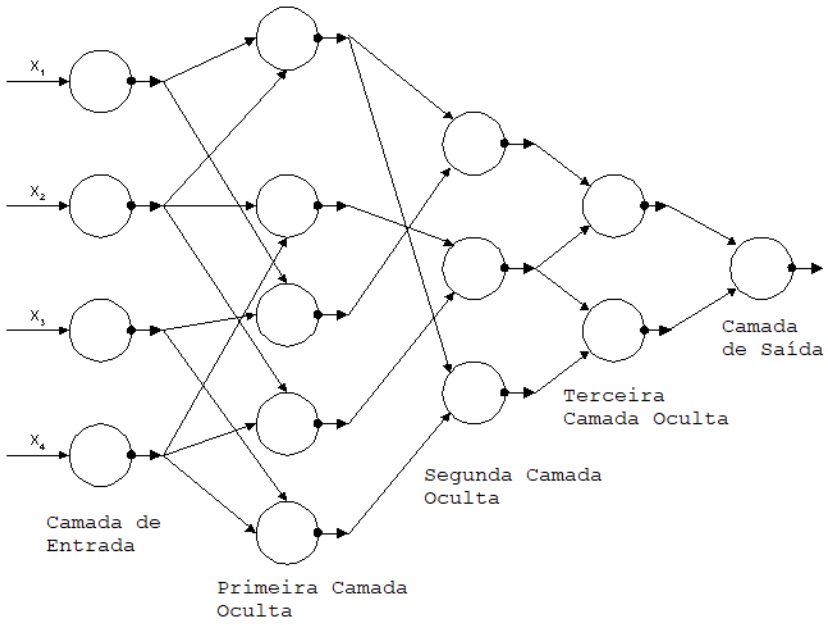
na parada antecipada do treinamento, geralmente quando o modelo para de aprender. A parada do treinamento é mais conhecida como *early-stopping*.

Apesar do conceito das RNAs de entregarem modelos mais complexos de tomada de decisão, a medida que a quantidade de camadas intermediárias aumenta, é sabido que a diferença de complexidade entre o problema a ser resolvido e a complexidade da RNA, causa efeitos inversos. Isto porque, o uso demasiado de camadas intermediárias, exerce uma influência negativa sobre o erro medido, durante a fase de treinamento, nas primeiras camadas (em consequência da propagação do erro), gerando um treinamento menos produtivo ou menos preciso (BRAGA et al., 2007).

3.5 Aprendizado Profundo

Com o advento das MLPs, juntamente com a popularização do *Big Data* e a Computação Paralela, o uso de RNAs disparou. Uma grande consequência disto foi o desenvolvimento de modelos com cada vez mais camadas intermediárias. Dessa maneira surgiu uma sub-área de *Machine Learning*, conhecida como Aprendizado Profundo, ou *Deep Learning* . A Figura 5 ilustra a primeira rede neural profunda, proposta por Alexey Ivakhnenko, considerado o "pai da aprendizagem profunda"(DEEPLEARNINGBOOK, 2021).

Figura 5 – Arquitetura da primeira RNA profunda



Fonte: Modificado de DEEPLEARNINGBOOK (2021)

A aplicação do *Deep Learning* em RNAs, impulsionou diversas áreas da computação, em especial, a Visão Computacional, onde pode-se destacar: reconhecimento de fala, processamento de linguagem natural, reconhecimento de objetos, dentre outros. Dentre os fatores que permitiram

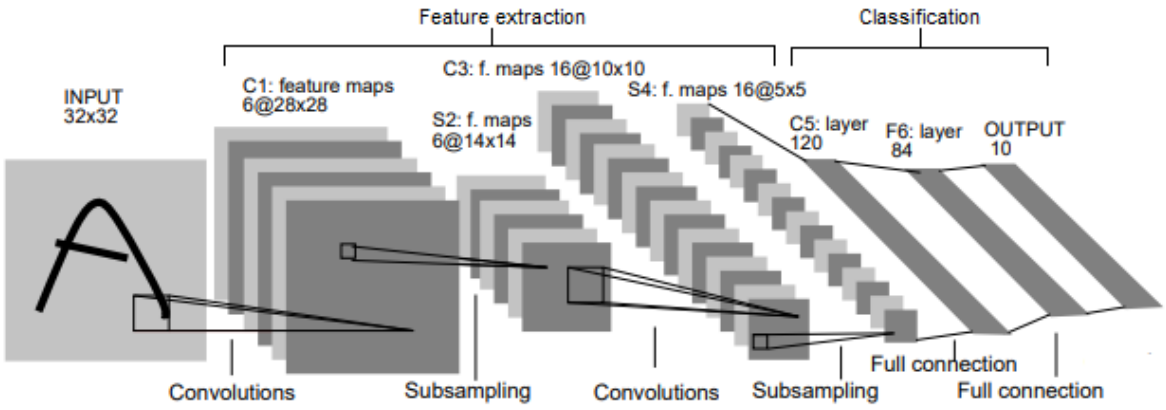
essa difusão, está a automatização da extração de recursos (características) dos dados. Dado que, a chegada do *Deep Learning*, levou a uma demanda no desenvolvimento de métodos que automaticamente extraem características relevantes. Dessa maneira, viabilizou a obtenção de recursos úteis para fins de treinamento, aprendizado e compreensão (BISHOP et al., 1995).

3.6 Redes Neurais Convolucionais

Uma Rede Neural Convolucional (CNN), também conhecida como *ConvNet*, é um tipo de RNA que emprega o *Deep Learning*. Uma estrutura fundamental em qualquer arquitetura de CNN, é a presença de camadas intermediárias de convolução. O primeiro modelo neural a usar *Deep Learning* e camadas convolutivas, foi o *Neocognitron*, proposto por Fukushima (1988). A arquitetura do *Neocognitron* permitiu ao modelo aprender a reconhecer padrões visuais. Apesar das contribuições do *Neocognitron*, uma abordagem prática para a resolução de problemas reais não foi alcançada (DEEPLEARNINGBOOK, 2021).

Somente com o avanço do algoritmo de *backpropagation* - o uso de erros no treinamento de modelos de RNAs profundas - possibilitou a aplicação de uma *ConvNet* em um problema real. Dessa forma, LeCun et al. (1998) apresentou a *LeNet*, uma arquitetura de CNN que foi utilizada para ler dígitos manuscritos de cartas de correspondências. De acordo com LeCun et al. (1998), as CNNs combinam 3 conceitos estruturais essenciais: os campos receptivos locais, o compartilhamento de pesos e a subamostragem (pooling).

Figura 6 – Arquitetura LeNet



Fonte: Modificado de LeCun et al. (1998)

Conforme a Figura 6, pode-se destacar as principais estruturas presentes em uma CNN:

1. **Camadas de convolução (*convolutions*)**: são responsáveis por filtrar todo espaço da imagem, detectando formas e padrões, através de filtros gerados pelo processo de convolução. O resultado desta operação é um mapa de recursos, também chamado de mapa de ativação ou de características. Além disto, cada filtro gerado é chamado de *kernel*, que por sua

vez, são formados por pesos. Para completar, cada região onde o *kernel* é aplicado é denominado, campo receptivo local (LECUN et al., 2010).

2. **Camadas de agrupamento (*subsampling/pooling*)**: são responsáveis por reduzir o tamanho das imagens, ou seja, simplificar as informações contidas na camada de convolução. Dentre os métodos de *pooling* mais utilizados, o *max pooling* é o mais indicado, visto que identifica as características mais importantes (LECUN et al., 2010).

3. **Camada de achatamanento (*flatten*)**: é responsável por achatar os dados em uma matriz uni-dimensional (vetor) (SAMMUT; WEBB, 2011). Processo necessário para a camada posterior.

4. **Camadas totalmente conectadas (*dense layers*)**: são responsáveis pela classificação do modelo. São as clássicas camadas totalmente conectadas, pois as suas entradas, são as saídas das camadas anteriores (SAMMUT; WEBB, 2011).

3.6.1 Algoritmo *backpropagation*

Dentre os algoritmos de aprendizado de RNAs, o *backpropagation* é o mais conhecido e usado. O desenvolvimento deste algoritmo, foi responsável pela retomada do interesse em RNAs e o fim do "inverno da IA". O *backpropagation* realiza o treinamendo da rede de forma supervisionada, em outras palavras, compara a saída obtida pela rede com a saída desejada, logo depois, faz o ajuste dos pesos. O treinamento deste algoritmo pode ser divido em duas fases: *forward* e *backward* (BRAGA et al., 2007).

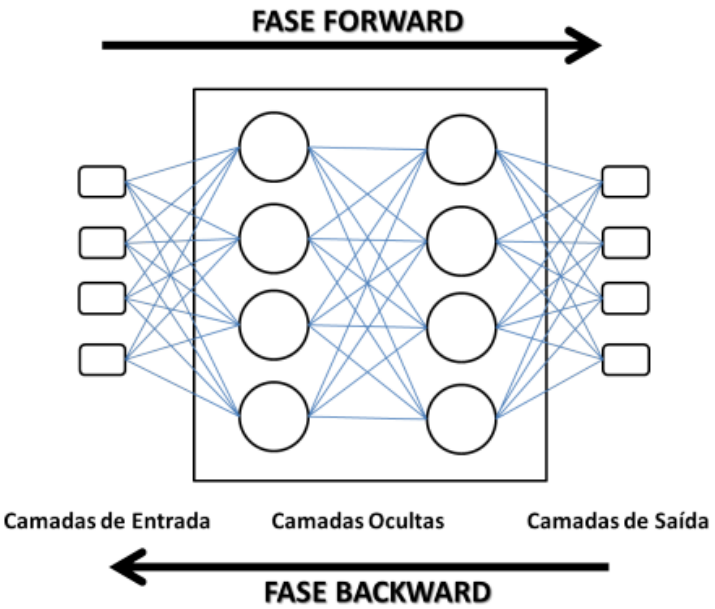
Isto posto, a primeira fase (*forward*) tem como objetivo processar os dados de entrada e fornecer uma saída para a RNA. E então, a segunda fase (*backward*) fica responsável por comparar os resultados alcançados, com o desejável, e logo em seguida, retornar ao início da rede para ajustar os pesos das entradas do neurônios (BRAGA et al., 2007). A Figura 7 apresenta uma visão do funcionamento do *backpropagation*.

3.7 Florestas aleatórias

Uma Floresta Aleatória é definida como um metaestimador, que ajusta vários classificadores de Árvores de Decisão, em várias subamostras do conjunto de dados a fim de melhorar a acurácia (PEDREGOSA et al., 2011). Para que seja possível a compreensão desta técnica, primeiramente faz-se necessário o estudo sobre Árvores de Decisão.

Segundo Oshiro (2013), as Árvores de Decisão empregam o conceito de dividir para conquistar, por consequência conseguem fragmentar um problema complexo, em vários problemas simples, de forma recursiva. O processo de construção de uma Árvore de Decisão é realizado com a seleção de um atributo aleatório que será um divisor do conjunto de dados.

Figura 7 – Fases do algoritmo *backpropagation*



Fonte: Neto (2017)

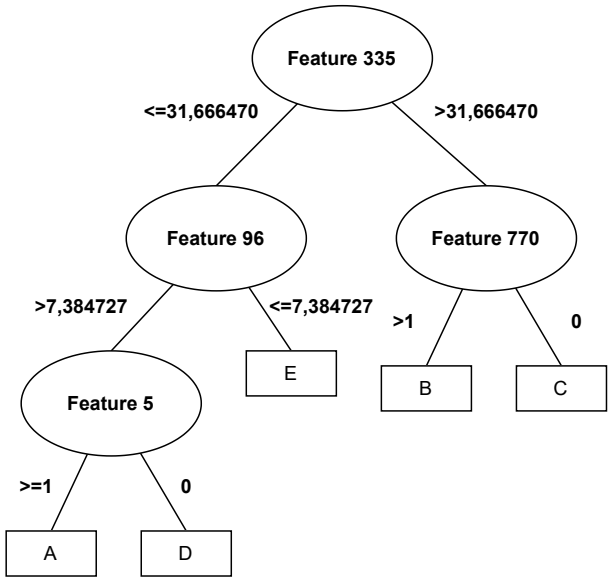
A Figura 8, ilustra a estrutura básica de uma Árvore de decisão, para a tarefa de classificação de imagens de gestos do alfabeto manual. Na Figura em questão, temos 5 classes (letras) para classificar, onde as características (*features*) são valores de pixels normalizados. Na raiz (topo) da árvore, localiza-se um atributo escolhido aleatoriamente de acordo com o critério definido, neste caso o "*feature* 335". Seguindo a estrutura lógica, se o valor de "*feature* 335» 31,666470, então o algoritmo segue para o lado direito da Árvore, chegando ao "*feature* 770"onde é realizada outra comparação até chegar em um nó folha com uma classe associada.

Uma Árvore de Decisão pode ser utilizada para classificar exemplos desconhecidos, além daqueles utilizados na fase de treinamento. Desse modo, ao receber uma nova amostra, percorre-se a Árvore de Decisão, partindo-se da raiz e desvia-se, em cada nó de decisão, até chegar em um nó folha, com a classe correspondente (OSHIRO, 2013).

A técnica de Florestas Aleatórias emprega diversas Árvores de Decisão para constituir a sua “floresta” de classificação. Conforme Lucas (2011), as Árvores de Decisão são excelentes preditores, entretanto nem sempre conseguem uma boa capacidade de generalização. Em contrapartida, as Florestas Aleatórias apresentam excelentes estatísticas de acurácia.

De acordo com Breiman (2001) e Boldt (2014), a técnica de Florestas Aleatórias é um algoritmo de particionamento recursivo que mescla predições feitas por um conjunto de Árvores de Decisão. Este algoritmo utiliza o mesmo método das Árvores de Decisão, onde os dados são divididos, recursivamente, em nós de classificação. Em outras palavras, o algoritmo de Florestas Aleatórias consiste em um aglomerado de Ávores de Decisão, onde cada Árvore computa suas classificações, e ao final, ocorre uma votação, onde as classificações mais votadas são escolhidas.

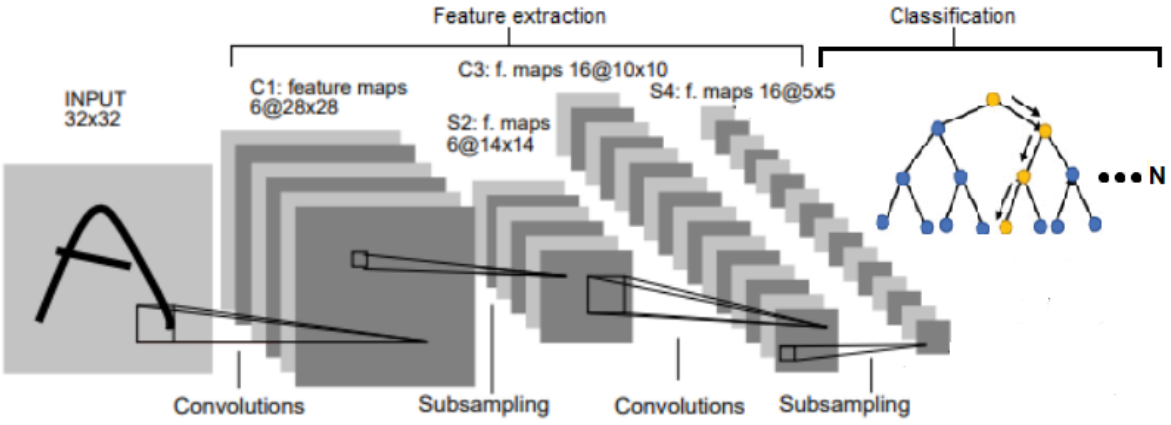
Figura 8 – Árvore de Decisão



Fonte: Do Autor (2021)

Para utilizar as Florestas Aleatórias com o Deep Learning, faz-se necessária uma abordagem alternativa, combinando as técnicas de Redes Neurais Convolucionais e Florestas Aleatórias. Como proposto em Sun et al. (2020), ao combinar as duas técnicas obtem-se a capacidade de extrair *features* através das CNNs e classificar amostras por meio das Florestas Aleatórias. Dessa forma, tem-se basicamente uma estrutura composta por 4 camadas: camada de convolução, camada de *pooling*, camada totalmente conectada e uma camada de Floresta Aleatória (SUN et al., 2020). A Figura 9 apresenta a estrutura do algoritmo CNN-RF.

Figura 9 – Floresta Aleatória com Rede Neural Convolucional



Fonte: Modificado de Sun et al. (2020)

3.8 Métodos de avaliação

Para avaliar os classificadores treinados foram utilizadas matrizes de confusão e métricas de avaliação de desempenho. No campo do Aprendizado de Máquina, uma matriz de confusão é uma tabela que permite a visualização do desempenho de um algoritmo de classificação (HART et al., 2000). As métricas de avaliação são utilizadas para medir a capacidade de generalização de um classificador treinado (HOSSIN; SULAIMAN, 2015). Os métodos utilizados são detalhados nas próximas secões.

3.8.1 Matriz de confusão

A matriz de confusão é uma tabela que apresenta uma visão geral do desempenho de um algoritmo classificador. Segundo Monard e Baranauskas (2003), a matriz mostra uma hipótese h do número de classificações corretas, em contraste com as predições do modelo, para cada classe do conjunto de dados T. Sendo assim, os resultados são totalizados em dois âmbitos: classes verdadeiras e classes preditas, para k classes diferentes $\{C1, C2, ..., Ck\}$.

Portanto, cada elemento de entrada $M(C_i, C_j)$ da matriz, $i, j = 1, 2, ..., k$, representa o número de amostras de T que realmente pertencem à classe C_i, mas que foram classificadas pela hipótese h como sendo da classe C_j. Esses valores podem ser obtidos através do cálculo:

$$M(C_i, C_j) = \sum \parallel h(x) = C_j \parallel \{\forall (x, y) \in T : y = C_i\}$$

Tabela 1 – Matriz de confusão de um classificador multiclasse

Classe	Predita C1	Predita C2	. . .	Predita Ck
Verdadeira C1	M (C1, C1)	M (C1, C2)	. . .	M (C1, Ck)
Verdadeira C2	M (C2, C1)	M (C2, C2)	. . .	M (C2, Ck)
. . .	. . .	. . .	. . .	. . .
Verdadeira Ck	M (Ck, C1)	M (Ck, C2)	. . .	M (Ck, Ck)

Fonte: Modificado de Monard e Baranauskas (2003)

Ademais, também pode-se destacar que em uma matriz de confusão, os elementos da diagonal principal $M(C_i, C_j)$ representam o número de acertos do modelo. Além disso, os demais elementos $M(C_i, C_j)$ para $i \neq j$ indicam as classificações erradas do modelo. Desta forma, uma matriz de confusão de um classificador ideal, é aquela onde todos os elementos, com exceção da diagonal principal, são iguais a zero, indicando a ausência de erros.

Em geral, para simplificar a compreensão, os elementos da matriz recebem um nome, tanto para facilitar os cálculos como para visualizar o significado de cada um. Logo, em uma matriz de confusão de um classificador binário, apresenta-se o seguinte:

1. **Verdadeiros Positivos (VP):** indica amostras cujo um rótulo real A, é previsto como A, ou seja, acertos de classificação.
2. **Falsos Positivos (FP):** indica amostras cujo um rótulo real B, é previsto como A, ou seja, erros de classificação.
3. **Falsos Negativos (FN):** indica amostras cujo um rótulo real A, é previsto como B, ou seja, erros de classificação.
4. **Verdadeiros Negativos (VN):** indica amostras cujo um rótulo real B, é previsto como B, ou seja, acertos de classificação.

A partir dos elementos VP, VN, FP e FN apresentados, a matriz de confusão de um classificador binário pode ser construída. A Tabela 2 demonstra uma matriz de confusão de ordem 3 para a classificação de 3 letras do alfabeto manual. Ou seja, a ordem da matriz é proporcional a quantidade de classes do conjunto de dados. Entretanto, em uma abordagem multiclasse, o elemento verdadeiro negativo (VN) não existe, dessa forma, a diagonal principal representa os VPs de cada classe (SAMMUT; WEBB, 2011).

Tabela 2 – Matriz de confusão de um classificador para 3 classes do alfabeto manual

Classificador		**Classe Predita**		
		A	B	C
Classe Real	A	VP	FN	FN
	B	FP	VP	FN
	C	FP	FP	VP

Fonte: Do Autor (2021)

3.8.2 Métricas

As métricas são medidas de desempenho usadas para avaliar a qualidade dos classificadores. Existem diversas métricas que possuem diferentes finalidades, e a importância de cada uma varia de acordo com cada problema (SAMMUT; WEBB, 2011). As métricas utilizadas neste trabalho são detalhadas nos itens a seguir:

- **Acurácia (*accuracy*):** A acurácia é uma medida que se refere ao grau em que as previsões de um modelo correspondem à realidade modelada (SAMMUT; WEBB, 2011). Em outras palavras, indica a proporção de previsões corretas sobre o número total de amostras avaliadas. Para problemas de classificação a acurácia é a métrica mais utilizada (BROWNLEE, 2017).

$$Accuracy = \frac{VP}{VP + FP + FN}$$

- **Erro (*error*):** O erro é uma medida que se refere a proporção de previsões incorretas sobre o número total de amostras avaliadas (HOSSIN; SULAIMAN, 2015).

$$Error = \frac{FP + FN}{VP + FP + FN}$$

- **Precisão (*precision*)**: A precisão, também chamada de valor preditivo positivo (PPV), é uma métrica mais específica que a acurácia, e permite uma análise por classe. Segundo Raschka e Mirjalili (2017), a precisão para uma classe A, é o número de amostras classificadas corretamente para a classe A, dividido pelo total de amostras classificadas como da classe A.

$$Precision = \frac{VP}{VP + FP}$$

- **Revogação (*recall*)**: A revogação, também chamada de sensibilidade, assim como a precisão, é uma métrica para análise por classe. Segundo Raschka e Mirjalili (2017), a revogação para uma classe A, é o número de amostras classificados corretamente para classe A, dividido pelo número de amostras da classe A.

$$Recall = \frac{VP}{VP + FN}$$

- **F1-Score (Medida F):** O F1-Score é uma medida que representa a média harmônica entre os valores de *precision* e *recall* (HOSSIN; SULAIMAN, 2015). Os valores de *F1-Score* variam entre 0 e 1, onde valores altos significam um alto desempenho de classificação (THARWAT, 2020).

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Para calcular os valores de precision, recall e f1-score para cada classe, para as redes neurais foi utilizada a função classification_report da biblioteca Scikit-Learn (PEDREGOSA et al., 2011). Enquanto, para as floretas aleatórias foi utilizada uma função de mesmo nome, da biblioteca Yellowbrick (BENGFORT; BILBRO, 2019).

4 METODOLOGIA

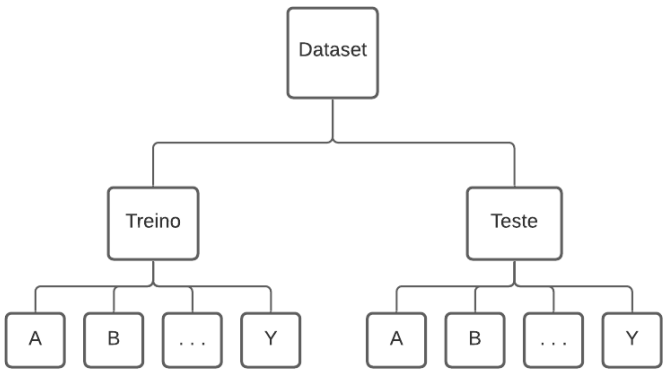
Esta seção descreve a metodologia adotada para o desenvolvimento deste trabalho, abordando as técnicas de aprendizado de máquina que foram utilizadas e os procedimentos executados para o reconhecimento e análise comparativa entre as técnicas empregadas. A seção foi decomposta, a fim de apresentar cada processo do desenvolvimento.

4.1 Base de dados

Como mencionado na seção 3.3.2 do referêncial teórico, as técnicas de aprendizado de máquina necessitam de uma fonte de experiência, na qual se dividem em duas categorias, de acordo com sua natureza de aprendizado. Em função do tema abordado neste trabalho, o reconhecimento de gestos por imagens estáticas ou dinâmicas (vídeos), envolve o uso de aprendizado supervisionado. Sendo assim, utilizou-se uma fonte de experiênca com os dados rotulados, que geralmente são encontrados em bases de dados (*datasets*).

Portanto, utilizou-se uma base de dados de imagens de gestos estáticos do alfabeto manual de Libras, disponível em um repositório público da plataforma GitHub [1]. A base de dados possui um total de 46.262 amostras de imagens, e se encontra dividido em pastas. E, ainda em relação a sua estrutura, o mesmo está dividido em duas pastas, treino e teste, na qual possuem respectivamente, 34.714 e 11.548 amostras. Devido ao seu formato, os rótulos das imagens são apresentados em sub-pastas, indicando o rótulo de cada letra do alfabeto. A Figura 10 apresenta um diagrama, que exemplifica melhor a estrutura da base de dados.

Figura 10 – Diagrama da estrutura de diretórios da base de dados



Fonte: Do Autor (2021)

Ademais, também pode-se destacar que as imagens estão no formato RGB (*Red, Green, Blue*), ou seja, são coloridas. A base de dados selecionada está parcialmente pré-processada,

[1] https://github.com/lucaaslb/cnn-libras/tree/master/dataset

facilitando a manipulação do mesmo. Para finalizar, uma observação importante quanto às letras do alfabeto presentes na base de dados, é a ausência das letras que precisam de gestos dinâmicos. Logo, a base de dados possui 21 classes, ao invés das 26. Isto ocorre em função da base de dados ser projetada para o reconhecimento de gestos estáticos.

Dessarte dos detalhes apresentados, pode-se exemplificar as letras presentes e ausentes. Sendo assim, estão presentes 21 letras (classes) que são elas: A, B, C, D, E, F, G, I, L, M, N, O, P, Q, R, S, T, U, V, W e Y. E dentre as ausentes, estão: H, J, K, X, e Z. A Figura 11, apresenta um pequeno recorte das imagens da base de dados usada.

Figura 11 – Seleção de imagens da base de dados

Fonte: Do Autor (2021)

4.1.1 Balanceamento da Base de dados

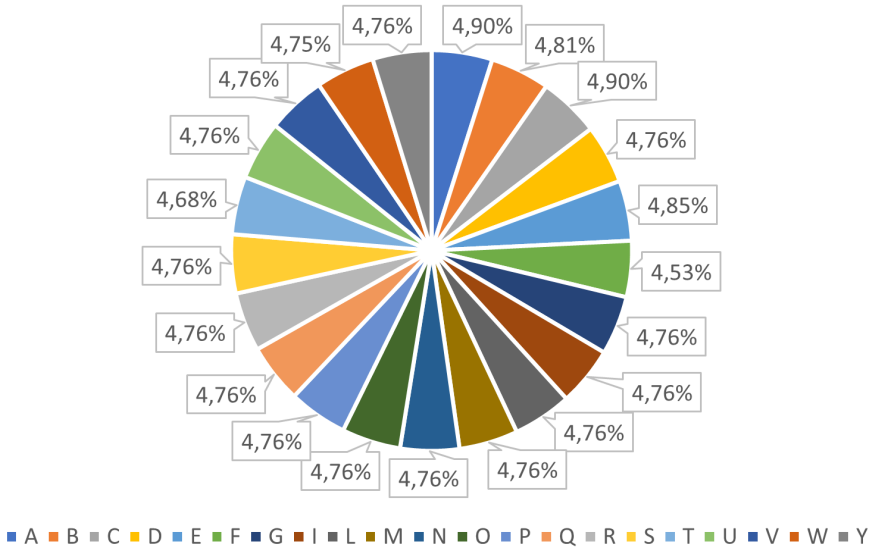
Um aspecto muito importante em aprendizado de máquina é o balanceamento de classes em uma base de dados. Isso por que os algoritmos de classificação tendem a maximizar a medida de desempenho de acurácia (*accuracy*). Logo, se a base de dados estiver fortemente desbalanceada, obtem-se resultados inconsistentes, em função das classes majoritárias e minoritárias. Uma das formas de verificar a disposição de classes em uma base de dados é através da fórmula de distribuição de classes.

Dessa maneira, dado um conjunto T, com n exemplos, para cada classe C_j em T, sua distribuição $distr(C_j)$ é calculada através do número de amostras de T, que possuem classe em C_j, dividido pelo número total de amostras de T. Isto posto, temos a proporção de amostras para cada classe, dada por:

$$distr(C_j) = \frac{1}{n} \sum_{i=1}^{n} \parallel y_i = C_j \parallel$$

Em vista do cálculo da distribuição de classes $distr(C_j)$ apresentada, foi possível calcular a proporção das amostras para cada classe da base de dados. Portanto, o gráfico da Figura 12, demonstra a distribuição das amostras. Nota-se, uma equivalência consistente e uniforme das amostras para cada classe.

Figura 12 – Proporção de amostras para cada classe

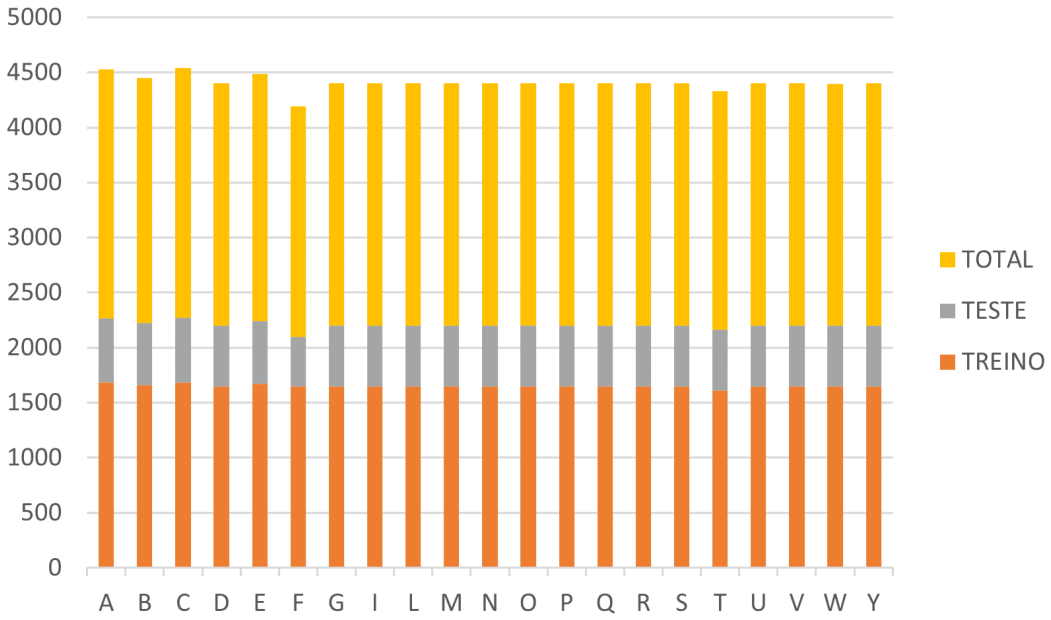


Fonte: Do Autor (2021)

Além disso, também foi calculada a distribuição das amostras em relação à base de dados como um todo. Dessa maneira, podemos verificar a proporção das amostras de cada classe encontradas nos diretórios de treino e teste. Logo, o gráfico da Figura 13 ilustra esta proporção. Podemos observar, assim como na distribuição de amostras de cada classe, uma justa conformidade entre as partes.

Deste modo, de acordo com as constatações observadas através dos gráficos, podemos afirmar que, a base de dados satisfaz as condições de equivalência de classes. Sendo assim, não foi necessário aplicar métodos de balanceamento.

Figura 13 – Proporção da base de dados



Fonte: Do Autor (2021)

4.2 Recursos utilizados

Para os treinamentos e testes das técnicas, utilizou-se um *notebook* com processador *intel core* i5-7200U com 2,71GHz e placa gráfica integrada *intel graphics* 620, equipado com 8,00GB de memória RAM e *Windows 10*. Além disso, para agilizar o ajuste dos parâmetros e pequenos testes, também foi utilizado a plataforma *Google Colaboratory*. O *Google Colaboratory* é um ambiente interativo que permite a execução de códigos *Python*[2] em nuvem.

Ademais, outro recurso importante empregado foi a criação de um ambiente virtual de desenvolvimento, realizado através do gerenciador de pacotes e sistemas, Conda[3]. Dessa forma, foi possível gerenciar as dependências e também evitou-se conflitos entre pacotes e versões existentes na máquina, já que o projeto fica isolado, em uma espécie de *container*.

4.3 Pré-processamento dos dados

A etapa de pré-processamento do conjunto de dados, é um dos principais processos no aprendizado de máquina, visto que ela é indispensável para aprimorar a eficácia do treinamento das técnicas de aprendizado. O uso de Redes Neurais Profundas e Florestas Aleatórias combinadas com filtros de convolução para a extração de recursos, resultam em uma abordagem moderna de aprendizado de máquina. Dessa forma, procedimentos como redução de ruído, aumento

[2] *Python* é uma linguagem de programação de alto nível.
[3] https://conda.io/projects/conda/en/latest/index.html

de contraste, tornam-se desnecessários, devido ao fato de não precisarmos gerar os recursos (*features*) de forma manual.

Ainda assim, práticas foram empregadas para garantir que as imagens estejam nas condições ideais para serem usadas em uma rede neural. Tal como, o redimensionamento da escala dos pixels das imagens, de um intervalo entre 0 a 255 para 0 a 1. A razão desse procedimento ser tão importante em redes neurais, é devido ao fato de que, trabalhar com uma escala de valores mais próximas, facilita os cálculos e ajustes dos pesos, logo, é mais viável computacionalmente. Além disso, também foi definido uma dimensão de largura e altura para as imagens, a fim de garantir que todas as imagens tenham tamanhos iguais.

4.4 Treino, teste e validação

O processo de separação dos dados em conjuntos de treino e teste, também é um fator essencial. Isso por que se os dados de treino fossem usados para teste, não seria possível analisar a capacidade de generalização em ambientes reais. Ao utilizar os mesmos dados para treino e teste, não seria possível saber como o modelo se comportaria com dados novos. Inclusive, dividir o conjunto de treino em um subconjunto de validação ou mesmo utilizar o conjunto de teste para validar o treinamento, é uma boa prática e ajuda a ajustar o modelo durante a fase de treinamento das redes neurais.

4.5 Rede Neural Convolucional

Consoante com o que foi apresentado no referencial teórico, pode-se notar o crescente uso das redes neurais convolucionais em aplicações de reconhecimento de padrões por imagens e vídeos. A fim de possibilitar a implementação das redes neurais convolucionais, foram utilizadas as bibliotecas, TensorFlow[4] e Keras[5]. O TensorFlow é uma biblioteca para aprendizado de máquina, e o Keras atua como uma interface voltada para a construção de redes neurais profundas.

4.5.1 Arquitetura

A arquitetura da rede neural convolucional, foi definida pelo método de tentativa e erro, tendo como base a arquitetura LeNet, proposta por LeCun, conforme a Figura 6. Dessa forma, analisa-se os resultados de treino e teste, e ajusta-se os parâmetros da rede até que atenda as necessidades de classificação. Uma vez encontrado uma combinação ideal, entre os parâmetros e os resultados, repete-se 10 vezes todo o processo de treino e teste, a fim de encontrar uma média de acertos confiável para esta técnica. Desse modo, pode-se descartar possíveis conexões entre a aleatoriedade dos pesos e o desempenho do modelo.

[4] https://www.tensorflow.org/
[5] https://keras.io/

4.5.2 Cuidados com *Overfitting* e *Underfitting*

Entre os fatores que são evitados durante o treinamento do modelo, estão o *overfitting* e *underfitting*. Existem diversas práticas adotadas a fim de conter a presença deles. A primeira prática adotada foi utilizar conjuntos de treino, validação e teste, a fim de poder monitorar e comparar o desempenho de ambos. Além disso, também foi usado uma função de retorno de chamada, conhecida como *callback*. A *callback* empregada chama-se *early stopping*, ou parada antecipada. É muito utilizada para evitar *overfitting* em modelos de RNAs.

Para complementar, outra técnica amplamente usada em RNAs para ajudar o modelo a generalizar é o *Dropout*. O *Dropout* é uma camada geralmente implementada entre as camadas densas, que possuem parâmetros treináveis, ou seja, pesos. Basicamente ela desativa alguns neurônios (nós) aleatoriamente, durante a fase de treinamento, em função de uma porcentagem definida. Dessa maneira, consegue-se simular o treinamento de diferentes RNAs.

4.5.3 Bibliotecas e Parâmetros

Para a implementação da rede neural convolucional, foi utilizado o módulo do *Keras*, *Sequential Model*. Este módulo permite a construção de camadas em sequência, onde cada camada possui um tensor de entrada e um tensor de saída. Além disso, o Keras possui uma função denominada *flow from directory*, muito útil para aplicar em bases de dados que encontram-se em formato de pastas, como no presente trabalho. Este possui parâmetros para rotular classes conforme as sub-pastas, assim como embaralhamento dos dados.

Conforme descrito na subseção 4.5.1, os parâmetros ideais foram encontrados após uma série de experimentos. Logo, os parâmetros utilizados foram:

1. Taxa de aprendizagem: 0,0001
2. Função de ativação das camadas ocultas: Relu
3. Função de ativação da camada de saída: Softmax
4. Otimizador: Adam
5. Quantidade de épocas: 100
6. Profundidade da rede: 8
7. Quantidade de filtros: 8, 16, 32
8. Tamanho de *kernel*: 3x3
9. Porção de treinamento: 70%
10. Porção de validação: 10%
11. Porção de teste: 20%

4.6 Rede Neural Densa

Assim como na implementação das redes neurais convolucionais, as bibliotecas *TensorFlow* e *Keras* foram usadas para aplicação das redes neurais densas. A principal diferença entre as técnicas é a ausência dos filtros de convolução, juntamente com a camada de *Pooling*. Dessa forma, existem mais parâmetros treináveis, já que cada *pixel* recebe um peso.

4.6.1 Arquitetura

A arquitetura da rede neural densa, foi determinada do mesmo modo que a rede neural convolucional, por tentativa e erro. Começando com uma camada oculta, e testando diferentes quantidades de neurônios na mesma. Seguindo nessa linha, aumentou-se gradativamente o número de camadas e neurônios. Uma vez encontrada a combinação entre os parâmetros e os resultados, támbém foi executado 10 vezes o processo de treino e teste, afim de coletar resultados confiáveis.

4.6.2 Cuidados com *Overfitting* e *Underfitting*

Além da boa prática já mencionada de trabalhar com conjuntos de treino, validação e teste, também foi usada a *callback*, *early stopping*, para evitar a ocorrência de *overfitting*. O diferencial em relação ao empregado na rede neural convolucional , é que foi usado um valor de tolerância mais baixo. Isso porque as redes neurais densas, tendem a memorizar os dados quando implementadas com várias camadas.

4.6.3 Bibliotecas e Parâmetros

Conforme descrito na subseção 4.6.1, os parâmetros ideais foram encontrados após uma série de experimentos. Logo, os parâmetros utilizados foram:

1. Taxa de aprendizagem: 0,001
2. Taxa de *momentum*: 0,1
3. Função de ativação das camadas ocultas: Relu
4. Função de ativação da camada de saída: Softmax
5. Otimizador: SGD
6. Quantidade de épocas: 100
7. Profundidade da rede: 3
8. Porção de treinamento: 70%
9. Porção de validação: 10%

10. Porção de teste e validação: 20%

4.7 Floresta Aleatória

Com o intuito de implementar a técnica de floresta aleatória através de uma abordagem de aprendizado de máquina moderna, foram utilizadas as bibliotecas *TensorFlow*, *Keras* e *Scikit-Learn*. O *TensorFlow* juntamente com o *Keras* foram empregados para a construção do extrator de recursos, que por sua vez, é baseado nas camadas de convolução e *pooling*. O *Scikit-Learn* é uma biblioteca de aprendizado de máquina de código aberto, que possui uma infinidade de algoritmos de classificação, regressão e etc.

4.7.1 Arquitetura

Seguindo na mesma linha das outras técnicas, os parâmetros do modelo da floresta aleatória foram definidos por tentativa e erro. O primeiro parâmetro ajustado foi o número de árvores na floresta. O primeiro teste foi realizado com 50 árvores, onde observou-se uma precisão entre 95 a 97% de acerto. Apesar disso, para verificar que com uma quantidade menor de árvores o modelo teria um desempenho pior, foram explorados valores com: 10, 20, 30 e 40 árvores. Consequentemente, notou-se uma grande perda de precisão no conjunto de teste.

Prosseguindo com os experimentos, iniciou-se a exploração de valores mais altos gradativamente. E, para este modelo em questão, foi constatado que, a partir de 50 árvores, a precisão parou de melhorar de maneira significativa. Logo após isso, foi dada enfâse na profundidade da árvore, que é definida pelo parâmetro *max_depth*, o qual por padrão faz com que cada árvore cresça até que sua folha seja pura. Dessa forma, foram experimentados valores entre 3 e 10, e observou-se que o desempenho foi superior utilizando uma profundidade máxima igual a 9.

Ainda, foram utilizadas sementes aleatórias, com o objetivo de garantir que a aleatoriedade das amostras de treino não influenciasse no desempenho do conjunto de teste. Posto isto, também foi executado 10 vezes o processo de treino e teste, a fim de coletar resultados confiáveis.

4.7.2 Cuidados com *Overfitting* e *Underfitting*

Uma das técnicas mais utilizadas para evitar *overfitting* em modelos de florestas aleatórias, é a aplicação de *pruning* (poda). O pruning pode ser aplicado através da definição de três parâmetros no momento da criação do modelo, são eles:

1. *n_estimators*: este parâmetro determina a quantidade de árvores na floresta.
2. *max_depth*: este parâmetro determina a profundidade máxima que as árvores vão se extender. Geralmente os valores recomendados para pruning são entre 3 a 7.

3. *max_features*: este parâmetro determina a quantidade máxima de recursos. É importante ressaltar que, o processo de ensacamento das florestas aleatórias, mitiga qualquer influência desse parâmetro em questões de overfitting.

4.7.3 Bibliotecas e Parâmetros

Conforme descrito na subseção 4.6.1, os parâmetros ideais foram encontrados após uma série de experimentos. Logo, os parâmetros utilizados foram:

1. *n_estimators*: 50
2. *seed*: aleatório entre 1 a 1000
3. *max_depth*: 9
4. *max_features*: 'auto', sem restrição
5. Quantidade de filtros: 8, 16, 32
6. Tamanho de *kernel*: 3x3
7. Porção de treino: 80%
8. Porção de teste: 20%

4.8 Integração com a *Webcam*

Para testar os modelos treinados em ambiente real, foi desenvolvida uma aplicação para classificar os gestos em tempo real. A aplicação foi escrita em código Python, utilizando as bibliotecas *OpenCV, Numpy e Keras*. A lógica consiste em carregar o modelo treinado através do módulo *load_model* do *Keras* e aplicar a classificação em uma região específica do vídeo, ou seja, de acordo com as dimensões das imagens utilizadas no treinamento dos modelos.

Para isso foi empregue a função *crop*, que tem como objetivo realizar cortes através de 4 coordenadas: *top, left, bottom e right*. Dessa forma, em um vídeo que basicamente é composto por uma sequência de frames, pode-se delimitar uma região de interesse. O uso da webcam nativa, no caso de um notebook, pode ser utilizada através do módulo *VideoCapture* do *OpenCV*. Sendo assim, com uma função para carregar o crop salvo, com o mesmo pré-processamento utilizado nas imagens da base de dados, é possível utilizar o módulo *predict* do *Keras* e capturar o resultado, e posteriomente formata-lo para exibir no vídeo.

5 RESULTADOS OBTIDOS

Apoiado na metodologia apresentada neste trabalho, utilizando a base de dados em evidência e aplicando as técnicas indicadas, foi possível obter diversos resultados. Esta seção foi dividida a fim de apresentar, de forma detalhada, os resultados de cada técnica de Inteligência Computacional abordada, além do comparativo final.

5.1 Redes Neurais Convolucionais

Depois de realizar o pré-processamento dos dados, a construção do modelo e treinamento da rede neural convolucional, foi notável a eficácia da técnica para o reconhecimento de imagens. Os erros e acurácias de validação foram obtidos através da função *fit* do Keras, onde foi inserido o conjunto de validação no parâmetro *validation_data* e a quantidade de validações por época com o parâmetro *validation_steps*.

Segundo os resultados apresentados na Tabela 3, pode-se notar a consistência do modelo construído, tanto para a alta taxa de acerto, que se manteve estável em ambos conjuntos, quanto para o desvio padrão, que se aproximou de zero, indicando a uniformidade dos resultados obtidos.

Tabela 3 – Percentual de acertos utilizando a CNN

Treinamento	**Acurácia do treino**	**Acurácia do teste**	**Duração do treinamento**
1	99,59%	98,81%	60,9 min
2	98,99%	98,19%	82,1 min
3	99,65%	98,97%	57,5 min
4	99,56%	98,91%	87,6 min
5	99,93%	99,65%	67,9 min
6	95,91%	95,01%	63,7 min
7	99,12%	98,22%	62,9 min
8	98,47%	98,22%	47,3 min
9	99,81%	99,56%	77,2 min
10	99,71%	99,48%	64,7 min
Média	**99,07%**	**98,50%**	-
Desvio Padrão	**0,0114**	**0,0128**	-

Fonte: Do Autor (2021)

Logo após, é demonstrado na Tabela 4, o relatório de classificação de cada classe no conjunto de teste no primeiro treinamento. Concisamente percebe-se o alto desempenho do modelo, na qual, das 21 classes presentes, acertou todas as amostras destas, em 8 classes. Além disto, manteve um alto grau de desempenho em todas as classes, visto que a média das métricas ficou muito próximo do valor máximo, 1.

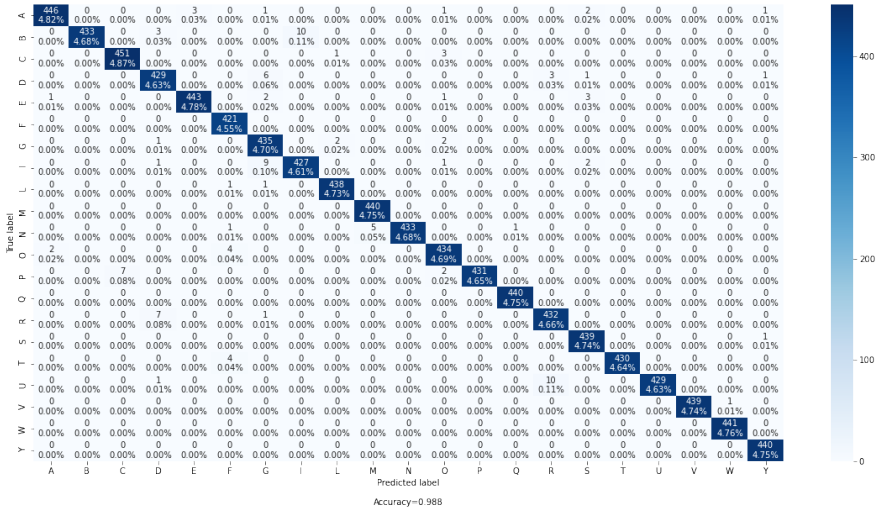
Tabela 4 – Relatório de classificação (primeiro treinamento) do conjunto de teste utilizando a CNN

Classes	**Precision**	**Recall**	**F1-Score**	**Support**
A	0,99	0,98	0,99	454
B	1,00	0,97	0,99	446
C	0,98	0,99	0,99	455
D	0,97	0,97	0,97	440
E	0,99	0,98	0,99	450
F	0,98	1,00	0,99	421
G	0,96	0,99	0,97	440
I	0,98	0,97	0,97	440
L	0,99	1,00	0,99	440
M	0,99	1,00	0,99	440
N	1,00	0,98	0,99	440
O	0,98	0,99	0,98	440
P	1,00	0,98	0,99	440
Q	1,00	1,00	1,00	440
R	0,97	0,98	0,98	440
S	0,98	1,00	0,99	440
T	1,00	0,99	1,00	434
U	1,00	0,97	0,99	440
V	1,00	1,00	1,00	440
W	1,00	1,00	1,00	441
Y	0,99	1,00	1,00	440
Média	0,988	0,988	0,989	9261

Fonte: Do Autor (2021)

A Figura 14, apresenta a matriz de confusão do conjunto de teste no primeiro treinamento, e pode-se observar uma ocorrência mínima de falsos positivos e falsos negativos. Dessa forma, tem-se uma diagonal principal contendo grande parte das classificações corretas.

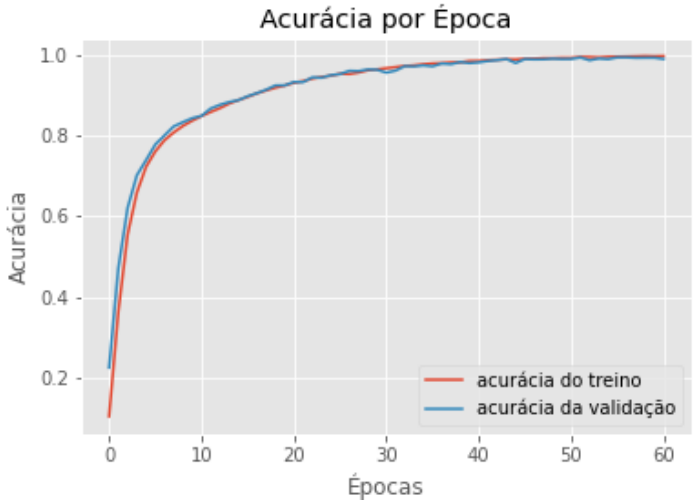
Figura 14 – Matriz de confusão do conjunto de teste (primeiro treinamento) utilizando a CNN



Fonte: Do Autor (2021)

Além disso, as Figuras 15 e 16, apresentam respectivamente a acurácia e o erro do modelo ao longo das épocas. Na figura 15, pode-se notar a curva de aprendizado do modelo, onde a linha vermelha indica a acurácia do modelo no conjunto de treino, ao passo em que a linha azul indica a acurácia do modelo no conjunto de validação.

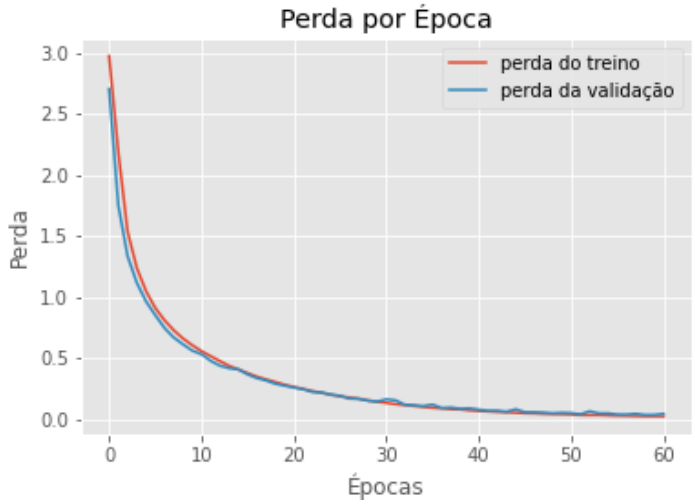
Figura 15 – Gráfico da acurácia por época (primeiro treinamento) utilizando a CNN



Fonte: Do Autor (2021)

Ainda na Figura 15, é possível notar que o treinamento foi encerrado antes das 100 épocas devido a função de parada antecipada, que determina a finalização do treinamento caso o modelo pare de melhorar, evitando o sobreajuste (*overfitting*). Já na Figura 16, temos o erro do modelo ao longo das épocas, e na última época, pode-se notar um valor mínimo, muito próximo de zero, indicando uma alta consistência.

Figura 16 – Gráfico do erro por época (primeiro treinamento) utilizando a CNN



Fonte: Do Autor (2021)

5.2 Redes Neurais Densas

As redes neurais densas seguem uma implementação semelhante das redes neurais convolucionais, com exceção da ausência do mecanismo de extração de recursos, caracterizado pelas camadas de convolução e *pooling* [1]. Assim como na técnica da CNN, os erros e acurácias de validação da DNN, foram obtidos por meio da função *fit* do Keras, utilizando os parâmetros *validation_data* e *validation_steps*.

De acordo com os resultados apresentados na Tabela 5, pode-se observar que o modelo teve um desempenho regular em ambos os conjuntos. Nota-se uma variação mínima da acurácia, colaborando para um desvio padrão baixo. Apesar dos resultados regulares, testes com diferentes taxas de aprendizado foram realizados, porém não foi constatado melhorias significativas nos resultados. Sendo assim, manteve-se os parâmetros selecionados.

Tabela 5 – Percentual de acertos utilizando a DNN

Treinamento	**Acurácia do treino**	**Acurácia do teste**	**Duração do treinamento**
1	94,52%	93,21%	36,5 min
2	95,22%	94,83%	39,7 min
3	91,66%	90,07%	27,1 min
4	94,43%	93,88%	39,7 min
5	95,23%	93,10%	39,9 min
6	95,99%	95,50%	39,8 min
7	93,55%	92,55%	30,2 min
8	95,40%	94,27%	37,9 min
9	95,18%	94,78%	40,1 min
10	95,82%	94,87%	39,9 min
Média	94,70%	93,71%	-
Desvio Padrão	0,0122	0,0150	-

Fonte: Do Autor (2021)

[1] Camada de *pooling* (agrupamento): operação que simplifica as informações da saída de camadas convolucionais.

O relatório de classificação do primeiro treinamento no conjunto teste, é exibido na Tabela 6. Em síntese, as piores pontuações registradas pelo modelo foram para as classes D e G, porém por motivos diferentes. Isto é observado ao analisar as pontuações de precisão e revogação de ambas as classes. Enquanto os resultados para a classe D, indicam confusão na precisão, ou seja, o modelo classificou de forma demasiada amostras de outras classes correspondentes à classe D. Em contrapartida, os resultados para a classe G, indicam confusão na revogação, ou seja, o modelo teve dificuldades em classificar corretamente as amostras referentes a sua classe. Além disso, verificou-se que nenhuma classe teve todas as amostras classificadas corretamente pelo modelo.

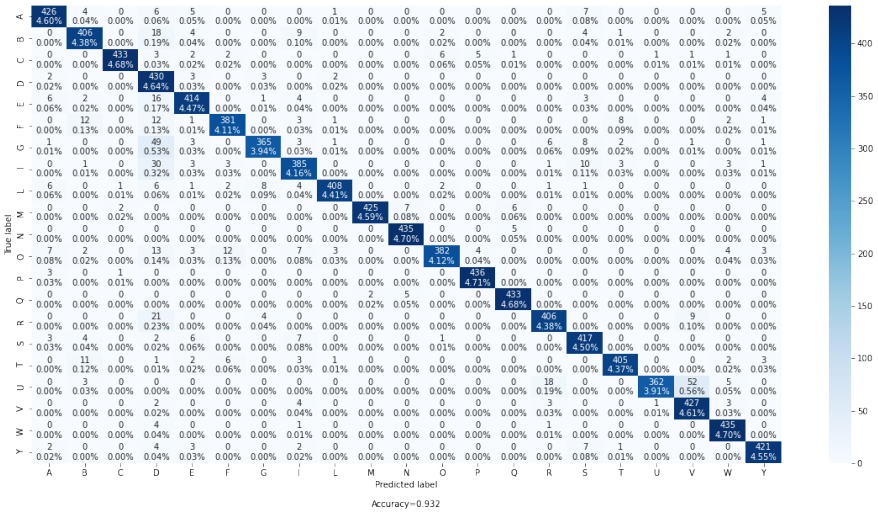
Tabela 6 – Relatório de classificação (primeiro treinamento) do conjunto de teste utilizando a DNN

Classes	**Precision**	**Recall**	**F1-Score**	**Support**
A	0,93	0,94	0,94	454
B	0,91	0,91	0,91	446
C	0,99	0,95	0,97	455
D	0,7	0,98	0,81	440
E	0,92	0,92	0,92	450
F	0,94	0,9	0,92	421
G	0,96	0,83	0,89	440
I	0,89	0,88	0,88	440
L	0,98	0,93	0,95	440
M	1,00	0,97	0,98	440
N	0,97	0,99	0,98	440
O	0,97	0,87	0,92	440
P	0,98	0,99	0,99	440
Q	0,97	0,98	0,98	440
R	0,93	0,92	0,93	440
S	0,91	0,95	0,93	440
T	0,96	0,93	0,95	434
U	0,99	0,82	0,9	440
V	0,87	0,97	0,92	440
W	0,95	0,99	0,97	441
Y	0,96	0,96	0,96	440
Média	0,937	0,932	0,933	9261

Fonte: Do Autor (2021)

A Figura 17 apresenta a matriz de confusão do conjunto de teste no primeiro treinamento, onde observa-se um certo grau de falsos positivos e falsos negativos, com realce maior nas letras G, I, U e V.

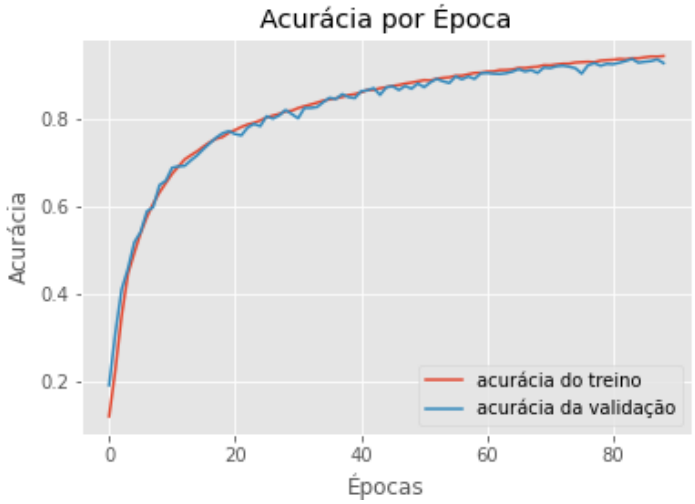
Figura 17 – Matriz de confusão do conjunto de teste (primeiro treinamento) utilizando a DNN



Fonte: Do Autor (2021)

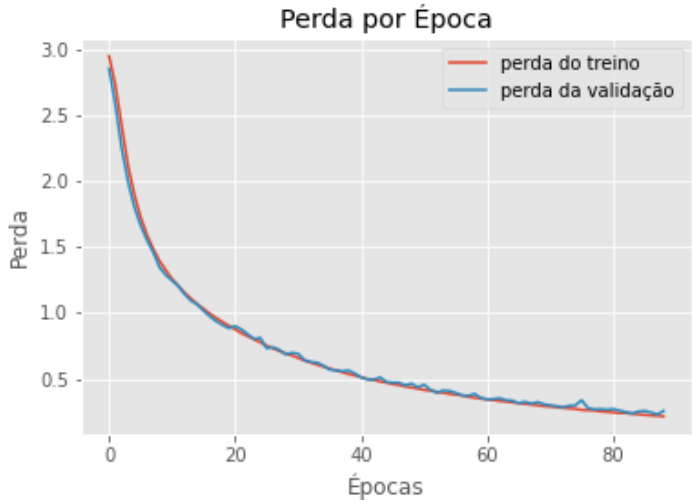
Além disto, as Figuras 18 e 19 apresentam, respectivamente, acurácia e erro do modelo ao longo das épocas de treinamento. Nestes gráficos pode-se observar a curva de aprendizagem e erro da DNN, e assim como na implementação da CNN, o treinamento encerrando antes das 100 épocas em função da parada antecipada.

Figura 18 – Gráfico da acurácia por época (primeiro treinamento) utilizando a DNN



Fonte: Do Autor (2021)

Figura 19 – Gráfico do erro por época (primeiro treinamento) utilizando a DNN



Fonte: Do Autor (2021)

5.3 Florestas Aleatórias

A utilização da biblioteca *Scikit-Learn*, a fim de realizar o treinamento da técnica de Florestas Aleatórias, provou ser adequada. Posto isto, a Tabela 6 apresenta os resultados desta técnica nos conjuntos de treino e teste. Pode-se notar um baixo desvio padrão, resultante da baixa variação do modelo em ambos conjuntos. Outro ponto a destacar, é a rapidez do treinamento das Florestas Aleatórias. Para deixar claro, o tempo de treinamento abrange somente o treinamento dos modelos, excluindo a leitura das imagens e processamento das mesmas.

Tabela 7 – Percentual de acertos utilizando a Floresta Aleatória

Treinamento	**Acurácia do treino**	**Acurácia do teste**	**Duração do treinamento**
1	97,42%	96,72%	3,1 min
2	97,34%	96,06%	2,8 min
3	97,47%	96,33%	3,4 min
4	97,65%	96,35%	3,2 min
5	97,53%	96,61%	3,4 min
6	97,48%	96,43%	2,5 min
7	98,04%	96,46%	2,7 min
8	97,76%	96,73%	3,4 min
9	97,96%	97,01%	3,6 min
10	97,53%	95,68%	2,9 min
Média	97,62%	96,44%	-
Desvio Padrão	0,0022	0,0035	-

Fonte: Do Autor (2021)

A Tabela 8 apresenta o relatório de classificação do modelo no primeiro treinamento no conjunto de teste. Em suma, podemos identificar que a classe com pior pontuação foi a letra D, seguido da letra O. Enquanto o modelo obteve êxito em classificar todas as amostras das classes C, P, Q, S e Y.

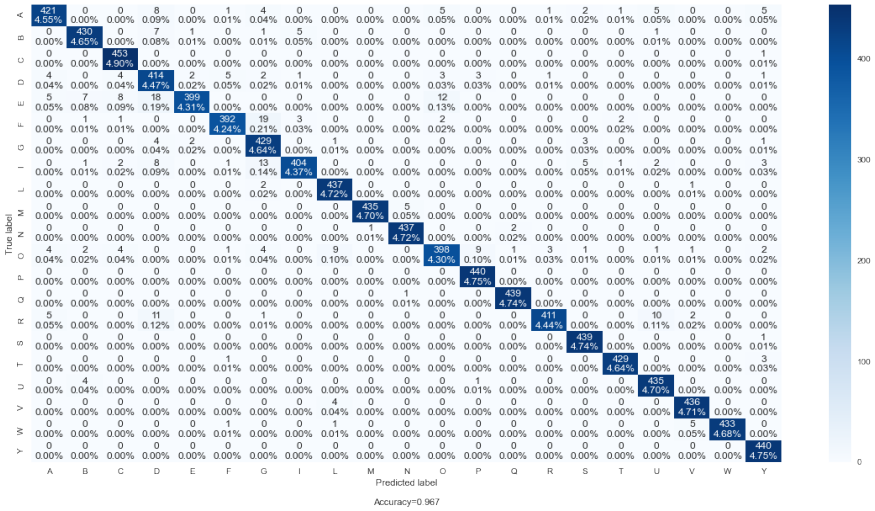
Tabela 8 – Relatório de classificação (primeiro treinamento) do conjunto de teste utilizando a Floresta Aleatória

Classes	Precision	Recall	F1-Score	Support
A	0,96	0,93	0,94	453
B	0,97	0,97	0,97	445
C	0,96	1,00	0,98	454
D	0,88	0,94	0,91	440
E	0,99	0,89	0,94	449
F	0,98	0,93	0,95	420
G	0,90	0,98	0,94	440
I	0,98	0,92	0,95	440
L	0,97	0,99	0,98	440
M	1,00	0,99	0,99	440
N	0,99	0,99	0,99	440
O	0,95	0,91	0,93	440
P	0,97	1,00	0,99	440
Q	0,99	1,00	1,00	440
R	0,99	0,93	0,96	440
S	0,98	1,00	0,99	440
T	0,99	0,99	0,99	433
U	0,96	0,99	0,97	440
V	0,98	0,99	0,99	440
W	1,00	0,98	0,99	440
Y	0,96	1,00	0,98	440
Média	0,968	0,967	0,967	9214

Fonte: Do Autor (2021)

Para completar, a Figura 20 apresenta a matriz de confusão do conjunto de teste no primeiro treinamento. Percebe-se a existência baixa de falsos positivos e falsos negativos, com destaque para a letra E. Para questões de esclarecimentos, a forma como trabalha as Florestas Aleatórias não contempla a elaboração de gráficos de treinamento, como na abordagem com redes neurais.

Figura 20 – Matriz de confusão do conjunto de teste (primeiro treinamento) utilizando a Floresta Aleatória



Fonte: Do Autor (2021)

5.4 Comparação entre as técnicas

Conforme os resultados apresentados nas seções 5.1, 5.2 e 5.3, é possível realizar um estudo avaliativo e comparativo das técnicas de Redes Neurais Convolucionais, Redes Neurais Densas e Florestas Aleatórias. De acordo com os métodos selecionados para a comparação, descritos na seção 3.8, pode-se determinar qual técnica teve um melhor desempenho na classificação de gestos do alfabeto manual. Esta seção apresenta um comparativo com base nas matrizes de confusão, nos relatórios de classificação e na acurácia média das técnicas.

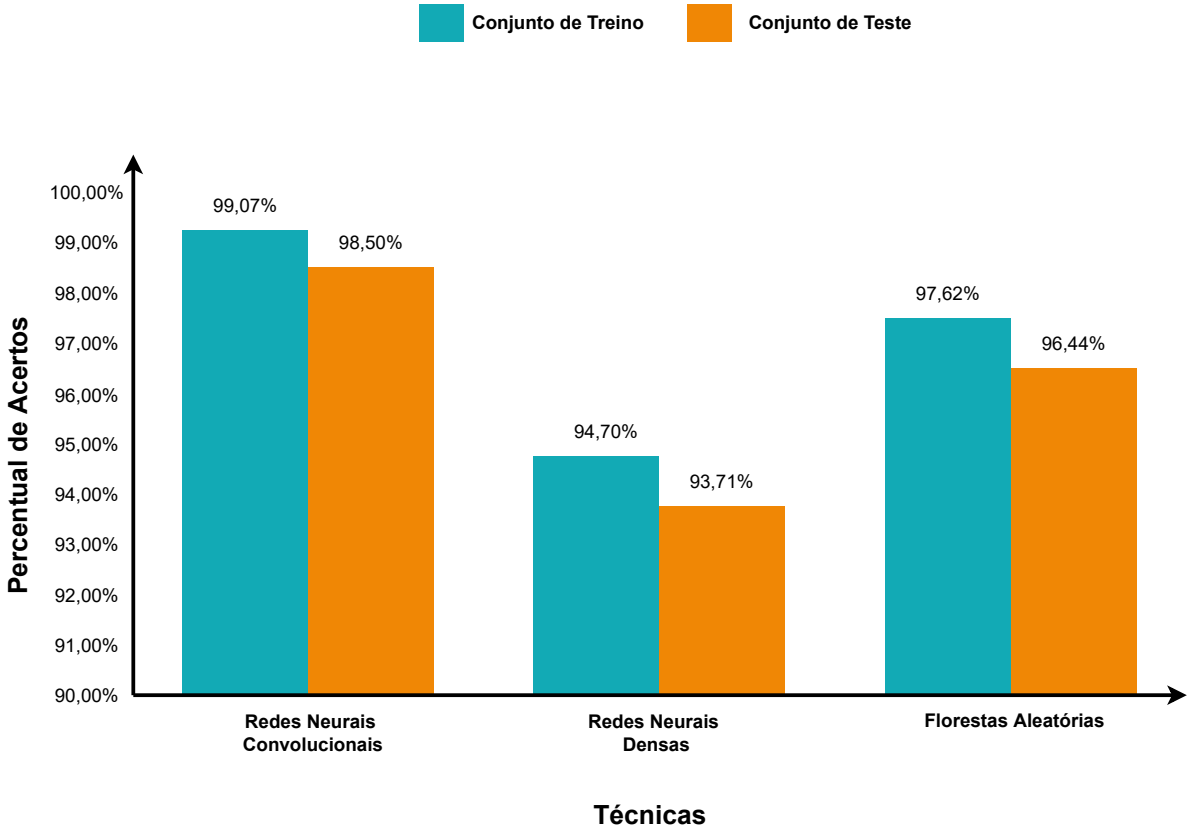
A partir das Figuras 14, 17 e 20 apresentadas nas respectivas seções de cada técnica, foi notável a maior presença de falsos positivos e falsos negativos na técnica por Redes Neurais Densas. Apesar de ser o modelo com mais parâmetros treináveis, o mesmo não foi eficaz em relacionar os valores individuais de cada *pixel*, a fim de encontrar padrões. Em contrapartida, mesmo utilizando uma abordagem alternativa, a técnica por Florestas Aleatórias obteve um desempenho bastante satisfatório, pois conseguiu identificar padrões, e isso foi visível analisando o relatório de classificação. Entretanto, a técnica por Redes Neurais Convolucionais, mostrou-se sólida ao identificar padrões, o que é verificado ao analisar os altos valores de precisão e revogação, que consequentemente geram uma pontuação F1 próximas a 1.

A partir dos relatórios de classificação obtidos de cada técnica sendo, respectivamente, as Tabelas 4, 6 e 8, foi possível investigar o desempenho das técnicas em cada classe. Neste contexto, apurou-se que a técnica por Redes Neurais Densas não teve êxito em classificar todas as amostras de alguma classe. Já a técnica por Florestas Aleatórias indicou um grau de confiabilidade melhor que a técnica por Redes Neurais Densas. Isso é observado ao verificar a quantidade de classes em

que a métrica de *recall* chegou ao seu valor máximo, nesse caso, em 5 das 21 classes presentes, o modelo obteve 100% de acerto. Por fim, a técnica por Redes Neurais Convolucionais foi a que alcançou melhor confiabilidade, isso é notado no alto valor da medida F1-Score, que na maior parte das classes obteve o valor máximo.

Posto isto, a acurácia média das técnicas de Inteligência Computacional é apresentada na Figura 21. Observa-se um alto desempenho dos resultados para a técnica por Redes Neurais Convolucionais, caracterizando-se principalmente pelo elevado grau de generalização em ambos conjuntos, com valores próximos a 99%. Em contrapartida, os resultados para a técnica por Redes Neurais Densas revelaram uma baixa efetividade na classificação comparada as outras técnicas. Por último, a técnica por Florestas Aleatórias demonstrou resultados promissores, onde o desempenho nos conjuntos de treino e teste apresentaram valores próximos, indicando uma boa generalização do modelo.

Figura 21 – Gráfico da acurácia média nos conjuntos de treino e teste utilizando todas as técnicas

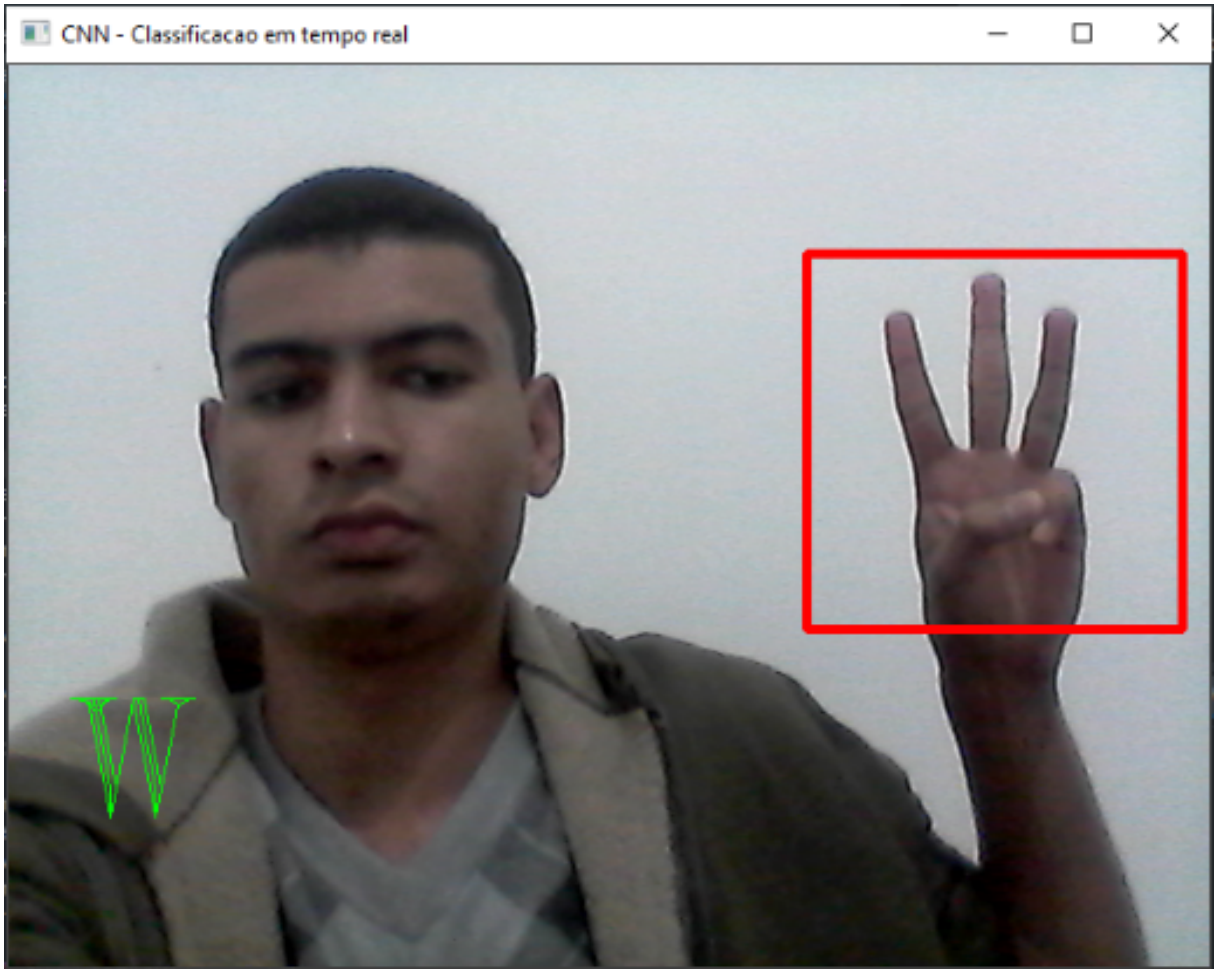


Fonte: Do Autor (2021)

5.5 Integração com a *Webcam*

A Figura 22 ilustra a aplicação em funcionamento utilizando o modelo da Rede Neural Convolucional, a qual pode-se observar a classificação correta do gesto da letra W. Ainda na Figura 22 observa-se o retângulo vermelho, a qual é a região onde o *crop* é realizado e tratado.

Figura 22 – Integração com a *webcam*



Fonte: Do Autor (2021)

6 CONCLUSÕES

O presente trabalho de conclusão de curso abordou um comparativo entre técnicas de apredizado de máquina, em que foi desenvolvido empregando redes neurais convolucionais, redes neurais densas e florestas aleatórias, todas utilizando aprendizado profundo para a tarefa de reconhecimento de gestos estáticos do alfabeto de Libras, questão esta carente na literatura. Além disto, a fim de validar os modelos treinados foi realizado uma integração em ambiente real, utilizando a webcam para avaliar o reconhecimento de gestos em tempo real.

Mediante a aplicação e análise das técnicas de Inteligência Computacional, observou-se uma melhor eficiência da técnica por redes neurais convolucionais em relação as demais técnicas aplicadas. Além disso, os resultados obtidos pela técnica por floretas aleatórias foram superiores aos da técnica por redes neurais densas e próximos aos das redes neurais convolucionais, o que indica uma boa alternativa, caso o uso da rede neural convolucional não seja viável.

Os índices médios de acurácia das técnicas por redes neurais convolucionais, redes neurais densas e florestas aleatórias atingiram até 99,07% no conjunto de treino, e até 98,50% no conjunto de teste. Para complementar, os índices de falsos positivos e falsos negativos nas técnicas por redes neurais convolucionais e floretas aleatórias foram plausíveis, em contrapartida, na técnica por redes neurais densas este índice foi notável.

Como propostas de futuros trabalhos estão: testar os modelos treinados em outras bases de dados, desenvolver uma aplicação web para aplicar inferência nas images através dos modelos treinados, criar uma base de dados com imagens de gestos de Libras, realizar testes com outras técnicas de Inteligência Computacional.

REFERÊNCIAS

ANATEL. Resolução anatel/cd nº 667 de 30/05/2016. 2016. Disponível em: <https://informacoes.anatel.gov.br/legislacao/resolucoes/2016/905-resolucao-n-667>.

ATTOH-OKINE, N. O. Analysis of learning rate and momentum term in backpropagation neural network algorithm trained to predict pavement performance. **Advances in engineering software**, Elsevier, v. 30, n. 4, p. 291–302, 1999.

BALLARD, D. H.; BROWN, C. M. **Computer Vision**. 1st. ed. [S.l.]: Prentice Hall Professional Technical Reference, 1982. ISBN 0131653164.

BARRETO, J. M. Introduçaoas redes neurais artificiais. **V Escola Regional de Informática. Sociedade Brasileira de Computaçao, Regional Sul, Santa Maria, Florianópolis, Maringá**, p. 5–10, 2002.

BENGFORT, B.; BILBRO, R. Yellowbrick: Visualizing the Scikit-Learn Model Selection Process. v. 4, n. 35, 2019. Disponível em: <http://joss.theoj.org/papers/10.21105/joss.01075>.

BISHOP, C. M. et al. **Neural networks for pattern recognition**. [S.l.]: Oxford university press, 1995.

BOGAS, J. V. A história da libras, a língua de sinais do brasil. **Comunidade surda, ensino de Libras.[2016]. Disponível em: http://blog. handtalk. me/historia-lingua-de-sinais. Acesso em**, v. 10, 2020.

BOLDT, A. S. Coleções nucleares e associação do teor de óleo de cártamo com variáveis ecogeográficas por inteligência computacional. Universidade Federal de Viçosa, 2014.

BRAGA, A. et al. **Redes neurais artificiais: teoria e aplicações**. [S.l.]: LTC editora Rio de Janeiro, Brazil:, 2007.

BRASIL. **Lei nº 10.436, de 24 de abril de 2002**. 2002. Disponível em: <http://www.planalto.gov.br/ccivil_03/LEIS/2002/L10436.htm>.

BRASIL. Lei nº 4.304 de 07 de abril de 2004. 2004. Disponível em: <https://gov-rj.jusbrasil.com.br/legislacao/136085/lei-4304-04>.

BRASIL. Lei nº 11.796, de 29 de outubro de 2008. 2008. Disponível em: <https://www.libras.com.br/lei-11796-de-2008>.

BRASIL. Lei nº 12.319, de 1º de setembro de 2010. 2010. Disponível em: <https://www.libras.com.br/lei-12319-de-2010>.

BRASIL. Lei nº 13.146, de 6 de julho de 2015. 2015. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm>.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001.

BROWNLEE, J. Machine learning mastery with python understand your data. **Create Accurate Model sand Work Projects End-To-End**, p. 123–145, 2017.

CARVALHO, P. V. d. História dos surdos no mundo e em portugal. **Lisboa: Surd'Universo**, 2007.

DEEPLEARNINGBOOK. **DeepLearningBook**. [S.l.]: Data Science Academy, 2021. <https://www.deeplearningbook.com.br/>.

DIAS, T. S. et al. **Luva instrumentada para reconhecimento de padrões de gestos em Libras**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2020.

FUKUSHIMA, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. **Neural networks**, Elsevier, v. 1, n. 2, p. 119–130, 1988.

GESSER, A. **Libras? que língua é essa?: crenças e preconceitos em torno da língua de sinais e da realidade surda**. [S.l.]: Parábola, 2009.

GODOI, E.; LIMA, M. D.; ANDRADE, V. A. B. **Língua Brasileira de Sinais: a formação continuada de professores**. [S.l.]: EDUFU, 2016. v. 3. (Material didático, v. 3).

GOODFELLOW, I. et al. **Deep Learning**. MIT Press, 2016. Disponível em: <http://www.deeplearningbook.org>.

HANDTALK. **APLICATIVO: HANDTALK**. 2012. Disponível em: <https://blog.handtalk.me/historia-lingua-de-sinais/>.

HART, P. et al. **Pattern classification**. [S.l.]: Wiley Hoboken, 2000.

HAYKIN, S. **Redes neurais: princípios e prática**. [S.l.]: Bookman Editora, 2007.

HOSSIN, M.; SULAIMAN, M. N. A review on evaluation metrics for data classification evaluations. **International journal of data mining & knowledge management process**, Academy & Industry Research Collaboration Center (AIRCC), v. 5, n. 2, p. 1, 2015.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

LECUN, Y. et al. Convolutional networks and applications in vision. In: IEEE. **Proceedings of 2010 IEEE international symposium on circuits and systems**. [S.l.], 2010. p. 253–256.

LINDA, G. **Shapiro, and George, C. Stockman (2001):"Computer Vision"**. [S.l.]: New Jersey, Prentice-Hall, 2001.

LUCAS, L. d. S. Árvores, florestas e sua função como preditores: Uma aplicação na avaliação do grau de maturidade de empresas. **Rev. Pmkt**, v. 4, n. 1, p. 6–11, 2011.

MITCHELL, T. M. et al. Machine learning. McGraw-hill New York, 1997.

MOLZ, R. F. Uma metodologia para o desenvolvimento de aplicações de visão computacional utilizando um projeto conjunto de hardware e software. 2001.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, Manole, v. 1, n. 1, p. 32, 2003.

NETO, H. S. Sistema de detecção de intrusão em redes de computadores com técnicas de inteligência computacional. 2017.

NIELSEN, M. A. **Neural networks and deep learning**. [S.l.]: Determination press San Francisco, CA, 2015. v. 25.

NISSEN, S. et al. Implementation of a fast artificial neural network library (fann). **Report, Department of Computer Science University of Copenhagen (DIKU)**, v. 31, p. 29, 2003.

OSHIRO, T. M. **Uma abordagem para a construção de uma única árvore a partir de uma Random Forest para classificação de bases de expressão gênica**. Tese (Doutorado) — Universidade de São Paulo, 2013.

PASSOS, B. T.; COMUNELLO, E. Reconhecimento do alfabeto datilológico da língua brasileira de sinais utilizando técnicas de visão computacional. **Anais do Computer on the Beach**, p. 762–764, 2019.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PESSOA, A. C. P. et al. Reconhecimento de gestos manuais para identificação de letras do alfabeto da lıngua brasileira de sinais (libras). **Universidade Federal do Maranhão, Relatório Técnico**, 2016.

PINHEIRO, L. M. Língua de sinais brasileira: libras i. **São Paulo: Know How**, 2010.

PRINCE, S. J. **Computer vision: models, learning, and inference**. [S.l.]: Cambridge University Press, 2012.

RASCHKA, S.; MIRJALILI, V. Python machine learning: Machine learning and deep learning with python. **Scikit-Learn, and TensorFlow. Second edition ed**, 2017.

RAUBER, T. W. Redes neurais artificiais. **Universidade Federal do Espírito Santo**, v. 29, 2005.

RESEARCH, U. of Illinois at Urbana-Champaign. Center for S.; DEVELOPMENT; CYBENKO, G. **Continuous valued neural networks with two hidden layers are sufficient**. [S.l.: s.n.], 1988.

SAMMUT, C.; WEBB, G. I. **Encyclopedia of machine learning**. [S.l.]: Springer Science & Business Media, 2011.

SEBE, N. et al. **Machine learning in computer vision**. [S.l.]: Springer Science & Business Media, 2005. v. 29.

SIDRA, I. Sistema ibge de recuperação automática. **Pesquisa Nacional por**, 2010.

SIMON, P. **Too big to ignore: the business case for big data**. [S.l.]: John Wiley & Sons, 2013. v. 72.

SUN, Y. et al. A new convolutional neural network with random forest method for hydrogen sensor fault diagnosis. **IEEE Access**, IEEE, v. 8, p. 85421–85430, 2020.

THARWAT, A. Classification assessment methods. **Applied Computing and Informatics**, Emerald Publishing Limited, 2020.

THEODORIDIS, S.; KOUTROUMBAS, K. Pattern recognition, academic press. **Burlington, MA.[Google Scholar]**, 2008.

VLIBRAS. **APLICATIVO: VLIBRAS**. 2016. Disponível em: <http://www.vlibras.gov.br/>.