Análise Comparativa entre Métodos de Reconhecimentos de Gestos Estáticos

Inteligência Computacional - ELT 451

Iure Rosa L Oliveira - 90688
Departamento de Engenharia Elétrica
Universidade Federal de Viçosa
Viçosa, Brasil
iure.oliveira@ufv.br

Ana Clara A Oliveira - 96698
Departamento de Engenharia Elétrica
Universidade Federal de Viçosa
Viçosa, Brasil
ana.c.altoe@ufv.br

***Abstract*— O reconhecimento automático de gestos humanos se mostra hoje como uma importante e desafiadora tarefa. Um exemplo de possível aplicação está na interpretação de linguagens de sinais. Outras aplicações incluem as novas possibilidades de interação de humanos com computadores e robôs. Tal problema vem sendo muito atacado ultimamente, porém ainda apresenta grandes desafios. Neste estudo foi atacado, especificamente, o problema de reconhecimento de gestos em LIBRAS (Linguagem Brasileira de Sinais), dando oportunidade para a realização de uma análise comparativas entre o uso de redes neurais artificiais (ANN) e o uso de lógica nebulosa.**

***Index Terms*— Redes neurais, lógica fuzzy, LIBRAS, reconhecimento de gestos.**

I. Introdução

A comunicação por Libras foi reconhecida oficialmente em 2002 e 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 comunidade de pessoas surdas do Brasil [1]. Embora existam leis que garantam o uso de Libras, em diversos contextos, ainda existem barreiras de acessibilidade para a comunidade surda, sendo ela, oriunda da falta de interpretes em locais como comércios, academias, escolas, etc..

Diversas tecnologias tem surgido para auxiliar a comunidade surda, como o *VLibras* [2] e o *Hand Talk* [3]. o *VLibras* é constituído por um conjunto de ferramentas utilizadas na tradução automática do português para Libras. O *Hand Talk* é um aplicativo que traduz frases de português para Libras. A proposta destas tecnologias é ajudar na comunicação entre as pessoas e fazer com que usuários conheçam um pouco mais sobre essa língua. Por meio de áudio ou de texto digitado, o serviço consegue, com ajuda de um avatar, transformar o áudio ou texto em sinais.

Duas relevantes linhas de pesquisa que envolvem o reconhecimento de gestos e tecnologias são a visão computacional e aprendizado de máquina. A visão computacional é definida como a ciência das máquinas que enxergam e, segundo [4], tem o objetivo de extrair informações úteis de imagens, permitindo tomar decisões sobre o ambiente e cenas por meio de imagens e vídeos [5]. O aprendizado de máquinas é um campo oriundo da inteligência computacional, que envolve a construção de algoritmos que permitem a máquina aprender sem que sejam programados explicitamente [6].

Diversos trabalhos discorrem a respeito do reconhecimento de gestos em imagens, sobretudo, com foco no alfabeto manual das línguas de sinais. [7] apresentaram $90,38\%$ de acurácia no reconhecimento de gestos do alfabeto americano, utilizando arquiteturas de redes neurais convolucionais pré-treinadas. [8] apresentaram $96,19\%$ de acurácia no reconhecimento de gestos de Libras utilizando redes neurais densas, porém com extração de recursos tradicionais. Em [9] foi obtido uma acurácia de 84% 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. Além disso, não foi encontrado a resultados interessantes utilizando lógica nebulosa. Desta forma, abre-se a oportunidade para o estudo do comparativo dessas técnicas quando aplicadas ao reconhecimento de gestos em LIBRAS, tema abordado neste trabalho.

II. Objetivos

Analisar as técnicas de redes neurais convolucionais (CNN) e Lógica Fuzzy na tarefa de classificação de gestos do alfabeto manual.

A. Objetivos Específicos

- Reconhecer padrões de Libras por meio da técnica de redes neurais convolucionais;
- Reconhecer padrões de Libras por meio da técnica de lógica nebulosa;
- Comparar as técnicas aplicadas, a fim de estabelecer a melhor técnica para o reconhecimento de gestos estáticos de Libras.

III. Referencial Teórico

Esta seção apresenta o referencial teórico sobre os conceitos e as técnicas utilizadas para a metodologia deste trabalho.

A. Linguagem Brasileira de Sinais

A LIBRAS, Língua Brasileira de Sinais, é utilizada para a comunicação entre deficientes auditivos, ou entre esta

comunidade e outras pessoas, sejam surdos ou ouvintes. Esta língua é reconhecida legalmente desde 24 de Abril de 2002, por meio da Lei nº 10.436 [10].

A comunicação nesse tipo de linguagem é feita através de gestos e expressões faciais e corporais. O que é denominado como 'palavra' na língua oral-auditiva, é chamado de 'sinal' na língua de sinais. A combinação de movimentos e expressões formam um sinal [11].

Na LIBRAS temos os conhecidos cinco parâmetros: Configuração da mão, Ponto de articulação, Movimento, Orientação e Expressão facial e/ou corporal. Através da combinação de cada um desses parâmetros os sinais são formados e a comunicação acontece. Sinais diferentes, por exemplo, podem compartilhar a mesma configuração de mão, mas serem utilizados com movimentos e orientações distintas [11].

B. 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 [12].

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, [13] apresentou a *LeNet*, uma arquitetura de CNN que foi utilizada para ler dígitos manuscritos de cartas de correspondências. De acordo com [14], as CNNs combinam 3 conceitos estruturais essenciais: os campos receptivos locais, o compartilhamento de pesos e a subamostragem (*pooling*).

De acordo com a Figura 1, pode-se destacar as principais estruturas presentes em uma CNN, descritas a seguir.

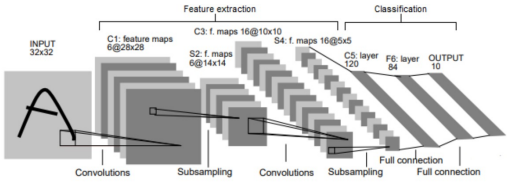


Fig. 1. Arquitetura *LeNet*.

- **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 disso, 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 denominada de campo receptivo local [14].
- **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 [14].
- **Camadas de achatamento (*flatten*):** é responsável por achatar os dados em uma matriz uni-dimensional (vetor) [15]. Este processo é necessário para a camada posterior.
- **Camadas totalmente conectadas (*dense layers*):** são responsáveis pela classificação do modelo. São as clássicas camadas totalmente conectadas, posto que as suas entradas são as saídas das camadas anteriores [15].

1) Algoritmo backpropagation: Dentre os algoritmos de aprendizado de RNAs, o *backpropagation* é o mais conhecido e utilizado. O desenvolvimento deste algoritmo foi responsável pela retomada do interesse em RNAs e o fim do "inverno da IA". O *backpropagation* realiza o treinamento 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 dividido em duas fases: *forward* e *backward* [16].

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 dos neurônios [16]. A Figura 2 apresenta uma visão do funcionamento do *backpropagation*.

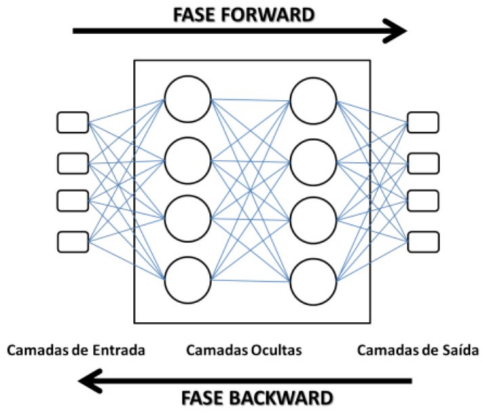


Fig. 2. Fases do algoritmo *backpropagation*.

C. Lógica Fuzzy

Na lógica *Fuzzy,* ou lógica Difusa, as variáveis possuem valores verdades entre 0 e 1, sendo números reais e 0 representando um valor falso e 1 um valor verdadeiro. Isso acontece de forma diferente da lógica Booliana, onde os valores podem ser apenas 0 ou 1. Temos então na lógica Fuzzy verdades parciais, buscando-se assim imitar o raciocínio humano [17].

Como tratam-se de cálculos onde há mais de dois valores verdade, essa lógica também é conhecida por multivalorada. Nela são aplicadas diversas regras a fim de avaliar determinada situação. Exemplos práticos da aplicação desta lógica são: medição de temperatura, reconhecimento de gestos, análise de condições de solo e sistemas de previsão de terremotos.

Na aplicação da lógica *Fuzzy* segue-se algumas etapas, sendo elas: a Fuzzyficação, a avaliação das regras, a agregação das regras e a Defuzzyficação [18].

- **Fuzzyficação:** obtenção do grau de pertinência das entradas a cada conjunto *fuzzy*.
- **Avaliação das regras:** aplicação das entradas fuzzyficadas nos antecedentes para a obtenção do consequente de cada regra.
- **Agregação das regras:** onde as funções membro dos consequentes de cada regras são agregadas.
- **Defuzzyficação:** obtenção de uma saída numérica.

D. Métodos de Avaliação

Para avaliar classificadores foram utilizados matrizes de confusão e métricas de avaliação de desempenho. No campo da Inteligência Computacional, uma matriz de confusão é uma tabela que permite a visualização do desempenho de um algoritmo de classificação [19]. As métricas de avaliação são utilizadas para medir a capacidade de generalização de um classificador treinado [20]. Os métodos utilizados são detalhados nas próximas seções.

1) Matriz de Confusão: A matriz de confusão é uma tabela que apresenta uma visão geral do desempenho de um algoritmo classificador. Segundo [21], 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 $C_1, C_2, ..., C_k$.

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:

$$M(C_i, C_j) = \sum \parallel h(x) = C_j \parallel \forall (x, y) \in T : y = C_i \quad (1)$$

TABLE I
MATRIZ DE CONFUSÃO DE UM CLASSIFICADOR MULTICLASSE

Classe	Predita C_1	Predita C_2	$\cdots$	Predita C_k
Verdadeira C_1	M(C_1, C_1)	M(C_1, C_2)	$\cdots$	M(C_1, C_k)
Verdadeira C_2	M(C_2, C_1)	M(C_2, C_2)	$\cdots$	M(C_2, C)
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Verdadeira C_k	M(C_k, C_1)	M(C_k, C_2)	$\cdots$	M(C_k, C_k)

Ademais, também pode-se destacar que em uma matriz de confusão, os elementos da diagonal principal representam o número de acertos do modelo. Além disso, os demais elementos 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:

- **Verdadeiros Positivos (VP):** indica amostras cujo um rótulo real A, é previsto como A, ou seja, acertos de classificação.
- **Falsos Positivos (FP):** indica amostras cujo rótulo real B, é previsto como A, ou seja, erros de classificação.
- **Falsos Negativos (FN):** indica amostras cujo um rótulo real A, é previsto como B, ou seja, erros de classificação.
- **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 II 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 [15].

TABLE II
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

2) Métricas: As métricas são medias 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 [15]. As métricas utilizadas neste trabalho são detalhadas nos itens a seguir.

- **Acurácia:** A acurácia é uma medida que se refere ao grau em que as previsões de um modelo correspondem à realidade modelada [15]. 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 [22].
- **Erro:** O erro é uma medida que se refere a proporção de previsões incorretas sobre o número total de amostras avaliadas [20].
- **Precisão:** A precisão, também chamada de valor preditivo positivo (PPV), é uma métrica mais específica que a acurácia, e permite uma análise por classe. Segundo [23], a precisão para uma classe A, dividido pelo total de amostras classificadas como da classe A.

- **Revogação:** A revogação, também chamada de sensibilidade, assim como a precisão, é uma métrica para análise por classe. Segundo [23], a revogação para uma classe A, é o número de amostras classificadas corretamente para a classe A, dividido pelo número de amostras da classe A.
- ***F1-Score*:** O *F1-Score* é uma medida que representa a média harmônica entre os valores de *precision* e *recall* [20]. Os valores altos significam um alto desempenho de classificação.

Para se calcular os valores de *precision, recall* e *f1-score*, pode-se utilizar a função *classification_report* da biblioteca *Scikit-Learn*.

IV. Materiais e Métodos

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

A. Recursos Utilizados

Para os treinamentos e testes das técnicas, utilizou-se um notebook com processador *Intel Core i5-7200* com $2,71GHz$ e placa gráfica integrada *Intel Graphics 620*, equipado com 12GB de RAM e *Windows 10*. Além disso, para agilizar o ajuste dos parâmetros e pequenos testes, também foi utilizado a plataforma *Google Colaboratory* [1].

Ademais, outro recurso importante empregado foi a criação de um ambiente virtual de desenvolvimento, realizado através do gerenciador de pacotes e sistemas, Conda [2]. 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 *contêiner*.

B. Base de Dados

As técnicas de inteligência computacional 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 experiência 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 [3]. 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 3 apresenta um diagrama, que exemplifica melhor a estrutura da base de dados.

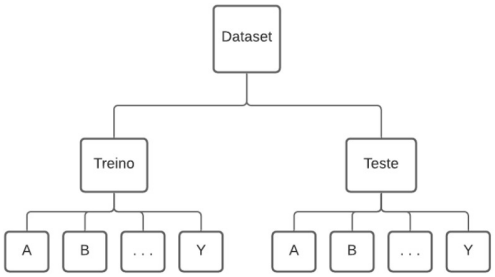


Fig. 3. Diagrama da estrutura de diretórios da base de dados.

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, facilitando a manipulação do mesmo. Para finalizar, uma observação importante quanto às letras do alfabeto presentes na 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 4 apresenta um pequeno recorte das imagens da base de dados usada.

Fig. 4. Seleção de imagens da base de dados.

1) ***Balanceamento da Base de Dados****:* Um aspecto muito importante em aprendizado de máquinas é o balanceamento de classes em uma base de dados. Isso porque os algoritmos de classificação tendem a maximizar a medida de desempenho de acurácia (*accuracy*). Logo, se a base de dados estiver fortemente desbalanceada, obtém-se resultados inconsistentes, em função das classes majoritárias e minoritárias.

[1]Ambiente interativo que permite a execução de códigos *Python* em nuvem

[2]https://conda.io/projects/conda/en/latest/index.html

[3]https://github.com/IureRosa/Computational-Intelligence

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 $C_j \in T$, 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 pela Equação 2.

$$distr(C_j) = \frac{1}{n} \sum_{i=1}^{n} \parallel y_i = C_j \parallel \tag{2}$$

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 5 demonstra a distribuição das amostras. Nota-se uma equivalência consistente e uniforme das amostras para cada classe.

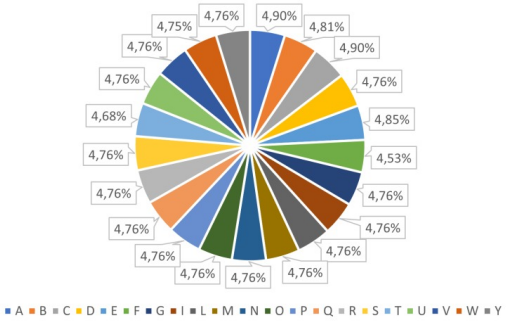


Fig. 5. Proporção de amostras para cada classe.

Além disso, também 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 encontrada nos diretórios de treino e teste. Logo, o gráfico da Figura 6 ilustra esta proporção. Podemos observar, assim como na distribuição de amostras de cada classe, uma justa conformidade entre as partes.

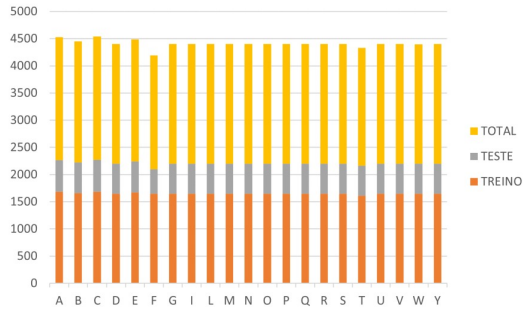


Fig. 6. Proporção da base de dados.

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.

C. 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.

Neste trabalho, algumas práticas foram empregadas para garantir que as imagens estivessem 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.

D. Treinamento 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 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.

E. 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.

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 1. 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.

2) Cuidados com Overfitting e Underfitting: Entre 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

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

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 completar, 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 alghuns neurônios (nós) aleatoriamente, durante a fase de treinamento, em funçãi de uma porcentagem definida. Dessa maneira, consegue-se simular o treinamento de diferentes RNAs.

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 Seção IV-E.1, os parâmetros ideais foram encontrados após uma série de experimentos.

F. 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 atrvavés 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 baiscamente 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, posteriormente, formatá-lo para exibir no vídeo.

G. Lógica Fuzzy

Seguindo os conceitos apresentados sobre lógica *Fuzzy* na Seção III, será apresentada a metodologia utilizada neste trabalho a fim de reconhecer os gestos presentes no *dataset* por meio desta técnica.

No método utilizado, o reconhecimento dos gestos de um banco de dados é feito através de uma luva de dados, conforme mostrada na Figura 7.

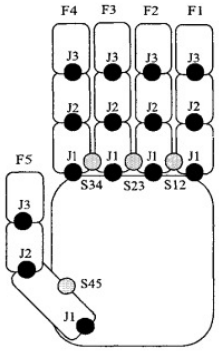


Fig. 7. Localização dos sensores na Luva de Dados.

- **Luva de Dados:** nela os dedos da mão humana são representados da seguinte maneira: F1 (dedo mínimo), F2 (anelar), F3 (dedo médio), F4 (indicador) e F5 (polegar). Além disso, é composta por quinze sensores, três juntas (J1, J2 e J3) e quatro separações (S12, S23, S34 e S45) [].

O conjunto de dados é determinado pela análise dos ângulos das articulações dos dedos (as juntas), a classificação das configurações de mão e as classificações das sequências de configurações de mão, assim um gesto é reconhecido. A segmentação dos gestos é baseado no conceito de segmento gestual monotônico, que é caracterizado por uma configuração inicial da mão, uma configuração terminal da mão e uma lista de intermediários relevantes [].

O processo de reconhecimento é dos gestos da mão é dividido em quatro etapas, sendo elas:

- **Reconhecimento das configurações dos dedos:** é reconhecida qual a posição de cada dedo, sendo estendido ou flexionado, por exemplo.
- **Reconhecimento das configurações dos mão:** é reconhecido qual a posição que a mão se encontra, ou seja, o conjunto dos dedos.
- **Segmentação do gesto em segmentos de mão monotônicos:** é reconhecido os gestos realizados pelas mãos na determinada configuração de mão.
- **Reconhecimento da sequência de seguimentos de mão monotônicos:** é reconhecida a sequência formada pelos gestos anteriormente detectados.

Para as análises acima, foram utilizadas algumas definições. Quanto aos dedos, foram classificados em reto, curvado ou semi curvado. Já as juntas, foram divididas em fechadas, semi abertas e abertas. Por fim, foram distribuídas as separações dos dedos em cruzados, fechados, semi abertos e abertos.

V. Resultados e Discussão

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.

TABLE III

PERCENTUAL DE ACERTOS UTILIZANDO A CNN

Treinamento	Acurácia do Treino %	Acurácia do Teste %	Duração do Treinamento (min)
1	99.59	98.81	60.9
2	98.99	98.19	82.1
3	99.65	98.97	57.5
4	99.56	98.91	87.6
5	99.93	99.65	67.9
6	95.91	95.01	63.7
7	99.12	98.22	62.9
8	98.47	98.22	47.3
9	99.81	99.56	77.2
10	99.71	99.48	64.7
Média	**99.07**	**98.50**	**-**
Desvio Padrão	**0.0114**	**0.0128**	**-**

A. *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 III, 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 manteve estável em ambos conjuntos, quanto para o desvio padrão, que se aproximou de zero, indicando a uniformidade dos resultados obtidos.

TABLE IV

RELATÓRIO DE CLASSIFICAÇÃO DO CONJUNTO DE TESTE.

Classe	Precisão	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

Logo após, é demonstrado na Tabela IV, 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.

A Figura 8 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. Além disso, as Figuras 9-(a-b) mostram, respectivamente, a acurácia e erro do modelo ao longo das épocas. Na Figura 9-a, 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.

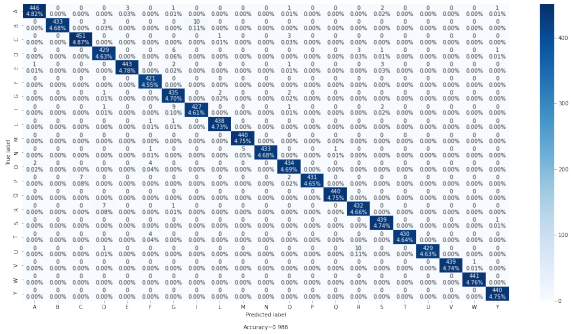


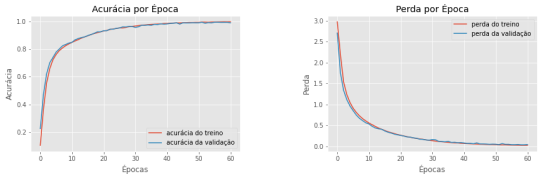
Fig. 8. Matriz de confusão do conjunto de teste (primeiro treinamento) utilizando CNN

.

Ainda na Figura 9-a, 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 9-b, 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.

B. *Lógica Fuzzy*

Logo após a etapa de validação, com a utilizando da luva de dados, foi possível classificar os gestos do *dataset*. Foram testados alguns sinais da LIBRAS e validados os reconhecimentos.



(a) Acurácia por época (b) Perdas por época

Fig. 9. Matriz de confusão do conjunto de teste (primeiro treinamento) utilizando CNN.

Uma das limitações encontradas, devido a utilização de imagens brutas, é a obtenção dos ângulos dos dedos. Devido a sobreposição dos dedos na maioria dos gestos, não é possível visualizar todos os dedos. Nesse caso, seria mais apropriado o uso de vídeos e a análise feita de diferentes ângulos.

C. Comparações entre as técnicas

Conforme os resultados apresentados nas Seções V-A e V-B, é possível realizar um estudo avaliativo e comparativo das técnicas de Redes Neurais Convolucionais e Lógica *Fuzzy*. De acordo com os métodos selecionados na Seção III-D, 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, relatórios de classificação, acurácia média obtida utilizando CNN e nas limitações de cada técnica estudada.

A técnica de Redes Neurais Convolucionais se mostrou 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. Além disso, utilizando esta técnica, observou-se um elevado nível de confiabilidade, dado que um alto valor de *F1-Score* apareceu na maior parte das classes. Porém, apesar do alto desempenho nos resultados obtidos pela CNN, caracterizando-se principalmente pelo elevado grau de generalização, com valores próximos a 99%, esta técnica possui limitações que não são problemas para a técnica de lógica *fuzzy*.

Utilizando redes neurais convolucionais, foi possível identificar apenas gestos estáticos. Algumas modificações poderiam ser implementadas para identificação de gestos dinâmicos, possibilitando a identificação de letras que necessitam de movimentação por parte do executor (J, K, X ou Z, por exemplo). No método de lógica *fuzzy*, dado que as regras se aplicam tanto ao processo de segmentação quanto aos ângulos entre dedos e juntas, é possível identificar gestos dinâmicos. Porém, no que tange a generalização, o método não é tão efetivo, pois em muitos casos, devido a sobreposição dos dedos, alguns âmgulos não são facilmente obtidos, tornando necessária, também, algumas modificações nas regras, tornando possível a visualização dos gestos de diferentes ângulos.

VI. Conclusões

O presente trabalho abordou um comparativo entre técnicas de redes neurais convolucionais e lógica *fuzzy*, aplicadas na 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 a técnica *fuzzy*. Além disso, os resultados obtidos utilizando lógica fuzzy também foram interessantes, o que indica uma boa alternativa caso o uso de rede neural convolucional não seja viável.

References

[1] "Brasil. lei nº 10.436, month = april, year = 2002, howpublished=urlhttp://www.planalto.gov.br/ccivil$_03/LEIS/2002/L10436.htm.$"

[2] "Vlibras, aplicativo," urlhttp://www.vlibras.gov.br/, May 2016.

[3] "Handtalk, aplicativo," urlhttps://blog.handtalk.me/historia-lingua-de-sinais/, May 2012.

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

[5] L. G. Shapiro, G. C. Stockman *et al.*, *Computer vision*. Prentice Hall New Jersey, 2001, vol. 3.

[6] P. Simon, *Too big to ignore: the business case for big data*. John Wiley & Sons, 2013, vol. 72.

[7] B. T. Passos and E. Comunello, "Reconhecimento do alfabeto datilológico da língua brasileira de sinais utilizando técnicas de visão computacional," *Anais do Computer on the Beach*, pp. 762–764, 2019.

[8] T. S. Dias *et al.*, "Luva instrumentada para reconhecimento de padrões de gestos em libras," Master's thesis, Universidade Tecnológica Federal do Paraná, 2020.

[9] A. C. P. Pessoa, G. Braz, L. B. Maia, R. M. P. Pereira, and T. A. Silva, "Reconhecimento de gestos manuais para identificação de letras do alfabeto da lıngua brasileira de sinais (libras)," *Universidade Federal do Maranhão, Relatório Técnico*, 2016.

[10] "Anatel. resolução anatel/cd nº 667, month = april, year = 2016, howpublished=urlhttps://informacoes.anatel.gov.br/legislacao/resolucoes/2016/905-resolucao-n-667."

[11] J. V. BOGAS, "A história da libras, a língua de sinais do brasil," *Comunidade surda, ensino de Libras.[2016]. Disponível em:¡ http://blog. handtalk. me/história–língua–de–sinais/¿. Acesso em*, vol. 10, 2020.

[12] "Deeplearningbook, month = april, year = 2021, howpublished=urlhttps://www.deeplearningbook.com.br/."

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE international symposium on circuits and systems*. IEEE, 2010, pp. 253–256.

[15] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[16] A. de Pádua Braga, A. C. P. de Leon Ferreira, and T. B. Ludermir, *Redes neurais artificiais: teoria e aplicações*. LTC editora, 2007.

[17] A. G. Aguado and M. A. Cantanhede, "Lógica fuzzy," *Artigo sem*, 2010.

[18] S. Chakraverty, D. M. Sahoo, and N. R. Mahato, "Defuzzification," in *Concepts of Soft Computing*. Springer, 2019, pp. 117–127.

[19] P. E. Hart, and R. O. Duda, *Pattern classification*. Wiley Hoboken, 2000.

[20] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.

[21] M. C. Monard and J. A. Baranauskas, "Conceitos sobre aprendizado de máquina," *Sistemas inteligentes-Fundamentos e aplicações*, vol. 1, no. 1, p. 32, 2003.

[22] J. Brownlee, *Machine learning mastery with Python: understand your data, create accurate models, and work projects end-to-end*. Machine Learning Mastery, 2016.

[23] S. Raschka and V. Mirjalili, *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.