



UNIVERSIDADE DO ESTADO DA BAHIA
DEPARTAMENTO DE CIÊNCIAS EXATAS E DA TERRA
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

ARON CAIUÁ VIANA DE BRITO

**ROBOSIGN: IA APLICADA PARA INTERAÇÃO ENTRE DEFICIENTES AUDITIVOS
E ROBÔS DE SERVIÇO**

SALVADOR

2023

ARON CAIUÁ VIANA DE BRITO

ROBOSIGN: IA APLICADA PARA INTERAÇÃO ENTRE DEFICIENTES AUDITIVOS E
ROBÔS DE SERVIÇO

Monografia apresentada ao curso de Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia - UNEB, como requisito à obtenção do grau de bacharel em Sistemas de Informação. Área de Concentração: Ciência da Computação

Orientadora: Ana Patrícia Fontes Magalhães Mascarenhas

Coorientadora: Débora Alcina Rego Chaves

SALVADOR

2023

Dedico esse trabalho a Ogum que abriu meus caminhos, me dando forças para enfrentar minhas batalhas e vencê-las. Ògún yè!!

Termo de Anuência do Orientador

Declaro para os devidos fins que li e revisei este trabalho e atesto sua qualidade como resultado final desta monografia. Confirmo que o referencial teórico apresentado é completo e suficiente para fundamentar os objetivos propostos e que a metodologia científica utilizada e os resultados finais são consistentes e com qualidade suficiente para submissão à banca examinadora final do Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação.



Prof. Ana Patrícia Fontes Magalhães

ARON CAIUÁ VIANA DE BRITO

ROBOSIGN: IA APLICADA PARA INTERAÇÃO ENTRE DEFICIENTES AUDITIVOS E
ROBÔS DE SERVIÇO

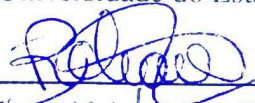
Monografia apresentada ao curso de Sistemas de Informação do Departamento de Ciências Exatas e da Terra da Universidade do Estado da Bahia - UNEB, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação. Área de Concentração: Ciência da Computação

Aprovada em:

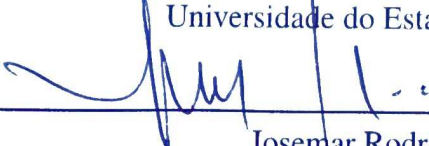
BANCA EXAMINADORA



Ana Patrícia Fontes Magalhães Mascarenhas (Orientadora)
Universidade do Estado da Bahia – UNEB



Débora Alcina Rego Chaves (Co-Orientadora)
Universidade do Estado da Bahia – UNEB



Josemar Rodrigues de Souza
Universidade do Estado da Bahia - UNEB

AGRADECIMENTOS

Primeiramente gostaria de agradecer a deus e a todos os orixás por terem me dado força, perseverança e caminho para enfrentar essa etapa da minha vida. Também agradecer a meus pais Valmir e Márcia por todo apoio, suporte e principalmente carinho. A meu Babalorixá Matheus por todos ensinamentos, cuidados e muito Axé, que me permitiram evoluir de forma a estar preparado para crescer, entender e conquistar a vida. A Luiza por todo amor, carinho, companherismo, por me ouvir e por estar presente quando preciso. A Bruna por ser minha irmã, parceira, melhor amiga e conselheira. A meu grupo de amigos Vongola por serem meus irmãos de outras mães com todo suporte e companherismo. A meu irmão Abner por todas as dicas de como vencer no caminho. A Andréia e Kleiton por todo companherismo, carinho e dividirem comigo a vida. A Sirius por ser o melhor cachorro do mundo e me entender com olhares e latidos. Por fim a Ana e Débora que me ajudaram e orientaram a desenvolver esse trabalho.

Passar por esse processo certamente não é fácil e exige muito esforço. Ainda bem que felizmente carrego comigo um pedacinho de todos, que me fortalece, me motiva e me faz ser quem sou, um resultado de experiências, interações, amores, amizades e energias que me permitem ser sempre melhor. Obrigado por tudo!

RESUMO

Robôs de serviço são construídos com intuito de realizar tarefas domésticas para ajudar na rotina dos seres humanos, performando atividades como tirar o lixo, buscar itens, limpar o chão, entre outras. Para iniciar essas atividades é necessário uma interação humano-robô onde o humano dê um comando, o robô compreenda e inicie a tarefa desejada. Essa comunicação é comumente realizada através de funcionalidades de reconhecimento de voz, por ser o meio natural de comunicação dos humanos. Entretanto faz-se necessário considerar pessoas que não possuem a fala e precisam interagir e usufruir das funcionalidades do robô também de forma natural. Nesse contexto, o problema de pesquisa trabalhado nesse projeto é a inclusão na interação entre robôs de serviço e deficientes auditivos. O objetivo é desenvolver uma solução de interação entre robô de serviço e pessoas com deficiência auditiva através de reconhecimento de comandos na Língua Brasileira de Sinais (Libras). A metodologia proposta no projeto contempla a construção de um dataset do zero simplificado para um computador de baixo poder computacional, treinamento de um modelo de rede neural recorrente utilizando MediaPipe Landmark Detector e LSTM para realizar o reconhecimento de sinais e utilização do modelo para integração com o sistema operacional do robô *Bot Intelligent Large Capacity and Low Cost* (BILL). Esse modelo foi validado durante treinamento com dados não conhecidos, atingindo 99% de acurácia média. O modelo também foi avaliado em testes executados em tempo real através de uma câmera, atingindo 85% de acurácia média. Esses resultados demonstram que é possível realizar interação humano-robô através do reconhecimento de Libras com acurácia considerável, sem alto custo computacional devido a utilização da simplificação da coleta de dados e uso algoritmo LSTM para treinamento da rede. O projeto se destaca pela relevância social trazendo acesso à informação onde o reconhecimento de Libras permite que pessoas surdas se comuniquem e interajam com robôs de forma mais eficiente e natural. Isso significa que eles podem obter acesso a informações, serviços e recursos que muitas vezes são limitados ou inacessíveis devido à barreira da comunicação.

Palavras-chave: Reconhecimento de Libras; Robôs de serviço; Interação humano-robô.

ABSTRACT

Service robots are built with the aim of carrying out domestic tasks to help human beings in their routines, performing activities such as taking out the trash, fetching items, cleaning the floor, among others. To start these activities, a human-robot interaction is necessary in a way that a human gives a command, the robot understands and starts the desired task. This communication is commonly carried out through voice recognition functionalities, as it is the natural means of communication for humans. However, it is necessary to consider people who cannot speak and need to interact and enjoy the robot's functionalities in a natural way. In this context, the research problem addressed in this project is the inclusion in the interaction between service robots and hearing impaired people. The objective is to develop an interaction solution between a service robot and people with hearing impairments through command recognition in Libras. The methodology proposed in the project includes the construction of a simplified dataset from scratch for a computer with low computational power, training of a recurrent neural network model using MediaPipe Landmark Detector and LSTM to perform signal recognition and use of the model for integration with the robot operating system BILL. This model was validated during training with unknown data, reaching 99% average accuracy. The model was also evaluated in tests carried out in real time using a camera, reaching 85% average accuracy. These results demonstrate that it is possible to perform human-robot interaction through Libras recognition with considerable accuracy, without high computational cost due to the use of simplified data collection and the use of the LSTM algorithm for network training. The project stands out for its social relevance, bringing access to information where the recognition of Libras allows deaf people to communicate and interact with robots in a more efficient and natural way. This means they can gain access to information, services and resources that are often limited or inaccessible due to the communication barrier.

Keywords: Libras recognition; Service robots; human robot-interaction.

LISTA DE FIGURAS

Figura 1 – Metodologia do projeto	13
Figura 2 – Sinais em libras	21
Figura 3 – Arquitetura de funcionamento CNN.	24
Figura 4 – Arquitetura de funcionamento 3DCNN.	25
Figura 5 – Primeiro momento do desenvolvimento do projeto	30
Figura 6 – Pontos chave da detecção da mão	31
Figura 7 – Arquitetura da rede neural	36
Figura 8 – Arquitetura de fluxo - Solução	39
Figura 9 – Interface gráfica - Solução	40
Figura 10 – Resultados em tempo real por classe	47

LISTA DE ABREVIATURAS E SIGLAS

3DCNN	<i>Convolutional Neural Network 3D</i>
ACSO	Centro de pesquisa em Arquitetura de Computadores, Sistemas inteligentes e Robótica
API	<i>Application Programming Interface</i>
BILL	<i>Bot Intelligent Large Capacity and Low Cost</i>
BILSTM	<i>Bi-directional Long Short Term Memory</i>
CNN	<i>Convolutional Neural Network</i>
KNN	<i>K-Nearest Neighbor</i>
Libras	Língua Brasileira de Sinais
LIBRAS	<i>Lingua Brasileira de Sinais</i>
LSTM	<i>Long Short Term Memory</i>
RNN	<i>Recurrent Neural Network</i>
ROS	Robot Operating System

SUMÁRIO

1	INTRODUÇÃO	9
1.1	Objetivos e Hipótese	11
1.2	Metodologia	12
2	TRABALHOS RELACIONADOS	15
3	FUNDAMENTAÇÃO TEÓRICA	18
3.1	Robôs de serviço	18
3.2	Interação humano-robô	19
3.3	Deficiência auditiva	20
3.4	Língua Brasileira de Sinais	21
3.5	Mediapipe Landmark detector	22
3.6	Redes neurais	23
3.6.1	Redes neurais convolucionais	23
3.6.2	Redes neurais convolucionais 3D	24
3.7	Redes Neurais Recorrentes	25
3.7.1	Long Short Term Memory	27
3.8	Reconhecimento de sinais	28
4	DESCRIÇÃO DO PROJETO	30
4.1	Coleta de dados	31
4.2	Etapa de pré-processamento	34
4.3	Criação da rede e treinamento	35
4.4	Classificação de sinais	39
5	VALIDAÇÃO	42
5.1	Resultados validação durante treinamento	44
5.2	Resultados validação em Tempo real	45
5.2.1	Discussão	46
6	CONSIDERAÇÕES FINAIS	49
	REFERÊNCIAS	52

1 INTRODUÇÃO

Na área da robótica é possível observar o desenvolvimento de robôs de serviço realizando tarefas domésticas com intuito de facilitar cada vez mais a rotina dos seres humanos (1). A execução dessas atividades se torna possível a partir de sensores que dão percepção do mundo real ao robô, permitindo assim realizar tarefas de forma autônoma ou semi-autônoma. A partir desses dados o robô consegue desenvolver ações básicas como reconhecer pessoas, reconhecer e manipular objetos, navegar de forma autônoma, reconhecer voz e planejar tarefas mais complexas como: limpar a casa, buscar objetos, tirar o lixo, recepcionar uma pessoa com suas preferências, servir comidas e bebidas, dentre outros que necessitam de funcionamento conjunto das funcionalidades básicas.

No processo de desenvolvimento de alguma tarefa, é necessário que exista uma interação entre o humano e o robô para que o mesmo possa identificar qual comando deve ser realizado e a medida que a atividade vai sendo concluída o robô pode apresentar *feedbacks* ao humano, de forma a sinalizar se ele está conseguindo concluir a atividade ou não, sendo assim uma comunicação exercida entre as partes (2). Nesse contexto, a forma de interação mais comum e prática é através da voz (2), justamente por ser a principal forma de comunicação utilizada pelos humanos. O robô realiza seu reconhecimento e o traduz para um comando que iniciará a atividade identificada. Entretanto é necessário considerar que pessoas com deficiência auditiva seriam excluídas da interação com robôs que utilizam voz como principal forma de receber comandos. Nesse cenário podemos considerar a intervenção de reconhecimento de sinais como a mais adequada para incluir essas pessoas na interação humano-robô.

Para desenvolvimento dessa intervenção foi considerado o robô de serviço BILL (3) proveniente do Centro de pesquisa em Arquitetura de Computadores, Sistemas inteligentes e Robótica (ACSO) (4). BILL é a proposta da equipe BahiaRT para participar da competição Robocup na liga RoboCup@Home. Nessa competição, diversos robôs de serviço são avaliados para determinar a qualidade e utilidade de suas funcionalidades. BILL foi desenvolvido em 2014 como resultado de pesquisas e projetos na área de robótica de serviço.

As principais habilidades de BILL incluem a capacidade de se comunicar com seres

humanos utilizando linguagem natural. Além disso, ele é capaz de reconhecer objetos, rostos, gênero, idade, bem como de se mover autonomamente em ambientes fechados. Suas funcionalidades são ativadas principalmente por meio do reconhecimento de voz, o que proporciona uma oportunidade para a aplicação do reconhecimento de sinais. Nesse sentido foi pensado na solução "ROBOSIGN", uma inteligência artificial treinada para reconhecer comandos executados em *Língua Brasileira de Sinais* (LIBRAS) tornando a interação entre robôs de serviço e seres humanos inclusiva.

No Brasil pessoas com deficiência auditiva possuem a LIBRAS como forma de comunicação oficial (5). A LIBRAS representa modalidade gestual-visual onde é possível se comunicar através de gestos, expressões faciais e corporais. É reconhecida como meio legal de comunicação e expressão desde 24 de Abril de 2002, através da Lei Federal nº 10.436 que diz "Entende-se como Língua Brasileira de Sinais - Libras a forma de comunicação e expressão, em que o sistema lingüístico de natureza visual-motora, com estrutura gramatical própria, constituem um sistema lingüístico de transmissão de idéias e fatos, oriundos de comunidades de pessoas surdas do Brasil". A LIBRAS é utilizada na comunicação com pessoas surdas, sendo, portanto, uma importante ferramenta de inclusão social. A partir dessa informação podemos considerar que para o meio brasileiro a interação humano-robô para pessoas com deficiência auditiva pode ser desenvolvida através do reconhecimento de LIBRAS, traduzindo os sinais reconhecidos para comandos.

Além do viés técnico esse projeto possui relevância social através de aspectos como: O reconhecimento de LIBRAS proporciona acesso à informação, permitindo que pessoas surdas se comuniquem e interajam de maneira mais eficiente e natural com robôs. Isso significa que eles podem ter acesso a informações, serviços e recursos que muitas vezes são limitados ou inacessíveis devido à barreira da comunicação. Robôs com capacidade de reconhecer LIBRAS podem ajudar a preencher essa lacuna e oferecer um acesso mais igualitário à informação. Além disso, essas tecnologias contribuem para a inclusão social de pessoas com deficiência auditiva, reduzindo seu isolamento social e promovendo sua participação plena em diversos setores da sociedade, como educação, trabalho, saúde e entretenimento. No contexto educacional, as interações humano-robô podem ser aplicadas para facilitar o aprendizado de estudantes surdos, atuando como assistentes virtuais que fornecem suporte, instruções e *feedback*, auxiliando no desenvolvimento de habilidades acadêmicas e linguísticas. Por fim, o reconhecimento de Libras

em robôs contribui para tornar os serviços e recursos tecnológicos mais acessíveis e proporcionar igualdade de oportunidades para a comunidade surda.

1.1 OBJETIVOS E HIPÓTESE

Com a contextualização da área através dos trabalhos relacionados citados no capítulo 2, é possível identificar a variedade de tecnologias que são utilizadas para diferentes aplicações no contexto de reconhecimento de sinais. No entanto, há poucos estudos voltados especificamente para reconhecimento de Libras e nenhum deles aborda a interação entre humanos e robôs, sendo assim uma lacuna no estado da arte e este projeto atua em cima dessa oportunidade. Complementar a isso, como citado anteriormente, temos diversas contribuições sociais que podem ser obtidas através dos resultados do projeto, reforçando ainda mais a motivação para sua implementação.

A partir das justificativas discutidas foi formulado que o problema da pesquisa consiste na inclusão de deficientes auditivos na interação com robôs de serviço. A hipótese é que o uso de reconhecimento de comandos em Libras pode viabilizar a interação entre deficientes auditivos e robôs de serviço usando o algoritmo LSTM e o ROS.

Nesse sentido o objetivo geral desse projeto consiste em desenvolver uma solução de interação entre um robô de serviço e pessoas com deficiência auditiva. Essa solução busca proporcionar uma comunicação efetiva e inclusiva, utilizando recursos de reconhecimento e interpretação da Libras. A ideia é permitir que pessoas possam se comunicar e interagir de forma natural com o robô, facilitando o acesso a informações, serviços e assistência, promovendo sua autonomia e inclusão social. Para atingir esse objetivo geral, foram propostos objetivos secundários/específicos que são pequenas partes que compõem o objetivo geral. Sendo eles:

- Mapear o estado da arte de soluções de interação homem-máquina para deficientes auditivos - A partir da revisão sistemática identificar quais redes neurais mais utilizadas, quais são suas vantagens e desvantagens, o que é necessário para utilizá-los, *datasets* de treinamentos, formas de pré-processamento de dados, entre outras técnicas que agregem à solução
- Definir a solução para o robô de serviço - Após obter o leque de tecnologias, escolher dentre elas qual melhor se encaixa no contexto do projeto.
- Implementar a solução - Realizar preparação dos dados, realizar o treinamento do modelo

utilizando a rede neural escolhida e o dataset selecionado, criar o código para capturar entradas da câmera, passar pelo modelo treinado e registrar o resultado, e por fim integrar o resultado obtido com o sistema do robô de serviço para disparar alguma ação.

- Validar a solução - Verificar os resultados obtidos do sistema através de um estudo de caso.

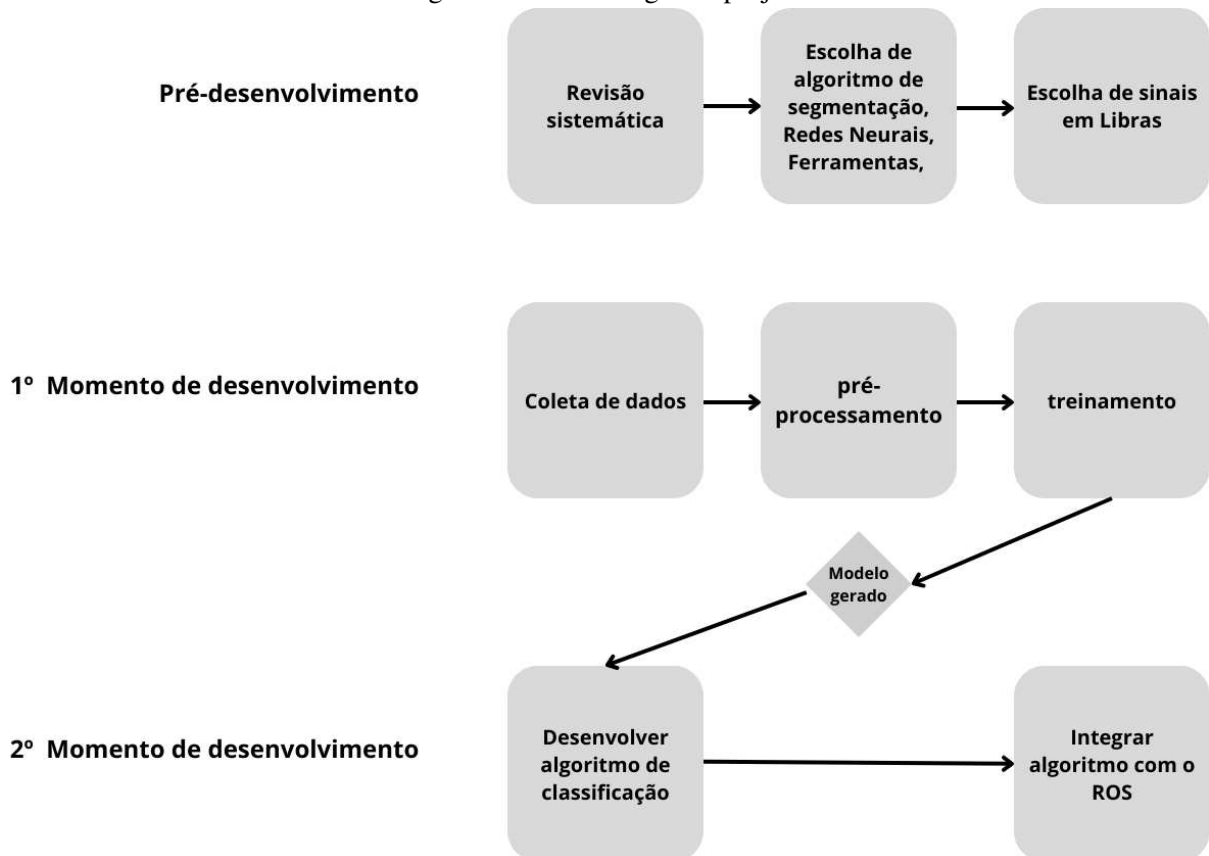
1.2 METODOLOGIA

A resolução desse projeto para atingir os objetivos propostos foi estruturada a partir da seguinte metodologia. Inicialmente foi realizado um mapeamento do estado arte em relação a interações humano-robô no que se refere a reconhecimento de sinais. Com esse intuito foi realizada uma revisão sistemática da literatura investigando algoritmos, tipos de interação, formas de treinamento no contexto de Libras. Com base nos resultados da pesquisa foram escolhidos o algoritmo de segmentação de mão, o tipo de rede neural que foi utilizada para treinamento, os conjuntos de ferramentas que compuseram a solução adotada e a criação de um dataset de treinamento, onde para isso foram escolhidos sinais significativos que representam palavras utilizadas como comandos para execução de tarefas do robô BILL no domínio de uma casa. O desenvolvimento do projeto foi contemplado em 2 momentos. O primeiro consistiu na construção do modelo de reconhecimento de comandos em libras, no qual é necessária coleta de dados, pré-processamento, treinar o modelo e por fim gerar o modelo treinado. O segundo consistiu no desenvolvimento de um algoritmo para utilizar o modelo que foi gerado no primeiro momento e integração dos resultados do modelo com o Robot Operating System (ROS), de forma que o código do modelo emita os sinais em texto e o ROS os receba para iniciar alguma funcionalidade, dessa forma concluindo a interação humano-robô. A metodologia descrita do projeto pode ser visualizada na figura 1

Para atingir a solução proposta é necessária uma etapa de preparo onde foi montado o ambiente de desenvolvimento no computador escolhido em questão que possui as seguintes configurações: 8GB memória ram, processador intel® Core™ i5-8265U CPU @ 1.60GHz × 8, placa gráfica NVIDIA Corporation GM108M [GeForce MX110], armazenamento 1 tera de HD e sistema operacional instalado Ubuntu 22.04.3 LTS. A instalação do Ubuntu é um pré-requisito para que seja instalado o sistema operacional do ROS e todas as dependências necessárias pelos algoritmos. O ROS foi instalado na versão 2, distribuição *Iron*.

O primeiro momento da solução começa com o levantamento dos sinais que precisam

Figura 1 – Metodologia do projeto



Fonte: Autoria própria

ser reconhecidos por BILL, tomando como base as suas funcionalidades existentes. Com a decisão dos sinais é possível iniciar a etapa de coleta de dados que contempla a gravação de vídeos que servirão como dados que representam os sinais e serão utilizados no treinamento da rede. A próxima etapa consiste no pré-processamento dos dados que consiste em converter os dados para um formato aceitável para o treinamento da rede escolhida. Por fim ao primeiro momento, a etapa de treinamento contempla a construção de uma rede neural recorrente LSTM utilizando a biblioteca tensorflow e salvar em disco o modelo treinado.

Com o fim do treinamento e obtenção do modelo capaz de identificar os sinais propostos no projeto é possível iniciar o segundo momento do projeto que é a construção do do algoritmo utilizando o modelo treinado, realizando classificação dos sinais propostos em tempo real através de uma câmera. Os resultados emitidos pelo modelo serão enviados ao ROS como um comando para inicialização de uma funcionalidade do robô.

Por fim, para verificar a qualidade e performance do modelo treinado foi decidido por duas formas de validação: Validação durante o treinamento utilizando uma porcentagem de dados retirado diretamente do dataset e validação em tempo real através de uma câmera captando

a execução dos sinais ao vivo e registrando as predições do modelo para analisar sua acurácia nesse cenário.

O projeto está organizado da seguinte maneira: O capítulo 2 trata sobre os estudos correlatos encontrados através de uma revisão sistemática. O capítulo 3 apresenta a fundamentação teórica, explicando conceitos, arquiteturas e algoritmos que serão desenvolvidos no projeto. O capítulo 4 contempla a descrição do projeto onde foi discutido detalhes necessários de sua implementação, como definição de algoritmos, coleta de dados, treinamento, entre outros. No capítulo 5 é exibido a validação e resultados adquiridos no projeto. Por fim no capítulo 6, conclusão, é abordado análise dos resultados, trabalhos futuros e considerações finais a respeito das relevâncias entregues pelo projeto.

2 TRABALHOS RELACIONADOS

A literatura atual referente ao uso de Libras para interação com robô apresenta poucos trabalhos, entretanto para outros usos como gestos, sinais americanos, indianos, filipinos, chineses, existe uma gama de artigos que utilizam diferentes abordagens para realizar o reconhecimento desses sinais. A partir desses trabalhos, foi possível entender o estado da arte em relação ao reconhecimento de sinais e possíveis algoritmos a serem utilizados no reconhecimento de Libras.

A área de reconhecimento de gestos é normalmente dividida entre abordagens baseadas em sensores e abordagens baseadas em visão computacional (6). A primeira abordagem faz utilização de sensores como luvas (7) e eletromiografias (8) para codificar e rastrear o movimento da mão. Esse método apesar de obter bons resultados em termos de precisão, é uma abordagem não confortável para o usuário e de custo elevado devido a estrutura dos sensores. Por outro lado na abordagem baseada em visão computacional (6) é necessário apenas uma câmera para captar a imagem de entrada e um algoritmo para realizar o reconhecimento do objeto exposto na imagem. Com o avanço crescente das redes neurais, essa opção se tornou a mais viável economicamente em questão de custo. Sua acurácia já alcança níveis consideráveis, e é confortável em relação as outras formas que envolvem vestir o sensor que reconheceria os sinais (9).

Dentro da área de reconhecimento de sinais via visão computacional existe outra divisão que considera sinais estáticos como é observado em (10), onde a análise é realizada apenas em um único *frame* do vídeo, sendo mais comum na identificação de sinais que não possuem movimento como algumas letras e palavras. Para tal esse estudo faz uso de classificadores de redes neurais e classificadores *K-Nearest Neighbor* (KNN) lidando especialmente quando há mudanças na iluminação, posição das mãos e fundo, tornando o reconhecimento mais complexo. Em (11), se discute sobre o reconhecimento dinâmico, segunda abordagem dos baseados em visão computacional. Nesse estudo é treinado um modelo *Convolutional Neural Network 3D* (3DCNN) para realizar o reconhecimento de 20 sinais da Linguagem de Sinais Indianos que possuem movimento em sua execução. A 3DCNN possui mecanismos de análise temporal que

identificam *frames* em sequência, permitindo assim reconhecer esses sinais com movimento, e consequentemente criando uma interação robusta e representativa.

O estado da arte contempla diversos estudos relacionados ao reconhecimento de sinais dinâmicos em diferentes aplicações para variados tipos de gesto, cada um com suas particularidades. Entretanto no geral, seus objetivos são reconhecer gestos complexos fazendo uma análise espaço-temporal na sequência de *frames* de um vídeo, com intuito de identificar padrões de movimento. O mais comum é a utilização de *Convolutional Neural Network* (CNN) modificadas para receberem uma camada de treinamento responsável por extrair essas características espaço-temporais dos dados, sendo chamada agora de 3DCNN (12)(11). Esse tipo de rede é comumente encontrada em aplicações que lidam com dados volumétricos, como vídeo, sequências temporais e dados espaciais em 3 dimensões, sendo considerada a base para estudos desse contexto. Entretanto sua desvantagem é a necessidade de grande quantidade de dados para treinamento e complexidade de parametrização. Existem variações com maior acurácia, menor complexidade no quesito de parametrização e quantidade de dados para treinamento e menor custo computacional comparado as 3DCNN tradicionais. Por exemplo, o trabalho apresentado em (13) justifica o uso da variante ResC3D que utiliza a arquitetura resnet pois ela possui menos parametros de treinamento em relação as tradicionais 3DCNN, o que torna mais fácil a sua implementação. Os autores de (14) apresentam uma versão alternativa chamada *3D separable convolution* que se diferencia pela utilização de mecanismo de atenção chamado *Self-Attention*. Esse algoritmo é responsável por analisar os dados dentro de uma sequência, comparando cada dado com os outros dados e tenta extrair o significado no contexto da sequência e não dos dados individualmente, consequentemente melhorando a acurácia da extração de *features*.

O uso das 3DCNN também é comumente encontrado em conjunto com outros tipos de redes como redes neurais recorrentes, criando modelos híbridos que conseguem analisar sequências de frames ainda maiores e aumentando a acurácia do reconhecimento, tornando o sistema ainda mais robusto e próximo da realidade. No trabalho apresentado em (13) é discutido um exemplo da utilização das 3DCNN em conjunto com as *Recurrent Neural Network* (RNN) e lá é descrito que modelos baseados em 3DCNN são incapazes de capturar informações temporais de longo alcance. Por isso, defende o uso conjunto do algoritmo *Long Short Term Memory* (LSTM) que tem seu funcionamento baseado em células de memória para armazenar e extrair features temporais, propondo assim um modelo misto que usa 3DCNN e LSTM, cada uma

focando em um nível espaço-temporal.

A utilização do algoritmo LSTM pode ser observado em (15), onde se destaca a importância do reconhecimento de gestos, impulsionado pelo avanço da inteligência artificial. Com foco na língua de sinais americana (ASL), é discutida uma abordagem eficiente baseada em visão para reconhecimento de gestos estáticos e dinâmicos. O sistema utiliza a plataforma Mediapipe do Google para obter pontos de referência da mão, enquanto o reconhecimento de gestos é realizado por meio de redes neurais recorrentes LSTM. O estudo experimental atingiu uma precisão de 99% na identificação dos 26 gestos do alfabeto ASL, oferecendo potencial para a conversão de gestos em texto, com implicações em comunicação, controle de robôs, automação residencial e aplicações médicas.

O uso conjunto do algoritmo de LSTM com as 3DCNN é recorrente em diferentes estudos como por exemplo em (16)(17)(12) onde são usados para extrair *features* temporais de longo e curto prazo identificando classes de gestos através de informações em sequência proveniente de vídeos. Os trabalhos apresentados em (12) e (18) temos a utilização de um aperfeiçoamento do LSTM chamado *Bi-directional Long Short Term Memory* (BILSTM), que utiliza informações do passado e do futuro da sequência de frames o que leva a uma melhora na acurácia devido a um maior contexto da análise, enquanto o LSTM básico utiliza apenas o passado como parametro de extração. Outra variante encontrada foi o *Residual LSTM* demonstrado em (19) que é um aperfeiçoamento feito no algoritmo LSTM a fim de evitar que o gradiente da imagem desapareça no processo de treinamento e extrair melhor as características espaciais da imagem.

Em (6) é abordado a criação de um modelo *deep learning multi-stream* de reconhecimento de sinais das linguagens brasileira, indiana e coreana. O modelo combina diferentes tipos de informação para a tarefa de classificação utilizando *single-stream* e *multi-stream* 3DCNN. Nesse estudo o treinamento do reconhecedor de sinais é feito exclusivamente com *datasets* oferecidos gratuitamente na internet, onde é utilizado apenas os dados que não possuem informações de profundidade, visto que é proposto a utilização de Redes Generativas Adversárias para gerar mapas de profundidade artificialmente. Esse estudo se destaca pela utilização do dataset MINDS-Libras que compõem 20 gestos realizados por 12 usuários, além da arquitetura independente de sensores de profundidade como câmera Kinect por exemplo.

3 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo introduz o conhecimento teórico necessário para entendimento do trabalho.

3.1 ROBÔS DE SERVIÇO

Robôs de serviço (1) são sistemas robóticos projetados para realizar tarefas específicas e interagir com seres humanos em diferentes contextos. Eles são desenvolvidos para auxiliar e facilitar atividades domésticas, comerciais ou industriais. Esses robôs são equipados com sensores, atuadores e inteligência artificial, permitindo que percebam o ambiente, interajam com os usuários e executem suas funções designadas.

O funcionamento dos robôs de serviço baseia-se em uma combinação de *hardware* e *software* (20). Eles são programados para responder a comandos e estímulos específicos, seja por meio de interfaces de voz, toque ou outras formas de interação. A inteligência artificial é frequentemente utilizada para aprimorar sua capacidade de aprendizado e adaptação às necessidades dos usuários. Além disso, algoritmos de planejamento e controle são empregados para garantir que as atividades sejam realizadas de forma eficiente e segura.

As principais características dos robôs de serviço incluem a capacidade de interação social, a adaptabilidade, a segurança e a autonomia (1). Eles são projetados para interagir de forma natural e intuitiva com seres humanos, tornando a comunicação mais eficaz. Além disso, são capazes de aprender com a experiência e se adaptar às preferências e necessidades dos usuários, utilizando algoritmos de reconhecimento facial, voz, objetos, dentre outros. A segurança é uma preocupação central, com sensores e algoritmos de detecção de colisão que garantem a integridade física dos envolvidos. Por fim, a autonomia é uma característica importante, permitindo que os robôs de serviço executem suas atividades sem a necessidade de constante supervisão humana.

Existem várias aplicações reais de robôs de serviço em funcionamento atualmente. Em hospitais e clínicas (21), eles são utilizados para auxiliar em cirurgias, transporte de materiais e apoio na reabilitação de pacientes. Em estabelecimentos comerciais (22), como lojas e

aeroportos, os robôs de serviço atuam como assistentes de atendimento ao cliente, fornecendo informações e orientações. Em residências, robôs de limpeza automatizados têm se tornado cada vez mais comuns. Esses são apenas alguns exemplos das muitas aplicações reais dos robôs de serviço.

3.2 INTERAÇÃO HUMANO-ROBÔ

A interação humano-robô refere-se ao processo de comunicação e colaboração entre seres humanos e robôs (23). É uma área de pesquisa e desenvolvimento que busca criar sistemas inteligentes capazes de interagir e se comunicar de maneira natural com os humanos, seja por meio de interfaces de voz, gestos, toque ou outras formas de interação. O objetivo é estabelecer uma comunicação eficaz e intuitiva, permitindo que os robôs compreendam as intenções dos humanos e respondam adequadamente.

Existem várias formas de interação humano-robô, cada uma com suas características e aplicações específicas. Uma forma comum é a interação física, onde os robôs podem ser controlados por meio de dispositivos de manipulação ou toque direto (24). Isso permite que os humanos guiem ou instruam os robôs em tarefas específicas. Além disso, a interação verbal é amplamente utilizada (25), permitindo que os robôs entendam comandos de voz e respondam por meio de síntese de fala. Também há a interação gestual (26), em que os robôs podem reconhecer e interpretar gestos humanos, como acenos, movimentos das mãos e expressões faciais, para interagir de maneira mais intuitiva.

Diversas tecnologias são empregadas para viabilizar a interação humano-robô. Os avanços em inteligência artificial, aprendizado de máquina e processamento de linguagem natural são fundamentais para permitir que os robôs compreendam e interpretem as intenções dos humanos. Sensores, como câmeras, microfones e sensores táteis, são utilizados para capturar informações do ambiente e das interações humanas (27). Além disso, técnicas de reconhecimento de voz, processamento de imagem e análise de sentimentos são aplicadas para melhorar a capacidade de percepção e compreensão dos robôs.

A interação humano-robô pode ser especialmente relevante para a inclusão de pessoas com deficiência. Por exemplo, pessoas com deficiência motora podem se beneficiar de interfaces de controle adaptadas, como comandos por voz ou dispositivos de controle acessíveis (26). Para pessoas com deficiência visual, os robôs podem fornecer informações sonoras ou táteis sobre o

ambiente, auxiliando na navegação e no reconhecimento de objetos (23). Além disso, a interação por meio de linguagem de sinais pode ser usada para permitir a comunicação entre pessoas surdas e robôs. Essas tecnologias e abordagens estão sendo desenvolvidas com o objetivo de promover a inclusão e a acessibilidade, proporcionando oportunidades iguais de interação e participação para todas as pessoas.

3.3 DEFICIÊNCIA AUDITIVA

A deficiência auditiva é uma condição que afeta a capacidade de uma pessoa em ouvir sons em diferentes graus (28). Pode variar desde uma perda auditiva leve até a surdez total. Pessoas com deficiência auditiva podem enfrentar dificuldades na percepção de sons, na compreensão da fala e na participação em conversas e atividades auditivas.

As causas da deficiência auditiva podem ser congênitas, ou seja, presentes desde o nascimento, ou adquiridas ao longo da vida. Entre as causas congênitas estão fatores genéticos, infecções durante a gravidez ou complicações no parto. Já as causas adquiridas podem incluir exposição a ruídos intensos, infecções, lesões, envelhecimento ou efeitos colaterais de certos medicamentos (29).

A deficiência auditiva é classificada em diferentes níveis, com base na gravidade da perda auditiva. Esses níveis são classificados como leve, moderado, severo e profundo (28). A classificação é baseada nos limiares de audição, que medem a menor intensidade de som que uma pessoa pode detectar em diferentes frequências (28).

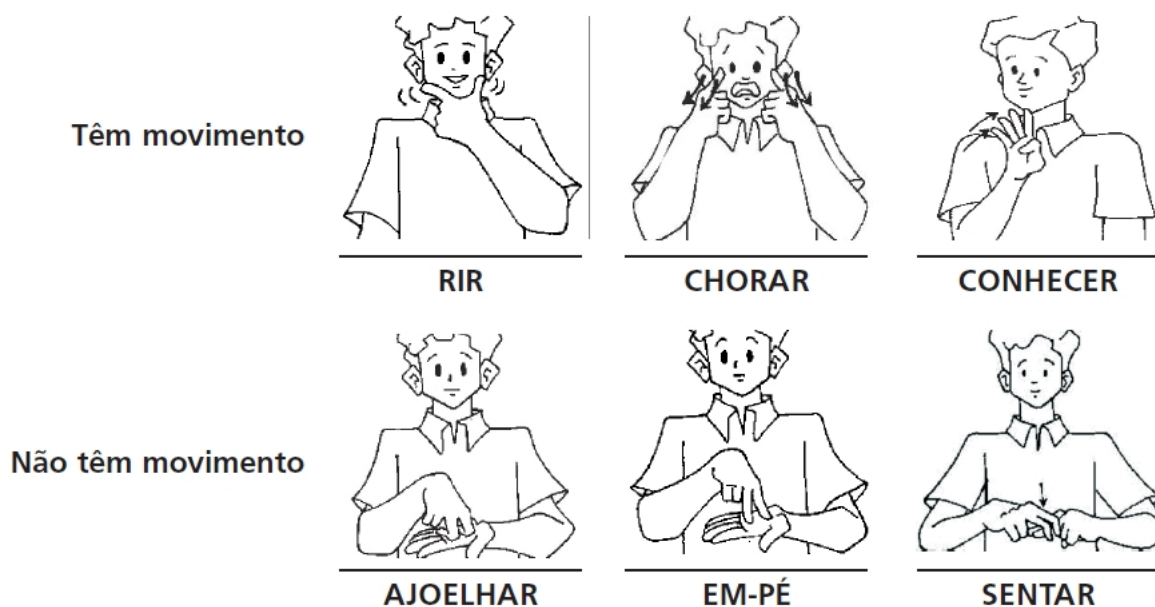
A deficiência auditiva pode ter impactos significativos na comunicação e na interação social (29). A dificuldade em perceber sons e compreender a fala pode levar a barreiras na comunicação verbal, tornando difícil acompanhar conversas, entender instruções ou participar plenamente de atividades em grupo. Isso pode levar a sentimentos de isolamento, frustração e exclusão social. Além disso, a deficiência auditiva também pode afetar a capacidade de se comunicar por telefone ou em ambientes ruidosos. A interpretação de sinais não verbais, como expressões faciais e linguagem corporal, também pode ser prejudicada, resultando em dificuldades na interpretação de nuances e emoções durante a interação social.

3.4 LÍNGUA BRASILEIRA DE SINAIS

Libras é a sigla para Língua Brasileira de Sinais, uma modalidade gestual-visual utilizada pela comunidade de pessoas surdas no Brasil (5). É uma língua completa, com estrutura gramatical própria, que permite a comunicação por meio de gestos, expressões faciais e movimentos corporais.

A Libras possui uma estrutura gramatical distinta da língua portuguesa. É uma língua visual-espaco-gramatical, em que a posição das mãos, a expressão facial e corporal, além do espaço e movimento, desempenham papéis importantes na comunicação. A gramática da Libras é baseada em princípios visuais e espaciais (30). Ela é estruturada em torno de parâmetros, como configuração de mãos, movimento, ponto de articulação e orientação das mãos. Esses parâmetros são combinados para formar sinais que representam palavras ou conceitos. A ordem das palavras em uma frase na Libras geralmente segue uma estrutura tópico-comentário, onde o tópico é introduzido antes do comentário principal. A figura a seguir exemplifica o uso de Libras.

Figura 2 – Sinais em libras



Fonte: Os Cinco Parâmetros da Libras (31)

A Libras é fundamental para a comunicação e inclusão das pessoas com deficiência auditiva (32). Ela permite que os surdos se expressem, compartilhem ideias, emoções e informações, assim como qualquer pessoa que se comunique por meio de uma língua oral. Através da Libras, as pessoas surdas podem participar ativamente da sociedade, ter acesso à educação,

ao mercado de trabalho e às atividades cotidianas. Além disso, a Libras promove a valorização da identidade surda, fortalecendo a cultura e a comunidade surda. A importância da Libras vai além da comunicação entre surdos. Ela também desempenha um papel crucial na interação entre surdos e ouvintes (32). Ao aprender Libras, as pessoas ouvintes podem se comunicar diretamente com os surdos, estabelecendo uma comunicação mais inclusiva e acessível. Isso contribui para a quebra de barreiras e para a promoção da igualdade de oportunidades para as pessoas surdas.

3.5 MEDIAPIPE LANDMARK DETECTOR

O *MediaPipe Landmark Detector* (33) é uma biblioteca de código aberto desenvolvida pelo Google que se destina a identificar pontos de referência em imagens ou vídeos, permitindo a detecção precisa de elementos anatômicos, como rostos, mãos e outros objetos. Sua principal aplicação é na área de visão computacional e rastreamento de movimento, onde é amplamente utilizado em uma variedade de aplicações, incluindo realidade aumentada, reconhecimento gestual, análise de postura e muito mais.

O funcionamento do *MediaPipe Landmark Detector* baseia-se em redes neurais convolucionais que foram treinadas em conjuntos de dados vastos e diversificados para identificar e localizar pontos de interesse em imagens. Esses pontos de interesse são representados por coordenadas 2D ou 3D que descrevem a posição de elementos específicos, como olhos, nariz, boca, dedos e articulações das mãos, em relação ao resto da imagem. A detecção é realizada em tempo real, o que torna o *MediaPipe Landmark Detector* uma ferramenta valiosa para aplicações interativas e em tempo real.

O *MediaPipe Landmark Detector* oferece suporte a vários tipos de detecção, com uma ênfase particular na detecção de rostos, mãos e objetos. No contexto da detecção de mãos, a biblioteca é capaz de identificar e rastrear com precisão as posições das principais articulações das mãos em tempo real. Isso inclui a localização dos nós dos dedos, as articulações dos dedos e a base da palma. Com essas informações, é possível criar aplicações que respondam aos movimentos das mãos, como gestos ou interações diretas com objetos virtuais.

A detecção de mãos pelo *MediaPipe Landmark Detector* é realizada através da análise de redes neurais que identificam a forma das mãos e, em seguida, estimam a posição das articulações com base nessa forma. A precisão e a robustez dessa detecção a tornam uma ferramenta valiosa em diversas aplicações, como controle gestual, realidade virtual, reconheci-

mento de gestos e muito mais. A biblioteca oferece uma *Application Programming Interface* (API) amigável que permite aos desenvolvedores integrar facilmente essa funcionalidade em seus próprios projetos, tornando-a acessível para uma ampla gama de aplicações criativas e práticas.

3.6 REDES NEURAIIS

Uma rede neural é um modelo computacional inspirado no funcionamento do cérebro humano. Ela é composta por um conjunto interconectado de unidades de processamento chamadas de neurônios artificiais (34). Cada neurônio recebe entradas, realiza cálculos e gera uma saída. Esses neurônios são organizados em camadas, formando uma estrutura em que a informação flui desde a camada de entrada até a camada de saída.

Uma rede neural funciona através do processo de aprendizado, no qual os pesos das conexões entre os neurônios são ajustados para que a rede seja capaz de realizar uma tarefa específica. Durante a fase de treinamento, a rede é exposta a um conjunto de dados de entrada e as saídas desejadas são fornecidas. Com base nessa informação, a rede ajusta seus pesos por meio de algoritmos de otimização, como o gradiente descendente, para minimizar o erro entre as saídas geradas e as saídas desejadas.

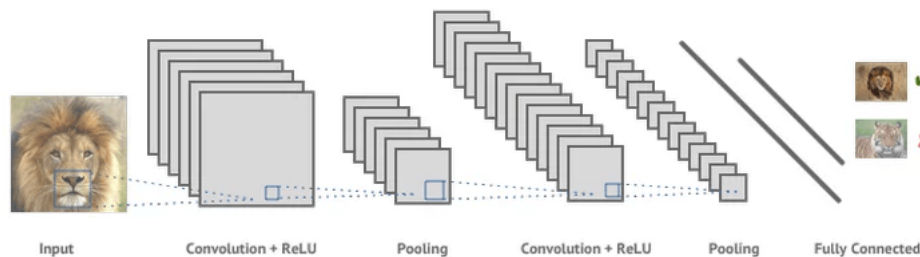
Uma vez treinada, a rede neural pode ser utilizada para fazer previsões ou tomar decisões com base em novos dados de entrada. Ela é capaz de realizar tarefas como classificação, reconhecimento de padrões, regressão, processamento de linguagem natural, entre outras. O funcionamento da rede neural é baseado em uma combinação de processamento paralelo e distribuído, permitindo que ela seja capaz de lidar com problemas complexos e encontrar padrões não lineares nos dados.

3.6.1 Redes neurais convolucionais

CNN são um tipo especializado de rede neural projetada para processar dados com estrutura de grade, como imagens (35). Elas são compostas por camadas convolucionais que extraem características relevantes das entradas e camadas totalmente conectadas que realizam a classificação. Essas redes possuem um histórico de alta acurácia para problemas de classificação devido à sua capacidade de aprender representações de características hierárquicas, ao compartilhamento de pesos, à aplicação de *pooling* para reduzir a dimensionalidade e ao uso de funções de ativação não lineares. Essas características combinadas permitem que as redes

capturem informações discriminativas dos dados de entrada e realizem classificações precisas em uma ampla gama de problemas. A figura 3 abaixo temos uma visualização da arquitetura básica de uma rede neural convolucional.

Figura 3 – Arquitetura de funcionamento CNN.



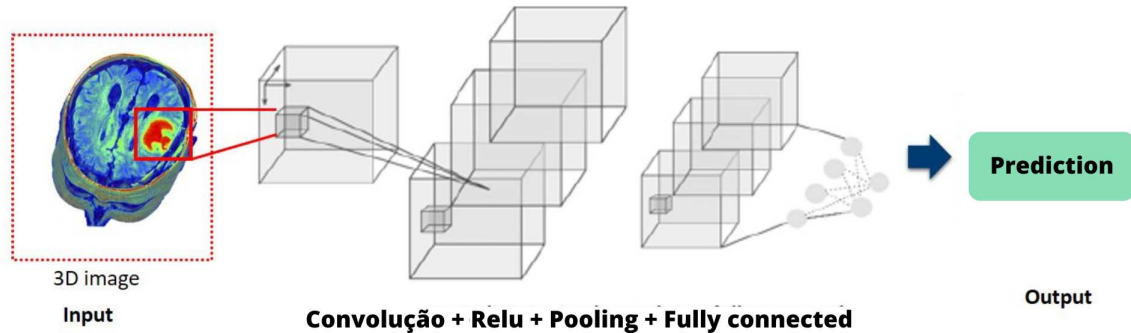
Fonte: Retirada de "O que são redes neurais convolucionais?"(29)

3.6.2 Redes neurais convolucionais 3D

Enquanto as CNN tradicionais são projetadas para processar dados bidimensionais, como imagens, as 3DCNN levam em consideração a dimensão temporal, adicionando uma terceira dimensão ao modelo (36). Isso permite que elas capturem informações espaço-temporais e explorem a evolução dos dados ao longo do tempo. O funcionamento básico das 3DCNN é semelhante ao das CNN tradicionais. Elas são compostas por camadas convolucionais, camadas de *pooling* e camadas totalmente conectadas. A principal diferença está na estrutura das camadas convolucionais. Enquanto nas CNN tradicionais as convoluções são bidimensionais, nas 3DCNN elas são tridimensionais, aplicadas em volumes de dados ao invés de imagens planas. Isso significa que os filtros convolucionais são aplicados em três dimensões: largura, altura e tempo. A arquitetura das 3DCNN permite que elas capturem padrões e características em diferentes níveis de abstração ao longo das três dimensões. Elas são capazes de aprender características espaço-temporais complexas, como movimento, mudanças de forma e interações entre objetos ao longo do tempo. Além disso, as 3DCNN também utilizam camadas de *pooling* 3D, que reduzem a dimensionalidade dos mapas de características, assim como as camadas de *pooling* nas CNN tradicionais. Isso ajuda a extrair características relevantes, preservando informações importantes nos dados volumétricos. A figura 4 ilustra a arquitetura padrão de divisão em camadas da 3DCNN

A camada de convolução é a pedra angular de uma 3DCNN. Ela é responsável por aprender padrões e características nas três dimensões dos dados de entrada (largura, altura

Figura 4 – Arquitetura de funcionamento 3DCNN.



Fonte: Retirada de "O que são redes neurais convolucionais?"(34)

e profundidade/tempo). No contexto de imagens em movimento (vídeos) ou volumes 3D, a profundidade se refere à dimensão temporal. A camada de *pooling* é usada para reduzir a dimensionalidade dos *feature maps* produzidos pelas camadas de convolução. Isso ajuda a reduzir a quantidade de parâmetros na rede e a torná-la menos suscetível a *overfitting*. As camadas totalmente conectadas são tipicamente usadas no final da rede, após as camadas de convolução e *pooling*. Elas são compostas por neurônios que estão conectados a todos os neurônios da camada anterior. Essas camadas são responsáveis por tomar as características extraídas das camadas anteriores e transformá-las em uma forma adequada para a tarefa de classificação ou regressão.

3.7 REDES NEURAIS RECORRENTES

RNN são um tipo de arquitetura de rede neural projetada para lidar com dados sequenciais, como sequências de texto, séries temporais, dados de fala e outros tipos de dados em que a ordem e a dependência temporal entre os elementos são cruciais (37). Ao contrário das redes neurais convolucionais, que são projetadas para processar dados com estrutura espacial, as RNN têm a capacidade de lidar com a dependência temporal nas sequências de dados.

A arquitetura básica de uma RNN consiste em células de memória que mantêm informações anteriores e as atualizam a cada novo elemento na sequência. A principal ideia por trás das RNNs é que cada célula mantém um estado oculto (*hidden state*) que é atualizado

com base no estado anterior e na entrada atual. Isso permite que a rede capture informações contextuais e dependências temporais. A célula recorrente mais comumente usada é a LSTM, que possui mecanismos adicionais para controlar o fluxo de informações ao longo da sequência e lidar com problemas de dependência de longo prazo.

O funcionamento de uma RNN pode ser descrito camada por camada:

- Camada de Entrada:
 - A camada de entrada é a primeira camada da rede e é responsável por receber os dados de entrada sequencial. Cada elemento na sequência de entrada é mapeado para um vetor de entrada. Por exemplo, em uma tarefa de processamento de linguagem natural, cada palavra em uma frase pode ser representada como um vetor *one-hot* ou incorporada em um espaço vetorial. Um vetor "*one-hot*" é uma representação vetorial binária usada para codificar categorias discretas, onde apenas um elemento do vetor é "ligado" (valor 1) e todos os outros são "desligados" (valor 0). Por fim cada elemento da sequência é fornecido à camada recorrente em cada etapa de tempo, permitindo que a rede processe a sequência elemento por elemento.
- Camada Recorrente:
 - A camada recorrente é a principal componente de uma RNN, pois ela é responsável por propagar a informação ao longo da sequência atualizando o estado interno (memória) em cada passo de tempo. Em cada etapa de tempo, a camada recorrente recebe dois tipos de entrada: o vetor de entrada atual (da camada de entrada) e o estado oculto anterior (do passo de tempo anterior). Isso permite que a rede mantenha uma memória das informações anteriores da sequência. A camada recorrente realiza operações matemáticas para combinar o vetor de entrada atual com o estado oculto anterior, produzindo um novo estado oculto. Esse novo estado oculto é passado para o próximo passo de tempo e pode ser usado para tomar decisões ou prever a saída. A célula LSTM é frequentemente usada nessa camada devido à sua capacidade de armazenar informações relevantes por períodos prolongados, evitando o problema de desvanecimento ou explosão do gradiente.
- Camada de Saída:
 - A camada de saída é a última camada da rede e é responsável por produzir as previsões ou saídas com base nas informações processadas pela camada recorrente. A estrutura

da camada de saída depende da tarefa. Para problemas de classificação, a camada de saída pode conter um ou mais neurônios, onde cada neurônio representa uma classe possível e produz uma pontuação de confiança para essa classe. Funções de ativação como Softmax são frequentemente usadas para converter essas pontuações em probabilidades de classe. Para tarefas de regressão, a camada de saída pode conter um único neurônio que produz uma estimativa contínua. As previsões da camada de saída são comparadas com os rótulos reais (no caso de tarefas supervisionadas) para calcular a perda, e o gradiente dessa perda é usado para atualizar os pesos da rede durante o treinamento.

É importante ressaltar que a camada recorrente é aplicada repetidamente para cada elemento da sequência, propagando informações passadas para os próximos passos de tempo. Dessa forma, a RNN é capaz de capturar dependências temporais e modelar a estrutura sequencial dos dados. No entanto, as RNN tradicionais podem sofrer do problema de desvanecimento ou explosão do gradiente, que dificulta o treinamento de dependências de longo prazo. Para lidar com isso, surgiram variantes como a LSTM.

3.7.1 Long Short Term Memory

Uma LSTM é composta por unidades de memória chamadas células, que possuem três componentes principais: uma porta de esquecimento (*forget gate*), uma porta de entrada (*input gate*) e uma porta de saída (*output gate*) (38). Essas portas controlam o fluxo de informações dentro da célula e permitem que a LSTM aprenda a armazenar e acessar informações relevantes em diferentes momentos no tempo. A porta de esquecimento decide quais informações antigas devem ser descartadas da célula, com base nas informações de entrada e nas informações passadas. A porta de entrada controla a atualização das informações da célula, levando em consideração a nova entrada e as informações antigas. A porta de saída determina qual parte das informações na célula será exposta à saída da LSTM.

Ao incorporar camadas LSTM nas redes neurais convolucionais 3D, é possível capturar informações espaço-temporais complexas de maneira mais eficiente. As camadas convolucionais 3D extraem características espaciais em cada quadro ou *slice* do volume de dados, enquanto as camadas LSTM permitem a captura de dependências temporais ao longo da sequência temporal. Essa combinação de 3DCNN com LSTM é poderosa para analisar e modelar

dados volumétricos que possuem uma componente espaço-temporal. A arquitetura da LSTM permite que ela lide com dependências temporais de longo prazo, aprendendo a armazenar e acessar informações relevantes em diferentes intervalos de tempo.

3.8 RECONHECIMENTO DE SINAIS

Sistemas de reconhecimento de sinais são comumente estruturados a partir dos seguintes passos (39): 1. aquisição de dados, nessa etapa as imagens são capturadas para treinar o modelo; 2. pré-processamento: Imagens são convertidas para formatos necessários para o treinamento dos modelos; 3. Segmentação: Detectar onde na imagem está a mão, entregando apenas esse dado para a próxima etapa; 4. Extração de *feature*: depois de reconhecer os segmentos necessários da imagem, as matrizes de recursos são preparadas e extraídas as características relevantes para identificação da classe, 5. Classificação: Passo final onde ocorre o reconhecimento. Foi observado que a maioria dos estudos se estrutura de uma forma similar, mudando apenas o que é utilizado em cada passo.

Na etapa de aquisição de dados há duas abordagens a serem seguidas: Utilizar um *dataset* disponível ou treinar seu próprio *dataset*. A exemplo da primeira abordagem existem *datasets* para gestos comuns como *zoom-out*, *zoom-in*, arrastar para os lados, clicar, entre outros que servem de propósito geral para interação básica com sistemas (40), linguagem de sinais filipinas (41), linguagem de sinais indiana (6), coreana (6) e brasileira (6). O *dataset* da língua de sinais brasileira (Libras) se chama MINDS-Libras *dataset* e foi o único encontrado em todos os estudos correlatos. Já para a segunda abordagem existem estudos que explicam do zero todo o passo a passo de como coletar dados e gerar um *dataset* próprio, como por exemplo para linguagem indiana contemplando 20 gestos (11) e linguagem de sinais americana (17).

A etapa de segmentação representa a detecção da mão no frame da imagem previamente ao reconhecimento do sinal realizado. A partir desse método podemos aplicar o algoritmo de reconhecimento de sinais somente na porção da mão identificada, o que reduz o processamento e aumenta a acurácia devido ao foco direto na mão sem precisar analisar todo o resto do frame. Alguns exemplos encontramos foram: *Mediapipe landmark detector* (9)(17); *Skeleton data extraction* (42), *OpenPose* citado em (18), Yolo v4 (19), *Convolutional Self-Organizing Map* (43). Todos utilizando redes neurais convolucionais como algoritmo de classificação para identificar as mãos na imagem.

Na etapa de extração é onde ocorrerá o treinamento do modelo a partir dos dados gerados na segmentação, utilizando redes neurais. Nessa fase, o reconhecimento dinâmico faz-se necessário a utilização da 3DCNN devido a sua capacidade de processar o tempo em uma sequência de frames (12). Como citado na secção de estudo correlatos, essas redes possuem diversas variações que realizam a extração de características do objeto treinado utilizando menos parametros de configuração, consequentemente facilitando a implementação. Os autores de (13) e (40) apresentam variações dessas redes que necessitam de menos parametros de configuração para realizar o treinamento desses tipos de redes tornando mais fácil a sua implementação.

4 DESCRIÇÃO DO PROJETO

Como visto na sessão anterior o desenvolvimento do projeto foi dividido em dois momentos. O primeiro tem seu fluxo expresso na figura 5. Nesse fluxo é reunido tudo que é necessário para treinar um modelo capaz de reconhecer sinais em Libras.



Fonte: Autoria própria

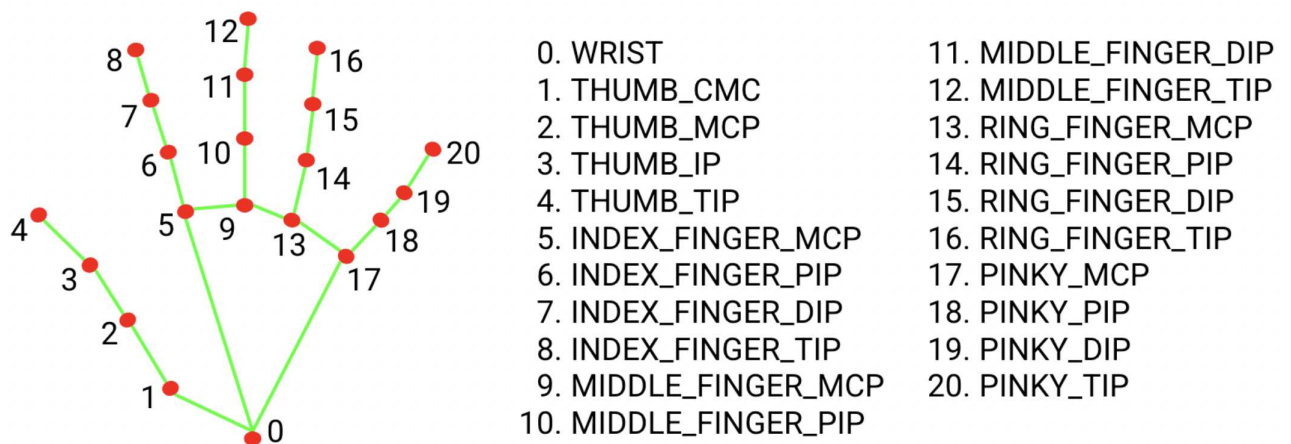
Inicialmente como explicito na imagem é necessário definir os sinais que precisam reconhecidos por BILL.

BILL participa de uma liga de robôs de serviço chamada Robocup@AtHome, lá existem tarefas de diversos níveis e foi pensado em escolher comandos que auxiliem no início e fim de funcionalidades que são utilizadas no nível 0, o mais básico. A partir disso os sinais escolhidos foram: Olá Bill, Siga-me, Pare, Treinar pessoas, Reconhecer pessoas, Reconhecer objeto, Sim, Não e Vazio. A ideia principal é que o módulo identifique primeiramente Olá Bill, para dar início a interação, após isso ele reconhecerá o sinal realizado e poderá iniciar as seguintes funcionalidades: Seguir, treinamento do reconhecimento de pessoas, reconhecimento de pessoas e reconhecimento de objetos, que estão diretamente relacionados respectivamente com os comandos "siga-me", "treinar pessoas", "reconhecer pessoas", "reconhecer objetos" e sinais auxiliares como "Sim", "Não" e "Pare" para interromper as tarefas e para os possíveis feedbacks que o robô deseje emitir e receber. O sinal de "Vazio" é uma forma de identificar quando o usuário não está realizando nenhum sinal, de forma que o algoritmo não tente prever o sinal sem que o mesmo esteja efetivamente sendo executado.

4.1 COLETA DE DADOS

A coleta de dados consiste em reunir material de treinamento para a rede neural baseado nos sinais de comandos escolhidos anteriormente. Esses dados em questão foram obtidos através de uma gravação em tempo real com uma câmera e um algoritmo para identificar a mão na imagem. O código de segmentação de mão faz a leitura de cada *frame* emitido pela imagem da câmera e envia esse dado para o modulo MediaPipe, responsável por realizar a detecção das mãos na imagem. O MediaPipe gera diversos landmarks para mão, onde cada um é um ponto chave de identificação da mão na imagem e cada um desses pontos possui valores de coordenada x,y,z. A figura 6 abaixo ilustra a disposição de cada landmark que é identificado na mão pelo modelo MediaPipe.

Figura 6 – Pontos chave da detecção da mão



Fonte: Hand landmarks detection guide (33)

Em seguida, procedeu-se à extração das coordenadas para cada landmark identificado na imagem da mão esquerda e direita. Cada *frame*, analisado pelo Mediapipe, teve seus landmarks identificados, cada um contendo valores x, y e z. Esses valores foram agregados a um numpy array, uma estrutura adequada para o treinamento de uma LSTM.

No âmbito deste projeto, a coleta de dados foi inicialmente realizada utilizando os valores x,y e z, entretanto após alguns testes de treinamento observou-se que os sinais escolhidos não são afetados pelo eixo Z, e para otimizar o tempo de treinamento do modelo e poder computacional necessário, optou-se por excluir o eixo Z da extração. Assim, foram considerados 21 landmarks para cada mão, totalizando 42. Cada landmark possui 2 dimensões, resultando em um numpy array de 84 posições para cada *frame* da imagem. Nos casos em que o mediapipe não identifica a mão no *frame*, o array é preenchido com zeros, seguindo a estratégia de padding

mencionada em (44).

A partir dessa estrutura de extração de dados, foi empregada a técnica de *Data Augmentation*, que consiste na aplicação de transformações nos dados coletados para gerar conjuntos adicionais de dados destinados ao treinamento. No contexto deste projeto, foram definidas quatro funções de extração transformadas: 1) Adição de 0.01 em X e Y; 2) Adição de 0.02 em X e Y; 3) Subtração de 0.01 em X e Y; 4) Subtração de 0.02 em X e Y. Os valores escolhidos para aplicar variação são pequenos para evitar interferências na execução do sinal como um todo. Essas transformações aplicam mudanças sutis em cada um dos eixos dos landmarks, gerando assim novos conjuntos de dados. Para cada *frame* coletado, são gerados mais quatro amostras, simplificando o processo de coleta de dados e proporcionando robustez e variabilidade ao modelo a ser treinado.

Com a estrutura de extração de *keypoints* para cada *frame* pronta, o próximo passo foi determinar a quantidade de *frames* para a coleta de dados. Essa decisão baseou-se nos 9 sinais propostos no projeto, sendo crucial escolher um valor que abrangesse a execução do sinal mais longo, uma vez que é necessário um número fixo de *frames* para o treinamento. Considerando que o sinal mais extenso em tempo de execução é o "Ola Bill", que, quando executado lentamente, tem uma duração média de 9 segundos, e levando em conta que a câmera utilizada possui uma taxa de 10 *frames* por segundo, resulta em um intervalo de 90 *frames* capturados a cada execução do sinal. A câmera em questão utilizada foi a câmera traseira do celular Xiaomi Redmi 9, 13 *mega pixels*.

Entretanto, testes realizados durante o desenvolvimento evidenciaram que esse valor gera ruído para os sinais executados em menor quantidade de *frames*, uma vez que os *frames* restantes são preenchidos com zeros, conforme a estratégia de padding definida anteriormente, afetando consideravelmente a acurácia do modelo. Dessa forma, optou-se por acelerar a execução do sinal "Ola Bill" para reduzir o ruído para os demais sinais. O valor final escolhido foi de 50 *frames* por captura. Além de facilitar o aprendizado dos sinais durante o treinamento, essa escolha demanda menos poder computacional e, conseqüentemente, reduz o tempo necessário para o treinamento do modelo.

Quanto à quantidade de vezes que cada sinal foi executado durante a coleta de dados, foram obtidas 100 amostras por sinal. A aplicação da técnica de *data augmentation* resultou na geração de mais 400 amostras por sinal, totalizando 500 amostras coletadas para cada sinal.

Posteriormente, uma estrutura de pastas foi criada para armazenar os dados registrados. Cada sinal possui uma pasta própria, totalizando 9 pastas. Cada pasta de sinal contém 500 subpastas, onde esse número é a quantidade de amostras. Dentro de cada subpasta de amostra, há 50 arquivos npy, sendo esse o número de arquivos que armazenam o numpy array referente ao *frame* analisado.

Por fim, foi desenvolvida uma interface para guiar o processo de coleta de dados. O código da coleta ativa a câmera em tempo real, faz a leitura do *frames*, exibe qual sinal deve ser realizado juntamente com o número da amostra e por fim usa o *medipipe* para extrair os landmarks da imagem. Essa interface é exibida por 5 segundos (50 *frames*), que é o tempo destinado à coleta. Em cada *frame* um arquivo npy é salvo com as coordenadas dos landmarks identificados no *frame* atual, logo cada execução de sinal salva 50 arquivos numpy com as coordenadas dos landmarks identificados em cada *frame*. Após o salvar os arquivos da execução, foi programado um intervalo de 3 segundos para congelar a câmera, proporcionando um tempo para a pessoa se preparar antes de repetir a execução do sinal. Essa rotina possibilitou a extração de todos os dados necessários para o subsequente treinamento dos 9 sinais selecionados.

Referente a forma de execução dos sinais para a coleta, inicialmente foi optado por executar todos os sinais com a mão direita, entretanto buscando melhorar a acurácia do modelo foi optado por aplicar algumas limitações para facilitar a aprendizagem de cada sinal durante o treinamento. Foi observado que alguns sinais possuíam certas semelhanças em sua execução e para que o modelo possa diferenciar os sinais que são parecidos é necessário mais dados de treinamento, o que leva para a limitação de poder computacional para realizar o treinamento. Dessa forma foram definidas as seguintes configurações: Olá bill executado com a mão direita, siga-me com molde de destro, treinar pessoa com molde de destro, reconhecer pessoa com a mão esquerda, reconhecer objeto com ambas as mãos, pare com molde de destro, sim com a mão direita e não com a mão esquerda. O molde de destro citado, significa que o sinal realizado usa as duas mãos mas a mão direita vai a frente da esquerda durante a execução do sinal.

Em sua maioria os sinais foram realizados no molde de destro, entretanto os sinais treinar pessoa e reconhecer pessoa se realizados ambos como destro tem seu início exatamente igual, então foi optado por realizá-los com orientações opostas em prol de melhorar a acurácia. O mesmo ocorre para o sim e o não que possuem similaridades em sua execução e também foram optados por realizar de forma oposta. Em relação a localização na imagem, se considerarmos

um plano cartesiano os sinais estão dispostos aproximadamente da seguinte forma: Ola bill - 1 e 2 quadrante, siga-me - 3 quadrante, treinar pessoa - todos os quadrante, reconhecer objeto - 3 e 4 quadrante, pare - no centro, sim - 1 quadrante, não - 3 quadrante. A execução dos sinais em diferentes posições traria uma maior robustez ao modelo e o aproximaria da realidade em relação ao contexto dos deficientes auditivos e sua utilização de Libras, entretanto para tal seria necessário a gravação de mais dados em diferentes posições, consequentemente aumentando o tempo de treinamento e poder computacional, o que justificou a escolha pelas simplificações percorridas nesse capítulo.

Dessa forma foram criados dois datasets para testes. O primeiro contemplando 500 amostras, 90 *frames* cada e cada *frame* extraíndo as coordenadas X,Y,Z de cada um dos landmarks identificados, com execução do sinal apenas a direita. O segundo dataset, simplificado, foi necessário regravar as amostras extraíndo apenas 50 *frames*, onde cada landmark possui agora apenas as coordenadas X e Y e com execução alternada para alguns sinais em prol de simplificar o treinamento e reduzir o custo computacional. Foram gravadas 100 amostras e utilizando a técnica de data augmentation foi gerado mais 400 amostras, totalizando em 500 amostras igualmente ao primeiro dataset gravado.

4.2 ETAPA DE PRÉ-PROCESSAMENTO

A etapa de pré processamento consiste em mapear os rótulos(*labels*) e features de cada um dos sinais para o treinamento. Isso significa atribuir um número para cada um dos sinais, como um dicionário e após isso iterar por todos os dados gravados na etapa anterior para relacionar com o número definido. Primeiro se percorre por cada uma das ações, depois por cada um dos seus vídeos e por último por cada *frame*, de forma a ler todos os dados de cada sinal e juntar isso em um único array. Isso significa que é definido um array com os dados de X vídeos cada um com Y *frames* e cada um com K *keypoints*. Nesse processo iterativo é criado um array para as *labels* que terá os valores que definimos previamente mas agora definido para a quantidade de dados de cada um dos sinais, então esse array de *labels* terá (quantidadeDeVideos,quantidadeDeSinais). Por fim foi necessário transformar o array de dados em um *numpyArray* para facilitar a sua manipulação, enquanto o array de *labels* tiveram seus valores convertidos para binário, formato esperado para o treinamento. Esses arrays transformados respectivamente serão o X e Y durante a próxima etapa.

4.3 CRIAÇÃO DA REDE E TREINAMENTO

A fase de treinamento do modelo teve início com a construção de uma rede neural recorrente LSTM utilizando a biblioteca TensorFlow. O estado da arte em relação a problemas de reconhecimento de sinais emprega em sua maioria a utilização 3DCNN e LSTM em conjunto. Contudo, devido ao escopo específico deste projeto e às limitações de poder computacional para o treinamento do modelo, optou-se por utilizar uma abordagem com a combinação do Mediapipe e LSTM. A vantagem é que ela requer menos dados de treinamento para produzir um modelo com boa acurácia e é mais rápida no treinamento, uma vez que a LSTM possui cerca de 500 mil parâmetros, em comparação com os aproximadamente 3 milhões de parâmetros de uma 3DCNN. Isso resulta em uma eficiência notável no processo de treinamento da rede.

A arquitetura usando LSTM também implica em vantagens durante o uso em tempo real, proporcionando maior agilidade em comparação com a 3DCNN. Além disso, a exigência de menos poder computacional para a execução em tempo real torna a abordagem mais viável em ambientes com recursos limitados. Essa escolha estratégica contribui para um desempenho eficaz do modelo, atendendo às necessidades específicas deste projeto.

Inicialmente, foi essencial estabelecer a arquitetura da rede de maneira alinhada com a natureza do problema abordado no projeto. Assim, optou-se por criar um modelo sequencial, permitindo a adição de diversas camadas de maneira sequencial dentro da rede. A construção dessa arquitetura foi um processo experimental que envolveu vários testes com diferentes configurações para identificar aquelas que produziam os melhores resultados.

Num primeiro momento, foram realizados testes de treinamento considerando o primeiro dataset gravado, que contempla as coordenadas X, Y e Z dos landmarks, juntamente com 90 *frames* coletados em cada amostra. Com essas especificações, foi desenvolvida uma rede com as configurações exibidas na figura 7

As camadas de LSTMs são especialmente relevantes para lidar com sequências temporais, uma vez que essas redes são capazes de capturar dependências temporais em dados, o que é crucial ao lidar com movimentos de sinais Libras. A rede possui várias camadas LSTM, cada uma com um número específico de unidades, seguidas por camadas densas para aprendizado mais profundo e complexo das representações. Os das unidades de cada uma das layers foram escolhidos empiricamente após várias tentativas e ajustes. A ativação ReLU em cada camada

Figura 7 – Arquitetura da rede neural

```

model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(50, 84)))
model.add(LSTM(256, return_sequences=True, activation='relu'))
model.add(LSTM(128, return_sequences=False, activation='relu'))
model.add(Dense(1024, activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(len(actions), activation='softmax'))

```

Fonte: Autoria própria

introduz não linearidade, enquanto a camada de saída utiliza a ativação softmax para produzir probabilidades normalizadas para cada classe de sinal Libras.

A utilização de *return_sequences=True* nas primeiras duas camadas LSTM indica que essas camadas estão configuradas para retornar sequências temporais completas, ou seja, as saídas de todas as etapas temporais são transmitidas para as próximas camadas LSTM. Isso é útil quando as camadas seguintes precisam de informações de todas as etapas temporais para realizar uma representação temporalmente rica.

Por outro lado, a terceira camada LSTM com *return_sequences=True* significa que essa camada está configurada para retornar apenas a saída correspondente à última etapa temporal. Isso é comum na última camada LSTM em uma pilha, pois ela geralmente é seguida por camadas densas ou de saída que não requerem sequências temporais, mas sim uma representação agregada da sequência temporal.

Essa configuração passou por um treinamento utilizando o conjunto de dados completo coletado, levando cerca de 5 horas para finalizar. No entanto, as métricas obtidas não atenderam às expectativas do projeto, indicando que, para esse cenário, seria necessário um volume maior de dados de treinamento. Essa demanda geraria um custo computacional para o treinamento além do viável para o escopo do projeto. Observando essas questões foi optado por simplificar alguns aspectos do modelo de forma a conseguir uma melhor acurácia dentro do possível considerando aspectos de poder computacional. A partir dessa observação foi tomada a decisão de refazer o treinamento utilizando o segundo dataset coletado que contempla apenas os eixos X e Y dos landmarks detectados e a redução para 50 *frames* coletados por amostra.

Com a simplificação dos dados de treinamento, foi repetido o treinamento com a mesma arquitetura de rede vista na figura 7 dessa vez observando uma melhora na acurácia,

atingindo convergência mais rápido e um tempo de treinamento relativamente baixo, cerca de 1 hora e 20 minutos, validando a ideia de simplificar os dados em prol de um menor custo computacional sem sacrificar a acurácia.

A solução também contempla funções de otimização e perda da rede neural. O otimizador refere-se ao algoritmo usado para ajustar os pesos da rede neural durante o treinamento, com o objetivo de minimizar a função de perda. A otimização é um aspecto fundamental do treinamento de redes neurais, e existem vários algoritmos de otimização disponíveis. 'Adam' foi o escolhido inicialmente para realizações de testes por ser um otimizador popular e amplamente usados para treinar redes neurais, entretanto foi optado por utilizar o 'Nadam'. O otimizador Nadam (Nesterov-accelerated Adaptive Moment Estimation) é uma variação do otimizador Adam (Adaptive Moment Estimation) que combina as ideias do otimizador Nesterov Accelerated Gradient (NAG) e do Adam. Ele incorpora a correção de viés de momento de Nesterov ao algoritmo Adam, buscando melhorar o desempenho em algumas situações.

Em seguida é necessário definir a função de perda que é uma medida que avalia a diferença entre as previsões da rede neural e os rótulos (ou alvos) reais do conjunto de dados. O objetivo do treinamento é minimizar essa função de perda, ajustando os pesos da rede. Na construção da rede, *categorical_crossentropy* foi especificado como a função de perda. Esta função de perda é comumente usada em problemas de classificação multiclasse, onde cada amostra de dados pertence a uma de várias classes exclusivas.

A entropia cruzada categórica é uma medida que quantifica o quanto as previsões da rede neural se desviam dos rótulos reais. Para além das configurações citadas anteriormente, com intuito de controlar o treinamento foram definidas 2 funções auxiliares, a *ModelCheckpoint* e *EarlyStopping*, que são *callbacks* usados durante o treinamento da rede neural para monitorar o progresso do treinamento e fazer decisões com base em métricas específicas.

O *ModelCheckpoint* é um *callback* que é usado para salvar os pesos do modelo durante o treinamento, particularmente quando uma métrica específica atinge um valor desejado. No caso em questão, o modelo é configurado para salvar os melhores pesos com base na métrica de validação *val_categorical_accuracy*, ou seja, sempre que a acurácia de validação atingir um novo maior valor, um modelo será salvo como o melhor modelo até então. O *EarlyStopping* é um *callback* usado para interromper o treinamento se uma métrica específica não mostrar melhora significativa em um determinado número de épocas. No caso, o treinamento será interrompido

se a métrica de validação *val_categorical_accuracy* não melhorar após 30 épocas. Em resumo, o *ModelCheckpoint* salva os melhores pesos do modelo com base na acurácia categórica de validação, enquanto o *EarlyStopping* interrompe o treinamento se a acurácia de validação não mostrar melhora após um certo número de épocas. Esses *callbacks* são úteis para evitar o superajuste (*overfitting*) e garantir que você tenha o melhor modelo possível.

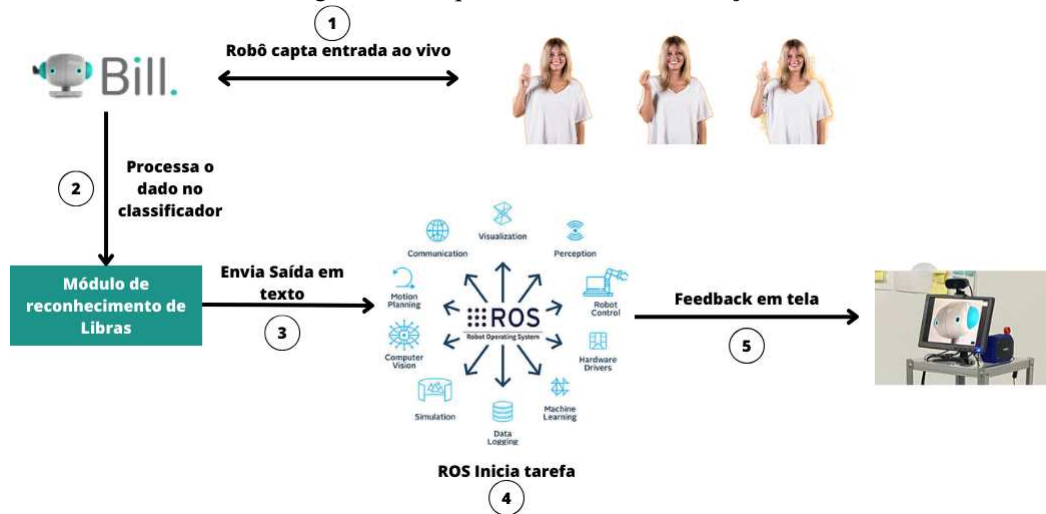
Na configuração da rede neural um ponto importante é a escolha apropriada do número de épocas para que o treinamento seja bem-sucedido. De forma geral essa escolha reflete diretamente em quanto tempo o modelo será treinado. No contexto do treinamento de redes neurais, uma "época"(*epoch*) refere-se a uma única passagem completa por todo o conjunto de treinamento durante o processo de treinamento. Em outras palavras, durante uma época, o modelo neural examina cada exemplo de treinamento uma vez. Alguns pontos importantes relacionados à quantidade de épocas são: Subajuste (*underfitting*): Se o número de épocas for muito baixo, o modelo pode não ter tempo suficiente para aprender padrões complexos nos dados. Isso resultará em um subajuste, onde o modelo não é capaz de fazer previsões precisas nem mesmo nos dados de treinamento. Convergência: A quantidade de épocas necessárias para que um modelo alcance a convergência, ou seja, para que a perda de treinamento pare de diminuir e o desempenho do modelo estabilize, pode variar de um problema para outro. Em alguns casos, o treinamento pode convergir rapidamente, enquanto em outros, pode ser necessário um grande número de épocas. *Overfitting*: Se o número de épocas for muito alto em relação à complexidade do modelo e à quantidade de dados, o modelo pode começar a se ajustar demais aos dados de treinamento. Isso resultará em um *overfitting*, onde o modelo se torna muito específico para os exemplos de treinamento e não generaliza bem para novos dados. A partir da quantidade de dados e arquitetura definida para a rede neural, foi decidido empiricamente o valor de 100 épocas de treinamento. Esse valor foi decidido empiricamente através de testes e consagrou-se um valor adequado para atingir convergência do modelo.

Após a declaração dessas configurações o modelo é compilado antes do treinamento recebendo como parâmetros as funções de otimização, perda e a lista de métricas utilizada para avaliar o modelo durante e após o treinamento. Por fim é utilizado da função *Fit* para iniciar o treinamento do modelo. Essa função recebe como parâmetro os dados usados para treinamento, os dados de validação e todos os callbacks que foram definidos, sendo nesse caso as funções *model_checkpoints*, *early_stopping*, explicadas anteriormente.

4.4 CLASSIFICAÇÃO DE SINAIS

O segundo momento do desenvolvimento do projeto contempla um fluxo de utilização do modelo LSTM e sua integração com o ROS para iniciar funcionalidades através dos comandos reconhecidos em Libras, assim finalizando a interação humano-robô. Dessa forma, foi definido uma arquitetura de fluxo que sumariza tudo que é necessário nesse momento. A figura 8 exibe os passos e a lógica de funcionamento.

Figura 8 – Arquitetura de fluxo - Solução



Fonte: Autoria própria

A construção do algoritmo de classificação consiste em captar o vídeo de entrada em tempo real por parte do robô BILL utilizando uma câmera. Em seguida, o processamento dos dados é realizado no módulo de reconhecimento de sinais já treinado, que emitirá um resultado em texto do que foi reconhecido. Esse resultado em texto é enviado para o sistema central de BILL, o ROS, este que ficará responsável por iniciar alguma atividade a partir do que foi reconhecido, sendo esse o 4 passo, inicialização de uma tarefa. Por último, o quinto passo é exibição de um feedback visual em tela que exibirá o que foi reconhecido pelo módulo de reconhecimento de Sinais e qual funcionalidade do robô será inicializada fechando-se assim uma interação humano-robô por completo.

A solução proposta compreende uma interface gráfica que exibe em tela os resultados do reconhecimento em tempo real em forma de texto e também exibe a porcentagem de confiança do resultado para cada um dos sinais, permitindo que o usuário identifique o resultado do sinal que foi realizado e qual acurácia daquele resultado. A imagem a seguir demonstra a interface desenvolvida.

Figura 9 – Interface gráfica - Solução



Fonte: Autoria própria

Nesse contexto o código da solução inicia-se com a inicialização da câmera em loop para coletar os *frames* da imagem. Em cada *frame* esse dado é analisado pela função do mediapipe landmark detector para realizar a segmentação da mão na imagem e extrair os *keypoints* da mão, salvando-os em um vetor chamado "sequence", definido com intuito de armazenar os *keypoints* da mão pelos últimos 50 *frames* da câmera, sendo este o mesmo valor utilizado durante o treinamento. Ao atingir 50 *frames*, significa que temos uma execução completa de um sinal, então o vetor é analisado pelo modelo treinado e emite um resultado do sinal que foi reconhecido. O resultado contempla valores de confiança, também chamado de *threshold*, para cada um dos sinais treinados, a partir disso o sinal que tiver maior confiança é o resultado utilizado pelo código para exibir em tela. Se o sinal tiver um *threshold* acima ou igual a 85%, o resultado é considerado como válido e é armazenado em um vetor chamado "sentence", para exibir os mesmos lado a lado na interface gráfica até que se atinja o valor de 3 resultados. Nesse momento o vetor é limpo para não poluir a visualização. Na interface gráfica também é exibido a porcentagem de confiança do resultado para todas as classes, dessa forma é possível identificar se o modelo

confunde os resultados de uma classe com outra classe por ambas possuírem alta porcentagem de threshold.

A integração do algoritmo de classificação de sinais Libras ao ambiente ROS, foi necessário criar um pacote ROS específico para a execução do código de classificação. No contexto do ROS (Robot Operating System), um pacote é uma estrutura organizacional que contém todos os arquivos relacionados a um determinado software. Ele é a unidade básica de construção e distribuição de software no ecossistema ROS.

Posteriormente, o código de classificação foi adaptado para operar como um Publisher no ROS. Em termos do ROS, um nó é uma instância em execução de um programa. No caso, o código de classificação foi configurado como um nó do ROS e especificado como um "talker". No ROS, um talker é um nó que envia mensagens (publica) em um tópico específico.

O tópico do ROS é um canal de comunicação que permite a troca de mensagens entre diferentes partes do sistema ROS. Neste contexto, sempre que um sinal Libras é reconhecido, o resultado em texto gerado pelo código de classificação é publicado no tópico escolhido previamente. Essa abordagem possibilita uma comunicação eficaz entre diferentes partes do sistema ROS.

Essa publicação de mensagens é contínua enquanto o nó/código estiver ativo. Assim, cada vez que um sinal é identificado, o resultado em texto correspondente é publicado no tópico associado. Essa exposição constante dos resultados em texto permite que qualquer outro pacote no ROS, devidamente configurado como um "listener" para esse tópico específico, ouça essas mensagens. Essa escuta possibilita a ativação de diversas atividades ou comandos, promovendo, assim, uma interação efetiva entre o humano e o robô no ambiente ROS.

5 VALIDAÇÃO

O objetivo desta validação é avaliar o modelo gerado com propósito de medir a acurácia no que se refere a erros e acertos no contexto dos sinais treinados, usando amostras de vídeos não vistos durante o treinamento e interação em tempo real. Dessa forma a validação proposta no projeto foi realizada em duas etapas: Validação durante o treinamento utilizando uma porcentagem de dados retirado diretamente do dataset de forma a validar a acurácia do modelo e validação em tempo real através de uma câmera captando a execução dos sinais ao vivo e registrando as predições do modelo para analisar sua acurácia nesse cenário.

A validação durante o treinamento sua configuração ainda na etapa de treinamento onde é necessário definir as partições de treinamento e testes, isso significa dividir a % de dados que será utilizado em treinamento e quanto será utilizado para validação do modelo em testes. Foi utilizado o método *K-fold cross-validation* para validação do desempenho do modelo. O uso desse método de validação foi escolhido devido a ajuda para mitigar problemas de superajuste, uma vez que o modelo é avaliado em várias divisões diferentes do conjunto de treinamento. Além disso, ele fornece uma avaliação mais confiável do desempenho do modelo, o que é particularmente importante em problemas de aprendizado de máquina, como as RNNs, onde o desempenho pode variar significativamente dependendo dos dados e hiperparâmetros. Nesse método, o conjunto de treinamento é dividido em K subconjuntos (ou "dobras") aproximadamente do mesmo tamanho. Neste projeto foi escolhido $K=5$, que significa que o conjunto de treinamento será dividido em 5 partes iguais, logo teremos 20% do dataset voltado apenas para validação e 80% voltado para o treinamento. A escolha de usar 5 folds no cross-validation é uma decisão que equilibra eficiência de uso de dados, robustez na avaliação do modelo e aplicabilidade geral. Ao dividir o conjunto de dados em cinco subconjuntos, podemos eficientemente avaliar o desempenho do modelo em diferentes partições, garantindo estimativas robustas do desempenho e um equilíbrio prático entre a capacidade computacional e a necessidade de obter *insights* confiáveis sobre a capacidade preditiva do modelo, como exibido em (45).

O treinamento e a avaliação da RNN são repetidos K vezes. A cada iteração (ou dobra), um dos K subconjuntos é usado como conjunto de validação, enquanto os outros K-1

subconjuntos são usados para treinamento. Portanto, o modelo é treinado K vezes, cada vez com um subconjunto diferente como conjunto de validação. Durante cada iteração do K-fold cross-validation, é possível calcular métricas de avaliação, como erro médio ou acurácia, para o modelo em relação ao conjunto de validação.

Essas métricas ajudaram a avaliar o desempenho do modelo em diferentes conjuntos de dados de validação. Após concluir todas as iterações, calcula-se a média das métricas de avaliação obtidas em cada dobra. Essa média fornece uma estimativa mais robusta do desempenho do modelo do que apenas usar um único conjunto de validação.

O processo de *cross-validation* pode ser usado para ajustar hiperparâmetros, como a taxa de aprendizado, o número de camadas da RNN, o tamanho da janela temporal, etc. Isso ajuda a encontrar a configuração de hiperparâmetros que produz o melhor desempenho médio em várias divisões do conjunto de treinamento.

A segunda forma de validação do projeto consistiu em testes com uma especialista em Libras, com o objetivo de validar a acurácia do robô BILL na predição de sinais em tempo real. Esses testes foram planejados e serão executados por uma professora de Libras na Universidade do Estado da Bahia operando diretamente o software desenvolvido.

A rotina de testes contempla uma avaliação de todos os sinais treinados, 5 vezes cada. Os sinais serão testados na seguinte ordem: 'Ola bill', 'siga-me', 'treinar pessoa', 'reconhecer pessoa', 'reconhecer objeto', 'pare', 'sim', 'nao'. A interprete se posicionará na frente da câmera a cerca de 1 metro de distância e fará a execução de cada sinal em uma janela de 5 segundos. A velocidade pode ser variada desde que não exceda o tempo de 5 segundos, visto que o modelo foi treinado até esse limite. Após a execução é necessário aguardar a exibição de "Vazio" na interface demonstrando que o modelo não está reconhecendo nenhum sinal no momento, a partir desse momento é possível repetir as outras tentativas, no sentido que interprete possa se reposicionar e realizar novamente o sinal. Após a exibição do resultado em tela o mesmo será enviado para o ROS e por fim será registrado em um arquivo qual sinal foi realizado e qual foi o resultado obtido naquela vez. Com a finalização da rotina de testes será realizado uma análise para calcular a média de acerto do modelo, seu desvio padrão e média de acerto para cada classe de forma a entendermos melhor a performance do modelo treinado e identificar pontos de melhora.

Com base em estudos do estado da arte, foi estabelecido um critério de aceitação

relacionado à porcentagem de acurácia gerada pelo modelo tanto no treinamento quanto em tempo real. Ao analisar os resultados em (6) é possível observar que as validações utilizando vídeos não vistos apresentam uma acurácia média igual ou superior a 90%. Portanto, esse valor foi selecionado como critério de aceitação para a validação do treinamento com vídeos. Por outro lado, em relação à validação em tempo real, os resultados em (46) indicam que as acurácias possuem uma média de 80%. Sendo assim, esse valor foi adotado como critério de aceitação para a validação em tempo real.

Com a execução dos métodos de validação foi possível coletar os resultados referentes a performance do modelo durante o treinamento, usando 20% do dataset e sua performance em tempo real utilizando uma câmera.

5.1 RESULTADOS VALIDAÇÃO DURANTE TREINAMENTO

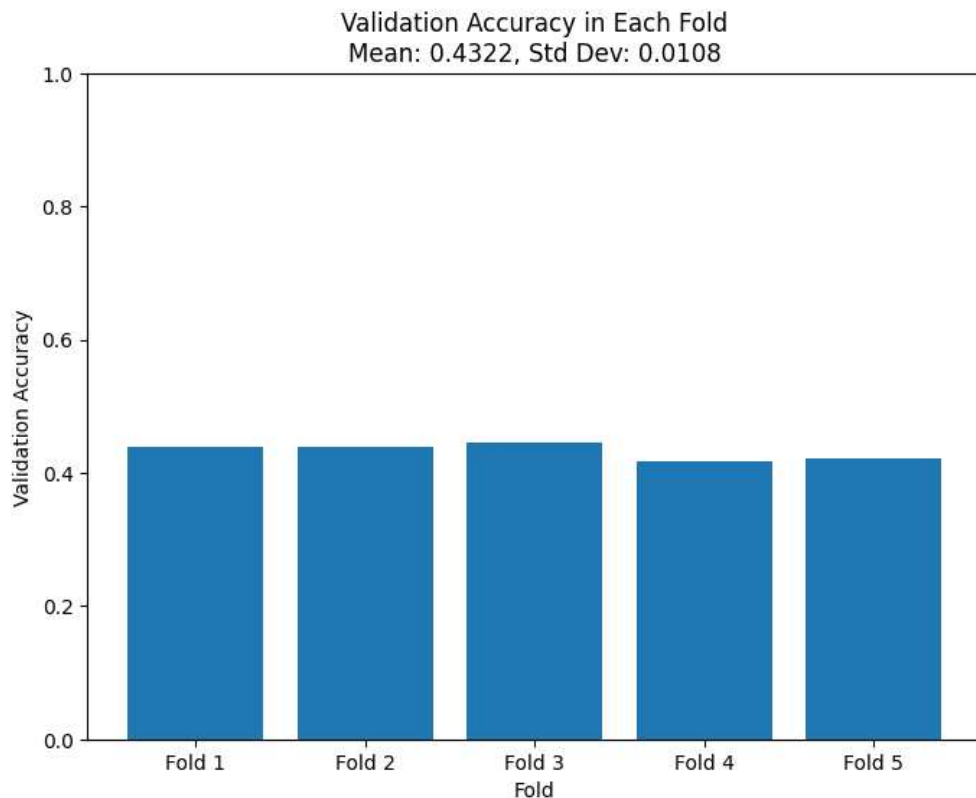
Conforme abordado no capítulo anterior, o teste inicial englobou um modelo aplicado ao primeiro conjunto de dados gravado, que inclui coordenadas X, Y e Z, com 90 frames por amostra e um total de 500 amostras por sinal. Ao empregar a arquitetura destacada na Figura 8, os resultados foram refletidos no gráfico apresentado na figuras subsequente:

O gráfico 1 ilustra os resultados adquiridos por meio da aplicação de 5 folds na técnica de cross-validation. Esse procedimento implica que o treinamento foi conduzido e validado em cinco iterações distintas. O gráfico apresenta a média de acurácia e seu desvio padrão, considerando os resultados gerados em cada iteração. Vale ressaltar que 80% do dataset foi destinado ao treinamento, enquanto os restantes 20% foram reservados para a fase de validação.

Nesse treinamento inicial, o gráfico 1 apresentado, foi construído utilizando o dataset completo, que consiste em 500 amostras para cada sinal proposto, 90 frames por amostra e coordenadas X,Y e Z para cada landmark identificado. Com essa configuração, o modelo alcançou uma média de acurácia de 43,22%, com um desvio padrão de 0,0109.

No segundo treinamento, foi utilizado o dataset simplificado que também é composto por 500 amostras por sinal entretanto cada uma contendo 50 frames e cada landmark com coordenadas X e Y. Com a redução de 40 frames e retirada do eixo z, foi possível obter os seguintes resultados:

Gráfico 1 - Resultados parciais primeira etapa validação



Fonte: Autoria própria

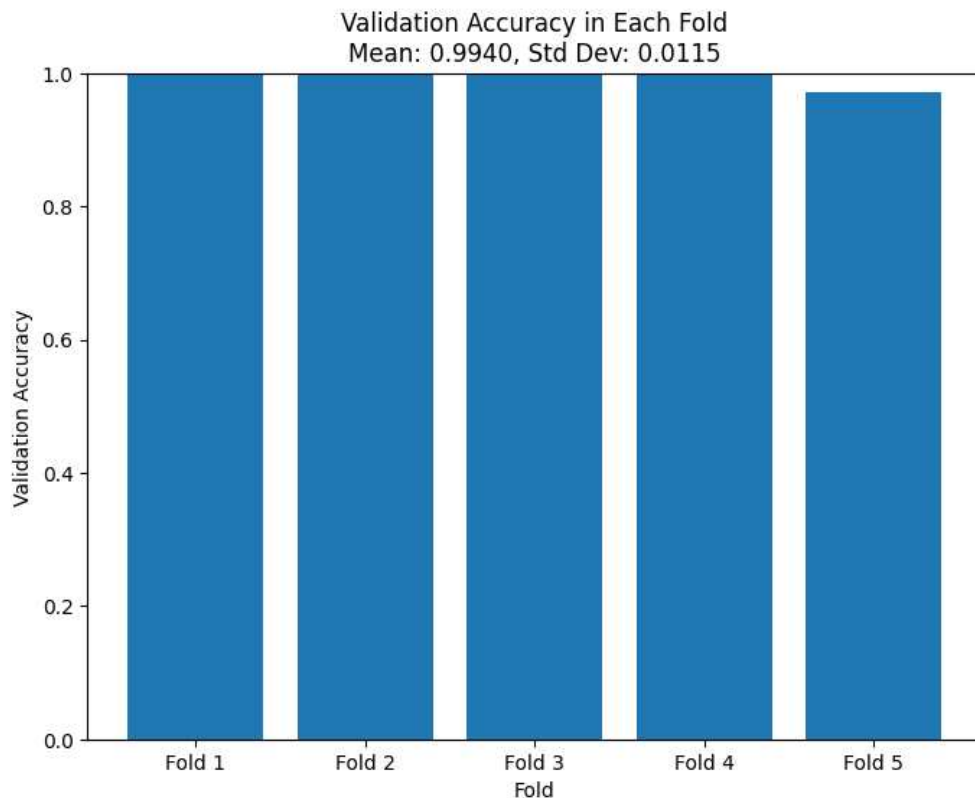
Com a configuração simplificada o modelo obteve cerca de 99% de acurácia média e 0.0115 de desvio padrão na validação durante o treinamento.

5.2 RESULTADOS VALIDAÇÃO EM TEMPO REAL

Na segunda fase de validação do projeto, foi implementada uma série de testes em tempo real por meio de uma câmera. Essa etapa utilizou o segundo modelo treinado, escolhido por apresentar a maior acurácia durante a validação do treinamento. A execução dos testes foi conduzida por um intérprete de Libras, repetindo cada sinal em 5 ocasiões. Os resultados desses testes foram registrados e são apresentados na imagem a seguir.

A tabela apresenta as seguintes taxas de acerto para cada classe: "Olá Bill- 20%, "Siga-me- 100%, "Treinar Pessoa- 60%, "Reconhecer Pessoa- 100%, "Reconhecer Objeto- 100%, "Pare- 100%, "Sim- 100%, "Não- 100%. Analisando os resultados globais, o modelo registrou 34 acertos e 6 erros em 40 execuções, resultando em uma acurácia de 85% e uma taxa de erro de 15%.

Gráfico 2 - Resultado treinamento dataset simplificado



Fonte: Autoria própria

5.2.1 Discussão

A etapa de validação durante o treinamento inicial, utilizando um dataset mais abrangente, produziu uma acurácia de 43,22% com um desvio padrão de 0,0108, considerando um conjunto de 500 vídeos por sinal. Esses resultados indicam uma convergência estável do modelo ao longo do treinamento, sugerindo uma configuração apropriada da rede neural para lidar com a complexidade do conjunto de dados utilizado.

Quando um modelo converge, esse fenômeno denota que as atualizações nos pesos estão diminuindo gradualmente, sinalizando que o algoritmo está se aproximando de uma solução ótima ou, no mínimo, de um mínimo local na função de perda. Em termos mais específicos, o modelo está efetivamente aprendendo com os dados de treinamento, e as alterações nos pesos estão se reduzindo à medida que o treinamento avança, indicando que o modelo absorveu o conhecimento disponível nos dados de treinamento. O processo desse treinamento inicial foi concluído em aproximadamente 5 horas.

Contudo, para aprimorar a acurácia mantendo a mesma configuração de 90 frames

Figura 10 – Resultados em tempo real por classe

Resultados por classe		
	ACERTOS	ERROS
OLA BILL	1	4
SIGA-ME	5	0
TREINAR PESSOA	3	2
RECONHECER PESSOA	5	0
RECONHECER OBJETO	5	0
PARE	5	0
SIM	5	0
NÃO	5	0

Fonte: Autoria própria

por amostra e coordenada contendo eixos X,Y,Z por landmark, seria necessário expandir a quantidade de dados de treinamento, o que acarretaria em custos computacionais mais elevados e um tempo de treinamento proporcionalmente maior. Adicionalmente, seria necessário ajustar a arquitetura da rede neural para se adequar à nova quantidade de dados. Diante desses desafios, essa abordagem foi inviabilizada devido às restrições temporais e à limitação de recursos computacionais.

No segundo treinamento, utilizando o dataset simplificado, o modelo foi novamente treinado demandando aproximadamente 1 hora e 20 minutos para ser concluído. Os resultados

revelaram uma média de acurácia de 99% e um desvio padrão de 0,0115. Esses achados demonstraram estabilidade nos resultados de cada Fold, indicando que o treinamento do modelo convergiu. Comparando ambos os treinamentos realizados, observou-se um aumento significativo de cerca de 55% na acurácia média do modelo.

A exclusão do eixo Z eliminou a necessidade de aprender o fator distância durante o treinamento, simplificando o processo de aprendizagem da rede. Além disso, a redução em 40 frames tornou o treinamento mais eficiente, o que justifica o aumento da acurácia observado.

A validação em tempo real utilizando o melhor modelo da validação durante o treinamento atingiu aproximadamente 85% de acurácia considerando todos os sinais. A validação em tempo real é mais complexa em comparação com a de treinamento, devido à falta de controle sobre o momento de início e fim da captação dos dados durante a execução do sinal. Isso adiciona ruído aos dados, podendo afetar a acurácia dos resultados. Além disso, o sinal pode ser executado em posições diferentes das treinadas, introduzindo um grau de dificuldade nas previsões.

Alguns sinais apresentaram 100% de acerto, no entanto, os sinais "Olá, Bill" e "Treinar pessoa" obtiveram acurácia abaixo do esperado. Entre todos os sinais, esses dois podem ser considerados os mais complexos devido à sua duração e maior movimentação, o que aumenta a dificuldade das previsões. Observou-se que esses sinais são reconhecidos corretamente apenas se executados em posições específicas da imagem, indicando que o modelo não consegue prever corretamente se a execução ocorrer em posições diferentes das treinadas. Essa observação aponta para uma coleta de dados com pouca variabilidade na execução dos sinais em termos de posicionamento na imagem, tornando o modelo menos robusto para uso em tempo real.

De acordo com o critério de aceite definido, a validação durante o treinamento utilizando 20% do dataset superou em 9% a meta de 90% estabelecida. Já a validação em tempo real superou em 5% a meta de 80% estabelecida. Esses valores são indicadores positivos que demonstram a viabilidade do modelo para utilização na interação humano-robô.

6 CONSIDERAÇÕES FINAIS

O desenvolvimento deste projeto centrou-se na criação de uma interação humano-robô por meio do reconhecimento de comandos em Libras, proporcionando uma comunicação acessível e natural entre indivíduos com deficiência auditiva e o ambiente tecnológico do robô de serviço.

O projeto incluiu o treinamento de 8 sinais de Libras e 1 sinal auxiliar, todos dinâmicos, executados ao longo de 5 segundos. Inicialmente foi realizada uma validação durante o treinamento, utilizando um dataset mais complexo que resultou em uma acurácia de 43,22% e um desvio padrão de 0,0108 para um conjunto de 500 vídeos por sinal. Apesar de ser um resultado estável considerando o baixo desvio padrão dentre os resultados de cada um dos folds, seria necessário aumentar a quantidade de dados de treinamento para atingir uma maior acurácia, o que inviabilizaria o projeto devido falta de poder computacional.

A decisão tomada foi empregar um dataset simplificado, sem a utilização do eixo Z e com redução na quantidade de frames por execução do sinal, mantendo, no entanto, a mesma quantidade de amostras (500 por sinal). O treinamento foi novamente conduzido com essa nova configuração alcançando uma média de acurácia de 99% e desvio padrão de 0,0115. Constatou-se um aumento de cerca de 55% na acurácia média do modelo, validando a estratégia de simplificação para aprimoramento da acurácia.

Utilizando o modelo com a maior acurácia, realizou-se a validação em tempo real com uma intérprete de Libras. Nesse cenário, o modelo atingiu aproximadamente 85% de acurácia considerando todos os sinais. Finalmente, os resultados em tempo real são emitidos em texto e encaminhados para o ROS, permitindo que o BILL inicialize suas funcionalidades. O modelo superou os critérios de aceite tanto para a validação durante o treinamento, quanto para a validação em tempo real, logo, à luz dos resultados apresentados, é possível validar a hipótese do projeto, afirmando que o reconhecimento de comandos em Libras pode viabilizar a interação entre deficientes auditivos e robôs de serviço por meio do algoritmo LSTM e do ROS. A partir disso também é possível responder ao problema da pesquisa e afirmar que robôs de serviço podem ser inclusivos a partir do desenvolvimento de *features* que englobem diferentes tipos de

deficiência. No caso desse projeto o reconhecimento de Libras através de visão computacional pôde tornar inclusivo a interação humano-robô para com deficientes auditivos.

É importante ressaltar as limitações do projeto, especialmente em relação ao poder computacional, o que levou à necessidade de aplicar restrições na configuração dos dados durante a coleta, como a redução da quantidade de frames, exclusão do eixo Z e baixa variabilidade nas posições de execução dos sinais. Essas adaptações foram cruciais para obter uma acurácia considerável que permitisse a interação humano-robô em tempo real. A escolha do LSTM foi motivada, em grande parte, por sua menor complexidade e custo computacional reduzido, tornando-se uma opção viável para aqueles que enfrentam limitações de poder computacional. Seu treinamento é relativamente rápido em comparação com outras opções de ponta, e, com uma estratégia bem definida para a coleta de dados, é possível criar modelos com acurácias elevadas adequadas para diversos casos de uso.

Este trabalho possui relevância no uso de Libras para comunicação entre robôs de serviço e seres humanos, podendo inspirar futuros trabalhos nessa mesma direção, impactando positivamente na qualidade de vida das pessoas com deficiência auditiva que podem acessar tecnologias como o robô de serviço proposto neste projeto. A metodologia empregada pode contribuir para o desenvolvimento de tecnologias similares aplicadas a outros dispositivos, como celulares, computadores, tablets, entre outros. Essa abordagem promove um acesso mais igualitário a informações, serviços e recursos, frequentemente limitados ou inacessíveis devido à barreira da comunicação. Assim, este trabalho contribui para uma maior inclusão social, reduzindo o isolamento e permitindo a participação plena no uso de tecnologias.

Ao considerar futuras pesquisas, a adição de mais sinais de Libras ao modelo pode ser explorada. Nesse contexto, a inclusão do eixo Z nas coordenadas dos landmarks, emitidos pelo MediaPipe durante a coleta de dados, torna-se necessária, especialmente para sinais que são influenciados pela distância. Outros aspectos importantes incluem a consideração de sinais executados com a mão esquerda, direita, molde de canhoto e molde de destro, bem como variações mais amplas no posicionamento dos sinais, abrangendo diferentes quadrantes e distâncias de forma que o modelo torne-se mais robusto e adequado para uso no dia a dia.

Além disso, a implementação do projeto pode ser expandida para dispositivos acessíveis, como celulares e tablets, equipados com câmeras, permitindo o uso em diversos ambientes. Adicionalmente, o projeto pode ser integrado a sistemas de assistentes virtuais embarcados em

dispositivos, oferecendo suas funcionalidades para pessoas com deficiência auditiva.

Entre as contribuições significativas deste trabalho, destacam-se: o desenvolvimento de um algoritmo para coleta de dados em tempo real, permitindo a construção de um dataset a partir do zero; a utilização do MediaPipe para a extração de dados; a construção de uma rede neural recorrente utilizando TensorFlow, com discussão detalhada sobre seus parâmetros de treinamento; a implementação de um algoritmo de classificação de sinais com um modelo treinado e uma interface gráfica para exibir resultados e probabilidades/confiança da predição; e a integração do algoritmo de classificação com o ROS, possibilitando que pessoas com deficiência auditiva usufruam das funcionalidades de um robô de serviço.

O estudo não apenas demonstra como desenvolver um modelo de classificação do zero, desde a gravação de dados até o treinamento, mas também discute técnicas para simplificar o processo, tornando-o acessível para quem possui baixo poder computacional. As abordagens e procedimentos delineados neste trabalho podem servir como base para outros estudos, contribuindo para a expansão do conhecimento sobre o reconhecimento de gestos e linguagem de sinais. A pesquisa resultante desse projeto possui o potencial de fornecer insights valiosos sobre métodos e desempenho do reconhecimento de Libras em contextos de interação com robôs, contribuindo para o desenvolvimento contínuo e aprimoramento nessa área.

REFERÊNCIAS

- 1 BELANCHE, D.; CASALÓ, L. V.; FLAVIÃO, C.; SCHEPERS, J. Service robot implementation: a theoretical framework and research agenda. **The Service Industries Journal**, Routledge, v. 40, n. 3-4, p. 203–225, 2020.
- 2 SONG, C. S.; KIM, Y.-K. The role of the human-robot interaction in consumers' acceptance of humanoid retail service robots. **Journal of Business Research**, v. 146, p. 489–503, jul. 2022. ISSN 0148-2963.
- 3 SOUZA, J. R. de; GUIMARAES, D.; PIMENTEL, F.; SAPUCAIA, F.; LARANJEIRA, C.; JESUS, L.; SIMOES, M.; FRIAS, D. **BahiaRT@ Home 2016 Team Description Paper**. [S.l.], 2015.
- 4 ACSO | Main / LinhaPesquisa. <<https://www.acso.uneb.br/acso/index.php?n=Main.LinhaPesquisa>>. (Accessed on 12/15/2023).
- 5 RAMOS, C. R. LIBRAS: A Língua de Sinais dos Surdos Brasileiros.
- 6 CASTRO, G. Z. de; GUERRA, R. R.; GUIMARÃES, F. G. Automatic translation of sign language with multi-stream 3D CNN and generation of artificial depth maps. **Expert Systems with Applications**, v. 215, p. 119394, abr. 2023. ISSN 0957-4174.
- 7 DONG, Y.; LIU, J.; YAN, W. Dynamic Hand Gesture Recognition Based on Signals From Specialized Data Glove and Deep Learning Algorithms. **IEEE Transactions on Instrumentation and Measurement**, v. 70, p. 1–14, 2021. ISSN 1557-9662. Conference Name: IEEE Transactions on Instrumentation and Measurement.
- 8 LI, Z.; LI, K.; WEI, N. A sEMG-Based Hand Gesture Recognition Using Multichannel CNN and MLP. In: **2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)**. [S.l.: s.n.], 2020. p. 867–871.
- 9 PERAL, M.; SANFELIU, A.; GARRELL, A. Efficient Hand Gesture Recognition for Human-Robot Interaction. **IEEE Robotics and Automation Letters**, v. 7, n. 4, p. 10272–10279, out. 2022. ISSN 2377-3766. Conference Name: IEEE Robotics and Automation Letters.
- 10 SADEDDINE, K.; CHELALI, F. Z.; DJERADI, R.; DJERADI, A.; BENABDERRAHMANE, S. Recognition of user-dependent and independent static hand gestures: Application to sign language. **Journal of Visual Communication and Image Representation**, v. 79, p. 103193, ago. 2021. ISSN 1047-3203.
- 11 SINGH, D. K. 3D-CNN based Dynamic Gesture Recognition for Indian Sign Language Modeling. **Procedia Computer Science**, v. 189, p. 76–83, jan. 2021. ISSN 1877-0509.
- 12 SUBBURAJ, S.; MURUGAVALLI, S. Survey on sign language recognition in context of vision-based and deep learning. **Measurement: Sensors**, v. 23, p. 100385, out. 2022. ISSN 2665-9174.

- 13 TANG, X.; YAN, Z.; PENG, J.; HAO, B.; WANG, H.; LI, J. Selective spatiotemporal features learning for dynamic gesture recognition. **Expert Systems with Applications**, v. 169, p. 114499, maio 2021. ISSN 0957-4174.
- 14 ZHANG, X.; TIE, Y.; QI, L. Dynamic Gesture Recognition Based on 3D Separable Convolutional LSTM Networks. In: **2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)**. [S.l.: s.n.], 2020. p. 180–183. ISSN: 2327-0594.
- 15 SUNDAR, B.; BAGYAMMAL, T. American sign language recognition for alphabets using mediapipe and lstm. **Procedia Computer Science**, v. 215, p. 642–651, 2022. ISSN 1877-0509. 4th International Conference on Innovative Data Communication Technology and Application.
- 16 VERMA, B. A two stream convolutional neural network with bi-directional GRU model to classify dynamic hand gesture. **Journal of Visual Communication and Image Representation**, v. 87, p. 103554, ago. 2022. ISSN 1047-3203.
- 17 SUNDAR, B.; BAGYAMMAL, T. American Sign Language Recognition for Alphabets Using MediaPipe and LSTM. **Procedia Computer Science**, v. 215, p. 642–651, jan. 2022. ISSN 1877-0509.
- 18 ALASHHAB, S.; GALLEGO, A. J.; LOZANO, M. Efficient gesture recognition for the assistance of visually impaired people using multi-head neural networks. **Engineering Applications of Artificial Intelligence**, v. 114, p. 105188, set. 2022. ISSN 0952-1976.
- 19 MA, D.; HIROTA, K.; DAI, Y.; JIA, Z. Dynamic Sign Language Recognition Based on Improved Residual-LSTM Network. 2021.
- 20 BELANCHE, D.; CASALÓ, L. V.; FLAVIÁN, C.; SCHEPERS, J. Service robot implementation: a theoretical framework and research agenda. **The Service Industries Journal**, v. 40, n. 3-4, p. 203–225, mar. 2020. ISSN 0264-2069.
- 21 WAN, S.; GU, Z.; NI, Q. Cognitive computing and wireless communications on the edge for healthcare service robots. **Computer Communications**, v. 149, p. 99–106, jan. 2020. ISSN 0140-3664.
- 22 HAYASHI, K.; ZHENG, W.; HAFI, L. E.; HAGIWARA, Y.; TANIGUCHI, T. Bidirectional Generation of Object Images and Positions using Deep Generative Models for Service Robotics Applications. In: **2021 IEEE/SICE International Symposium on System Integration (SII)**. [S.l.: s.n.], 2021. p. 325–329. ISSN: 2474-2325.
- 23 SCHNEIDER, S.; CLAS, W.; BRŠČIĆ, D. Human-Robot Interaction in Public Spaces. In: **2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)**. [S.l.: s.n.], 2022. p. 1287–1289.
- 24 MOHAN, M.; KUCHENBECKER, K. J. A Design Tool for Therapeutic Social-Physical Human-Robot Interactions. In: **2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)**. [S.l.: s.n.], 2019. p. 727–729. ISSN: 2167-2148.
- 25 PLEVA, M.; JUHAR, J.; ONDAS, S.; HUDSON, C. R.; BETHEL, C. L.; CARRUTH, D. W. Novice User Experiences with a Voice-Enabled Human-Robot Interaction Tool. In: **2019 29th International Conference Radioelektronika (RADIOELEKTRONIKA)**. [S.l.: s.n.], 2019. p. 1–5.

- 26 QBILAT, M.; IGLESIAS, A.; BELPAEME, T. A Proposal of Accessibility Guidelines for Human-Robot Interaction. **Electronics**, v. 10, n. 5, p. 561, jan. 2021. ISSN 2079-9292. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.
- 27 RODRÍGUEZ-GUERRA, D.; SORROSAL, G.; CABANES, I.; CALLEJA, C. Human-Robot Interaction Review: Challenges and Solutions for Modern Industrial Environments. **IEEE Access**, v. 9, p. 108557–108578, 2021. ISSN 2169-3536. Conference Name: IEEE Access.
- 28 SOLEMAN, C.; BOUSQUAT, A. Políticas de saúde e concepções de surdez e de deficiência auditiva no SUS: um monólogo? **Cadernos de Saúde Pública**, v. 37, p. e00206620, ago. 2021. ISSN 0102-311X, 0102-311X, 1678-4464. Publisher: Escola Nacional de Saúde Pública Sergio Arouca, Fundação Oswaldo Cruz.
- 29 NASCIMENTO, J. A. d. A.; SEIXAS, J. A. DEFICIÊNCIA AUDITIVA E SURDEZ: DO ABANDONO À INCLUSÃO. **Boletim de Conjuntura (BOCA)**, v. 8, n. 24, p. 74–86, dez. 2021. ISSN 2675-1488. Number: 24.
- 30 SOUZA, T. O. de; OLIVEIRA, A. S. C. Lessa-de; MATOS, C. G. de. NATUREZA GRAMATICAL, PROCESSAMENTO E ACEITABILIDADE DOS ELEMENTOS LINGUISTICOS: AÇÕES CONSTRUÍDAS E CLASSIFICADORES EM LIBRAS.
- 31 CRISTIANO, A. **Os Cinco Parâmetros da Libras**. [Accessed 29-10-2023].
- 32 CARVALHO, E. P. N. d.; BARBOSA, L. L. C.; CARVALHO, P. L. d. A importância do ensino de Libras desde a educação infantil: uma questão de inclusão social. **Research, Society and Development**, v. 10, n. 12, p. e587101221102–e587101221102, out. 2021. ISSN 2525-3409. Number: 12.
- 33 GOOGLE. **Hand landmarks detection guide | MediaPipe | Google for Developers — developers.google.com**. [Accessed 29-10-2023].
- 34 HANCOCK, J. T.; KHOSHGOFTAAR, T. M. Survey on categorical data for neural networks. **Journal of Big Data**, v. 7, n. 1, p. 28, abr. 2020. ISSN 2196-1115.
- 35 LI, Z.; LIU, F.; YANG, W.; PENG, S.; ZHOU, J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. **IEEE Transactions on Neural Networks and Learning Systems**, v. 33, n. 12, p. 6999–7019, dez. 2022. ISSN 2162-2388. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.
- 36 KOPUKLU, O.; KOSE, N.; GUNDUZ, A.; RIGOLL, G. Resource Efficient 3D Convolutional Neural Networks. In: . [S.l.: s.n.], 2019. p. 0–0.
- 37 HEWAMALAGE, H.; BERGMEIR, C.; BANDARA, K. Recurrent neural networks for time series forecasting: Current status and future directions. **International Journal of Forecasting**, v. 37, n. 1, p. 388–427, 2021. ISSN 0169-2070.
- 38 HOUDT, G. V.; MOSQUERA, C.; NÁPOLES, G. A review on the long short-term memory model. **Artificial Intelligence Review**, v. 53, n. 8, p. 5929–5955, dez. 2020. ISSN 1573-7462.
- 39 SAMAL, B.; PANDA, M. Integrative Review on Vision-Based Dynamic Indian Sign Language Recognition Systems. In: **2021 1st Odisha International Conference on Electrical Power Engineering, Communication and Computing Technology(ODICON)**. [S.l.: s.n.], 2021. p. 1–6.

- 40 ZHANG, W.; WANG, J.; LAN, F. Dynamic hand gesture recognition based on short-term sampling neural networks. **IEEE/CAA Journal of Automatica Sinica**, v. 8, n. 1, p. 110–120, jan. 2021. ISSN 2329-9274. Conference Name: IEEE/CAA Journal of Automatica Sinica.
- 41 JARABESE, M. B. D.; MARZAN, C. S.; BOADO, J. Q.; LOPEZ, R. R. M. F.; OFIANA, L. G. B.; PILARCA, K. J. P. Sign to Speech Convolutional Neural Network-Based Filipino Sign Language Hand Gesture Recognition System. In: **2021 International Symposium on Computer Science and Intelligent Controls (ISCSIC)**. [S.l.: s.n.], 2021. p. 147–153.
- 42 TSAI, C.-T.; DING, J.-J. Combined 2D and 3D Convolution Residual Attention Network for Hand Gesture Recognition. In: **2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)**. [S.l.: s.n.], 2022. p. 104–108. ISSN: 2640-0103.
- 43 ALY, S.; ALY, W. DeepArSLR: A Novel Signer-Independent Deep Learning Framework for Isolated Arabic Sign Language Gestures Recognition. **IEEE Access**, v. 8, p. 83199–83212, 2020. ISSN 2169-3536. Conference Name: IEEE Access.
- 44 DWARAMPUDI, M.; REDDY, N. V. S. Effects of padding on lstms and cnns. **CoRR**, abs/1903.07288, 2019.
- 45 JUNG, Y. Multiple predicting k-fold cross-validation for model selection. **Journal of Nonparametric Statistics**, Taylor Francis, v. 30, n. 1, p. 197–215, 2018.
- 46 KöPüKLü, O.; GUNDUZ, A.; KOSE, N.; RIGOLL, G. Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks. In: **2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)**. [S.l.: s.n.], 2019. p. 1–8.