ERICA AMOEDO SIQUEIRO COUTINHO DE ALMEIDA

EYEASSISTIVE: UM SISTEMA ASSISTIVO PARA SUPORTE A COMUNICAÇÃO POR MEIO DO OLHAR BASEADO EM DEEP LEARNING

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Sistemas de Informação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Bacharel em Sistemas de Informação.

Orientador(a): Prof. Dr. Fabio Santos da Silva

Manaus – Julho – 2022

Resumo

As tecnologias assistivas são fundamentais para prover mais independência a pessoas portadoras de deficiência, mas nem sempre são acessíveis para aqueles que as necessitam. Para pacientes não verbais com paralisia, a obtenção de aparelhos que facilitam a comunicação muitas vezes é difícil ou impossível devido a valores altos e falta de disponibilidade. Diante desta situação, este trabalho propõe um sistema de comunicação baseado em visão computacional e deep learning, que utiliza apenas um computador conectado a uma webcam como forma de diminuir custos. O uso destas técnicas permite a detecção de rosto e classificação de movimentação do olhar, que é convertida em comandos para o controle do programa.

Palavras-Chave: Redes neurais convolucionais; Deep learning; Tecnologia assistiva; Comunicação; Visão computacional.

Abstract

Assistive technologies are essential to provide more independence to people with disabilities, but they are not always accessible to those who need them. For non-verbal patients with paralysis, obtaining devices that facilitate communication is often difficult or impossible due to high costs and lack of availability. Considering this situation, this work proposes a communication system based on computer vision and deep learning, which utilizes only a computer connected to a webcam as a way to decrease costs. The use of these techniques allows face detection and gaze movement classification, which is converted into commands for controlling the program.

Keywords: Convolutional neural networks; Deep learning; Assistive technology; Communication; Computer vision.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

As tecnologias assistivas são definidas como recursos e serviços que contribuem para proporcionar ou ampliar as habilidades funcionais de pessoas que possuem deficiência. Tal tecnologia permite melhorar a inclusão e independência dessas pessoas.

Essas tecnologias podem variar desde periféricos adaptados e sistemas operacionais modernos que possuem controle por voz e leitores de tela, até sistemas mais avançados que permitem uma maior independência para seus usuários. Com isso, trazem melhoras significativas para a vida de pacientes com vários tipos de doenças.

Por exemplo, para pessoas com paralisia muscular, existem periféricos adaptados e sistemas operacionais modernos que possuem controle por voz e leitores de tela como funções básicas. Vale ressaltar que os avanços tecnológicos estão viabilizando o desenvolvimento de sistemas assistivos cada vez melhores, que permitem uma independência maior para seus usuários.

Segundo um relatório produzido em 2021 pela Organização Mundial da Propriedade Intelectual, mundialmente cerca de 1 bilhão de pessoas necessitam do apoio de tecnologias assistivas, porém apenas uma em dez pessoas tem acesso aos produtos que precisam.(OMPI, 2021)

Para pessoas com mais de um tipo de deficiência, a acessibilidade pode ser ainda mais desafiadora. Pacientes que possuem condições que dificultam ou impossibilitam a comunicação verbal, conhecidos como pacientes não verbais, enfrentam desafios adicionais para usar tecnologias assistivas. Para esses pacientes, as tecnologias assistivas que dependem do controle por voz podem não ser úteis, mas existem outras opções disponíveis, como tecnologias que usam

o controle por olhar, que podem ajudar pacientes com tetraplegia ou paraplegia a controlar dispositivos eletrônicos com o movimento dos olhos.

Em suma, as tecnologias assistivas são um componente importante da inclusão e independência de pessoas com deficiências, permitindo que elas participem plenamente da sociedade e realizem atividades diárias com mais facilidade. À medida que a tecnologia continua a evoluir, espera-se que mais soluções inovadoras e acessíveis sejam desenvolvidas para atender às necessidades de pessoas com deficiências em todo o mundo.

1.1 Descrição do Problema

Torna-se importante ressaltar que para pessoas não-verbais com tetraplegia, existem dispositivos com câmeras ou óculos que rastreiam os movimentos dos olhos do usuário e traduzem isso para movimentos na tela. Além disso, também há pesquisas para o desenvolvimento de produtos que traduzem ondas cerebrais para texto.(KARIDIS, 2016)(HERFF et al., 2015)

Entretanto, esses dispositivos não são disponibilizados de forma gratuita pelo sistema público de saúde, normalmente, possuem um alto valor, o que os torna inacessíveis para a população mais vulnerável.

A falta de sistemas de comunicação acessíveis para pessoas não verbais com paralisia pode levar à exclusão social e à dificuldade em realizar atividades cotidianas, como fazer compras, trabalhar e interagir com amigos e familiares. Portanto, é importante desenvolver tecnologias de comunicação acessíveis e fáceis de usar para ajudar as pessoas com paralisia a se comunicarem com o mundo ao seu redor.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um sistema assistivo baseado em tecnologias abertas e de baixo custo para suporte a comunicação humana mediada pelo olhar utilizando deep learning.

1.2.2 Objetivos Específicos

1. Analisar trabalhos similares para comparar seus objetivos e implementação com o sistema proposto neste trabalho.

2. Criar um dataset com images classificadas de acordo com a direção do olhar para o treinamento do sistema.

3. Criar um modelo de deep learning que obtém resultados satisfatórios.

4. Desenvolver um sistema para capturar imagens da webcam e classifica-las usando visão computacional.

5. Criar um teclado virtual e um painel contendo imagens que representam situações comuns controlado pelo sistema de classificação de imagens.

6. Testar a solução desenvolvida neste trabalho e realizar melhorias de acordo com os resultados obtidos.

1.3 Justificativa

Diante do problema descrito, este trabalho propõe a criação de um sistema de comunicação controlado pelo olhar como uma alternativa aos dispositivos de digitação especializados. O sistema requer apenas um computador com webcam, o que o torna mais acessível a pessoas que não possuem meios de comprar tais dispositivos, não conseguem criar seus próprios dispositivos adaptados ou possuem formas de paralisia que dificultam o uso dos periféricos existentes.

A criação desse sistema também pode trazer benefícios para outras áreas, como saúde e tecnologia. Ele poderia ser usado em clínicas e hospitais, ajudando pacientes que perderam temporariamente a capacidade de falar após uma cirurgia ou trauma. Além disso, a tecnologia pode ser aplicada em outras áreas, como jogos e entretenimento, ampliando o acesso a esses recursos para pessoas com paralisia.

1.4 Metodologia

A metodologia empregada neste trabalho consiste nas seguintes etapas, sendo usado o modelo de desenvolvimento SCRUM adaptado:

1. Realização de uma revisão bibliográfica para estudar os conceitos necessários para o trabalho proposto, principalmente a área de deep learning, tecnologias assistivas e visão computacional.
2. Levantamento e análise de trabalhos relacionados.
3. Modelagem do sistema e criação do dataset.
4. Implementação e análise do modelo de deep learning desenvolvido.
5. Implementação do painel e teclado virtual controlados pelo modelo desenvolvido.
6. Realização de testes das funções do sistema.

1.5 Estrutura do Trabalho

No capítulo 1 foi feita a apresentação da problemática do trabalho, seu tema, os objetivos e a metodologia usada. No capítulo 2 é apresentado o referencial teórico abordando conceitos de tecnologias assistivas, deep learning, redes neurais convolucionais e visão computacional. Também é feita a apresentação de trabalhos e produtos semelhantes, incluindo uma comparação com a aplicação proposta.

O capítulo 3 introduz a solução proposta para o problema, seu desenvolvimento e quais tecnologias foram usadas na sua criação. No capítulo 4 são mostrados os resultados parciais obtidos e no capítulo 5 é feita a conclusão.

Capítulo 2

Fundamentação Teórica

Nas seções seguintes serão apresentados conceitos relevantes ao desenvolvimento do trabalho nas áreas de tecnologias assistivas, deep learning, redes neurais convolucionais e visão computacional.

2.1 Tecnologias Assistivas

O conceito de tecnologias assistivas se refere ao uso a qualquer tipo de recurso ou serviço que contribui para proporcionar ou ampliar as habilidades funcionais de uma pessoa com deficiência, consequentemente ampliando sua independência ou inclusão. No Brasil, o Comitê de Ajudas Técnicas definiu o conceito como:

> ”Tecnologia Assistiva é uma área do conhecimento, de característica interdisciplinar, que engloba produtos, recursos, metodologias, estratégias, práticas e serviços que objetivam promovera funcionalidade, relacionada à atividade e participação, de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social”

(ROCHA, 2005) (BERSCH, 2017)

De acordo com a Pesquisa Nacional de Saúde, no Brasil mais de 17,3 milhões de pessoas com dois anos ou mais possuem algum tipo de deficiência, tornando claro a necessidade dessas

tecnologias. (IBGE, 2021)

Cada tipo de deficiência necessita de tipos de acessibilidade diferentes, fazendo com que a área possa ser dividida em vários tipos, incluindo: auxílios para a vida diária, recursos de acessibilidade ao computador, auxílios de mobilidade, adaptações em veículos e sistemas de comunicações alternativos.

Atualmente, as ferramentas de comunicação alternativa mais usadas são pranchas e cartões de comunicação. Neles são feitos cartões físicos que correspondem a conceitos e objetos, permitindo que usuário se expresse por meio da interação com os cartões. As pranchas utilizam o mesmo conceito, mas também podem incluir letras e números no lugar de imagens.(SARTORETTO, 2022)

Para as pranchas, existem diferentes sistemas simbólicos que possuem uma identificação própria e podem ser agrupados de acordo com diversos aspectos. Um desses sistemas é o PCS (Picture Communication Symbols), constituído por mais de 3000 desenhos que representam substantivos, pronomes, verbos e adjetivos, sendo indicado para crianças. (SARTORETTO, 2022)

Figura 2.1: Exemplo de prancha que utiliza o sistema PCS.

Fonte: (SARTORETTO, 2022)

2.2 Deep Learning

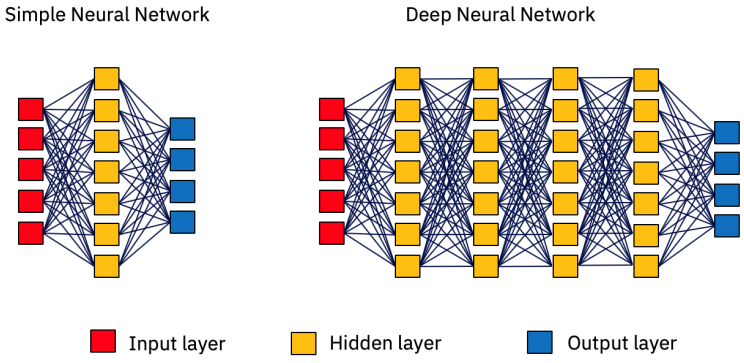
Deep Learning é uma subárea da inteligência artificial que foca na criação de grandes modelos de redes neurais capazes de tomar decisões precisas baseadas em dados. Essa área está trazendo grandes avanços na resolução de problemas difíceis para a comunidade de inteligência artificial, devido a sua capacidade de encontrar estruturas complexas em dados com grandes dimensões.

Para entender sobre Deep Learning, é necessário primeiro compreender a área de Machine Learning. Pode-se dizer que o objetivo de Machine Learning é aprender funções tendo como base dados, conhecidos como datasets. Uma forma de representar essas funções é por meio de uma rede neural. De fato, os padrões que os algoritmos de deep learning obtém dos datasets são funções representadas como redes neurais. (LECUN; BENGIO; HINTON, 2015)

Uma rede neural aprende por meio de uma estratégia de dividir e conquistar: cada neurônio na rede aprende uma função simples, e a função mais complexa, definida pela rede, é criada pela combinação dessas funções mais simples. (KELLEHER, 2019)

O nome ”Deep Learning”surgiu desse conceito. A palavra ”deep”se refere a profundidade do modelo, dada pelo número de camadas, que são as funções aprendidas pelos neurônios da rede. Por exemplo, em um modelo de reconhecimento facial, cada camada irá aprender uma característica como nariz, olhos, boca, etc. e a camada final, conhecida como output layer ou camada de saída, irá identificar que a imagem contém um rosto. A figura 2.2 demonstra a diferença entre redes neurais simples e redes neurais que usam deep learning.

Figura 2.2: Rede neural simples e rede neural com deep learning



Fonte: (CERON, 2020)

Apesar do modelo ser capaz de determinar quais características deve aprender em cada camada por si mesmo, ainda há a necessidade de ajustes manuais para melhorar a performance do modelo, principalmente pelo ajuste do número de camadas. Um modelo muito complexo para dados muito simples pode acabar aprendendo características que não são relevantes, gerando resultados ruins.

O processo mais comum para o treinamento de uma rede neural é iniciar o treinamento com uma rede onde os parâmetros são inicializados randomicamente. Essa rede inicial irá produzir resultados equivocados na sua capacidade de relacionar os dados de entrada e os valores de saída esperados do dataset.

Com isso, o processo de treinamento irá iterar pelos exemplos no dataset, usando a diferença entre a saída retornada pela rede e a saída esperada pelo dataset de treinamento para ajustar os parâmetros de acordo com os dados. Quando o algoritmo encontra uma função suficientemente precisa, o processo de treinamento encerra e o modelo final é salvo.

Após esse processo, se inicia a segunda fase, chamada de inferência. Nela o modelo será aplicado a novos exemplos, ou seja, dados que não fazem parte do dataset de treinamento. A partir desse ponto o modelo pode ser integrado a outras aplicações. (KELLEHER, 2019)

2.2.1 Função de Ativação

A função de ativação, também conhecida como transfer function, decide se um neurônio na rede irá ser ativado ou não, ou seja, determina se a informação contida por esse neurônio é importante para a rede. Essa decisão é derivada da saída de uma operação feita sobre um conjunto de valores de entrada passados para um neurônio ou camada.

O motivo do uso da função de ativação é adicionar a não-linearidade a uma rede neural por meio de passos adicionais na aprendizagem. Por exemplo, caso uma rede neural não use funções de ativação, ela apenas irá realizar transformações lineares e todas as camadas irão se comportar da mesma forma, fazendo com que o aprendizado de tarefas complexas se torne impossível.

Existem vários funções diferentes que são utilizadas para a ativação, porem as mais comuns

são:

- ReLU: É a função mais comum para as camadas ocultas (as camadas presentes entre a entrada e a saída) e também é uma das mais simples. Consiste em uma função linear que retorna zero se a entrada é negativa e retorna a mesma entrada caso seja positiva.

- Sigmoid: Leva qualquer valor real como entrada e retorna valores entre 0 e 1. Quanto maior a entrada, mais próxima a saída será de 1 e quanto menor, mais próxima será de 0. É usada principalmente para problemas de classificação binária.

- Softmax: Usada principalmente na camada de saída com problemas que envolvem a classificação multiclasse. Podendo ser descrita como uma combinação de múltiplas funções sigmoid, ela calcula probabilidades relativas.

(BAHETI, 2022) (WOOD, 2020)

2.2.2 Métricas de Desempenho

As métricas são resultados obtidos diante da testagem de um modelo usando dados novos. Um modelo desenvolvido deve ser constantemente monitorado e testado com dados reais para verificar se é adequado para a realidade para qual foi desenvolvido. Por meio desses resultados é possível identificar falhas que ocorreram na criação do dataset ou durante o treinamento.

As métricas usadas dependem de qual tipo de modelo está sendo desenvolvido e de seu objetivo. Para tarefas de classificação, as métricas mais comuns são:

- Accuracy: Se refere a acurácia da classificação, dada pela razão entre o número de previsões corretas e o número total de amostras de entrada.

- Precision: É o número de resultados positivos corretos dividido pelo número de resultados positivos previstos pelo classificador.

- Recall: É o número de resultados positivos dividido pelo número de todas as amostras relevantes (todas que deveriam ser identificadas como positivas).

- F1 Score: É a média harmônica entre a precisão e recall. Mostra quão preciso o classificador (quantas instâncias são classificadas corretamente) e quão robusto (não erra uma quantidade significante de instâncias).

(BAJAJ, 2021) (MISHRA, 2018)

2.2.2.1 Matriz de Confusão

A Matriz de Confusão é uma visualização tabular das previsões dos modelos contra seus rótulos verdadeiros. Cada linha da matriz de confusão representa as instâncias em uma classe prevista e cada coluna representa as instâncias das classes corretas. Não se trata exatamente de uma métrica de performance, mas sim de uma base por qual as os outras métricas avaliam resultados.

Cada célula na matriz representa um fator de avaliação:

- Verdadeiros Positivos indicam quantas amostras de classe positivas foram previstas corretamente pelo modelo.
- Verdadeiros Negativos indicam quantas amostras de classe negativas foram previstas corretamente pelo modelo.
- Falsos Positivos indicam quantas amostras de classe negativas foram previstas incorretamente pelo modelo.
- Falsos Negativos indicam quantas amostras de classe positivas foram previstas incorretamente pelo modelo.

(ALTEXSOFT, 2022)

2.2.3 Hold-out

No treinamento de modelos de machine learning, o método hold-out é uma técnica de validação da performance de modelos. Ele envolve a divisão do dataset em conjuntos separados para treinamento e teste. Assim, é possível verificar o funcionamento do modelo com dados novos.

A porcentagem de dados que serão separados depende da quantidade de amostras no dataset. Uma quantidade muito pequena no conjunto de treinamento pode gerar resultados ruins, mas uma quantidade muito grande pode gerar modelos que não se generalizam bem. Geralmente são usados 70% para o treinamento e 30% para a testagem do modelo. (FIRMIN, 2019) (C3.AI, 2022)

2.3 Redes Neurais Convolucionais

As redes neurais convolucionais são uma classe de redes neurais artificiais que utilizam camadas convolucionais para filtrar as informações uteis de uma entrada. A convolução é uma operação matemática que descreve a regra de como mesclar duas funções ou partes de informação. (DETTMERS, 2015)

Na terminologia das redes neurais convolucionais, o primeiro argumento é chamado de input, o segundo é o kernel e o output é chamado de feature map. Em aplicações de machine learning o input tende a ser um array multidimensional de dados e o kernel um array multidimensional de parâmetros que são adaptados pelo algoritmo de aprendizado. (GOODFELLOW; BENGIO; COURVILLE, 2016)

2.3.1 Convolução

Em uma camada de convolução, é feita a operação de convolução entre os feature maps da camada anterior e os kernels de aprendisado. O resultado será passado pela função de ativação para formar o feature map que será a saída dessa camada. Cada mapa de saída pode combinar convoluções com múltiplos mapas de input. (BOUVRIE, 2006)

O uso de convolução permite que o modelo utilize interações esparsas, diferente dos modelos tradicionais que utilizam multiplicação de matrizes. Nesses modelos é feita uma multiplicação de uma matriz de parâmetros com um parâmetro separado que descreve a interação de cada unidade de entrada e cada unidade de saída. Dessa forma, todas as unidades de saída interagem com todas as unidades de entrada.

Já nas interações esparsas são feitas operações de convolução com o kernel menor que o input, permitindo que durante o processamento de uma imagem seja feita a detecção de características menores em imagens grandes. O uso dessas operações significa que a quantidade de parâmetros que deve ser guardada é menor, trazendo um menor uso de memória e maior eficiência computacional.

Um benefício adicional é a possibilidade de camadas mais profundas interagirem com porções grandes do input, permitindo que interações complexas sejam descritas por interações entre blocos simples que descrevem apenas interações esparsas. Além disso, parâmetros podem ser utilizados para mais de uma função no modelo, conhecido como compartilhamento de parâmetros. (GOODFELLOW; BENGIO; COURVILLE, 2016)

2.3.2 Camadas Totalmente Conectadas

As camadas totalmente conectadas em redes neurais convolucionais possuem um funcionamento similar a outro tipo de rede neural, as perceptron multicamadas. Esse tipo de camada leva como entrada um vetor de nós que haviam sido ativados em camadas de convolução anteriores. O vetor de entrada passa por duas ou três camadas densas e uma função de ativação final antes de ser enviada à camada de saída.(MCDERMOTT, 2022)

O objetivo do uso deste tipo de camada é a geração de predições, portanto geralmente está localizada no final de uma arquitetura de rede, para que possa usar os dados de características aprendidas para esse propósito. O tipo de função de ativação deve ser escolhida de acordo com as características do problema que se pretende resolver. (MCDERMOTT, 2022)

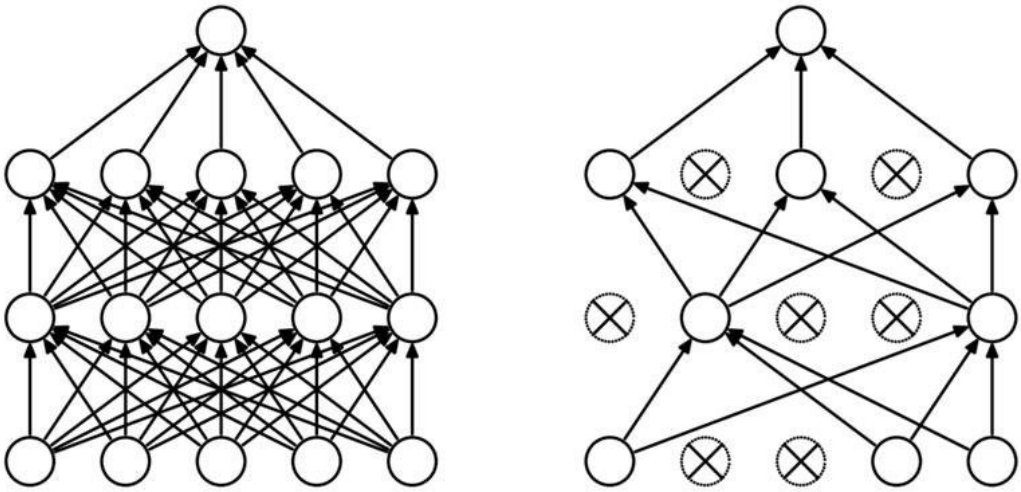
2.3.3 Droupout

Dropout é uma técnica para regularização com objetivo de reduzir o overfitting. O overfitting ocorre quando as funções aprendidas pela rede neural acabam se tornando próximas demais aos pontos de dados usados no treinamento, trazendo resultados ruins quando o modelo é usado em dados novos. (ROSEBROCK, 2021)

O uso de camadas de dropout reduz esse efeito por meio da nulificação da contribuição de

uma determinada quantidade de neurônios na próxima camada e deixa o restante sem modificações. Isso previne que certos pesos acabem se especializando demais em certas características, melhorando a capacidade de generalização da rede. (BAHETI, 2022) (ROSEBROCK, 2021)

Figura 2.3: Exemplo do uso de dropout em uma rede



(a) Standard Neural Net (b) After applying dropout.

Fonte: (SRIVASTAVA et al., 2014)

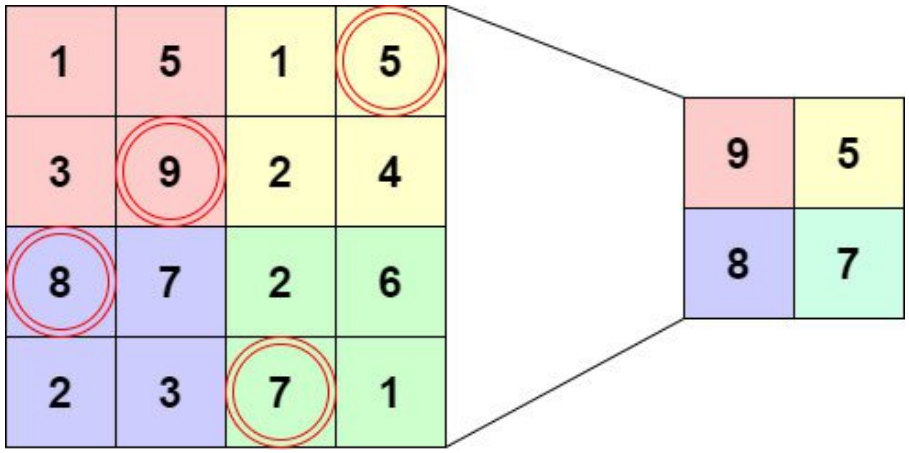
2.3.4 Pooling

Outro procedimento comumente usado pelas redes neurais convolucionais é o pooling. Esse procedimento usa uma área do input e o reduz para apenas um valor. Existem vários tipos de pooling diferentes, mas todos eles tem como objetivo prover invariância básica a rotações e translações, ou seja, faz com que pequenas mudanças no local de uma característica não causem alterações em sua detecção. (DETTMERS, 2015)

O pooling também é essencial para lidar com imagens de tamanhos variáveis. Como a camada de classificação recebe inputs de determinado tamanho, o pooling é utilizado para que essa camada receba inputs com o mesmo valor independente do tamanho da imagem.

Quanto maior for uma área de pooling, mais informação é condensada, gerando uma rede neural que ocupa menos espaço na memória. Porém o uso de uma área de pooling grande demais pode levar a uma perda de capacidade de preditiva devido a diminuição da quantidade de informação na camada.(KELLEHER, 2019) (BROWNLEE, 2019)

Figura 2.4: Exemplo de pooling por valor máximo



Fonte: (MCDERMOTT, 2022)

2.4 Visão Computacional

Visão Computacional é uma área da ciência da computação que tem como foco criar sistemas digitais que processam, analisam e interpretam dados visuais de forma similar a humanos. O uso de redes neurais convolucionais permite a criação de modelos mais complexos e precisos, aumentando também a quantidade de aplicações possíveis para essa área. Algumas das principais aplicações são: (BABICH, 2020) (VOULODIMOS et al., 2018)

1. Classificação de imagens: atribuição de uma classe pré-definida uma dada entrada.
2. Detecção de objetos: identificação e localização de determinados objetos.
3. Rastreamento de objetos: detecção e rastreamento da trajetória de objetos em vídeos ou sequências de imagens.
4. Segmentação semântica: detecção de objetos baseada nos pixeis específicos relacionados ao objeto.
5. Restauração de imagem: geração de uma imagem limpa tendo como partida uma imagem de baixa qualidade.

De acordo com o livro Introdução à Visão Computacional, mesmo com vários os tipos de aplicações existentes, sistemas que utilizam a visão computacional geralmente possuem um fluxo similar. (BARELLI, 2018)

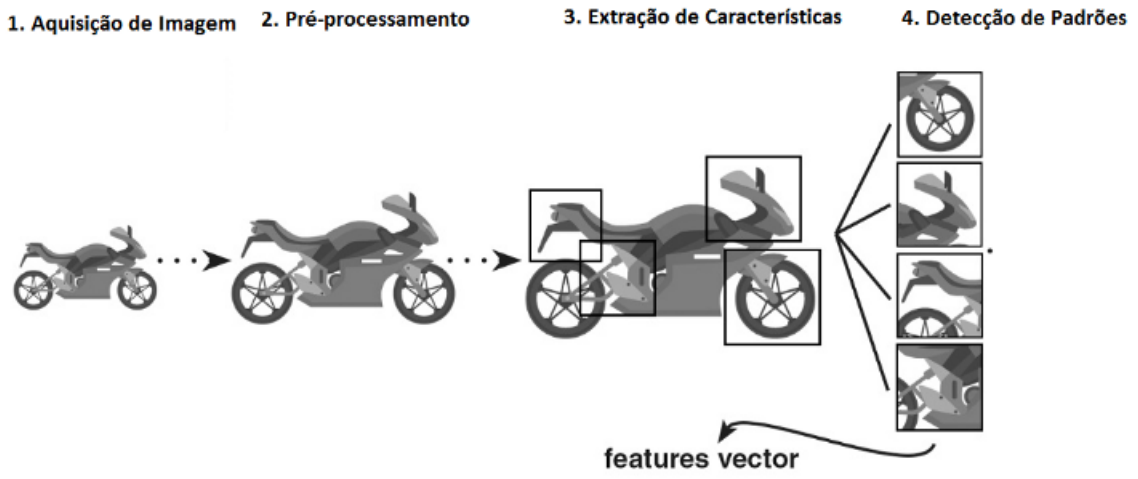
Figura 2.5: Fluxo de um sistema que utiliza visão computacional

Fonte: (BARELLI, 2018)

A primeira etapa realizada é a aquisição da imagem. O sistema irá usar os sensores que possui para obter e digitalizar a imagem que será trabalhada, podendo ser apenas uma imagem ou um vídeo. Após a imagem digitalizada ser obtida, o sistema passa para a segunda etapa, que consiste no pré-processamento.

No pré-processamento são aplicadas técnicas que realçam bordas e figuras geométricas e que diminuem o ruído presente na imagem. O uso destas técnicas auxilia na obtenção de informações, aumentando a precisão do modelo .

Figura 2.6: Esquema de pré-processamento para visão computacional



Fonte: (PUBLICATIONS, 2019)

Na etapa seguinte, de segmentação, os objetos de interesse são separados da imagem original para facilitar a extração de características dos objetos. Após as regiões de interesse terem sido segmentadas, é feita a extração de características do objeto, como tamanho e cor. Por fim, é feito o processamento de alto nível, onde o objeto segmentado é reconhecido e classificado de acordo com suas características. (BARELLI, 2018)

2.5 Trabalhos Relacionados

Nesta seção serão apresentadas outras ferramentas de acessibilidade que também utilizam da visão para facilitar que o usuário se comunique.

2.5.1 EyeOn Eye Tracking

EyeOn é um produto de rastreamento de visão para controle de dispositivos de comunicação, produzido pela empresa EyeTech Digital Systems. Ele utiliza deep learning para fazer detecção e rastreamento dos olhos do usuário com alta precisão.

As funções específicas variam de acordo com o modelo específico, mas em geral consiste em um tablet com uma câmera que roda um sistema proprietário para realizar o rastreamento do olhar. Ele também inclui um sistema infravermelho para que possa funcionar em lugares escuros e alto falantes para uso de tecnologias de conversão de texto em fala.

Também inclui o software de acessibilidade OnBright, que inclui funções de controle de dispositivos como smart TVs, um navegador web, controle de mouse e teclado, um sistema de comunicação baseado em texto e um sistema de comunicação baseado em símbolos, todos controlados por meio da visão do usuário. (SYSTEMS, 2022)

2.5.2 Image Based Eye Controlled Assistive System

Proposto no artigo "An Image Based Eye Controlled Assistive System for Paralytic Patients", consiste em uma aplicação para computadores que utiliza o movimento dos olhos para a comunicação. Ele utiliza técnicas de processamento de imagem para realizar a detecção do olhar.

Seu fluxo de funcionamento consiste em três módulos principais: detecção de rosto e olhos, detecção de pupilas e detecção de olhar e assistência. No primeiro módulo, o sistema utiliza classificadores Haar-cascade para a detecção facial. O segundo módulo utiliza o algoritmo Circular Hough Transform, feito para a detecção de círculos em imagens, para a detecção da pupila. No último módulo, são feitos cálculos baseados na pupila detectada e sua distância à margem esquerda do olho para determinar se o usuário está olhando para a esquerda ou direita.

A parte de comunicação do sistema funciona por meio da exibição de uma grade de imagens. O usuário olha para o lado da grade que contém a imagem que deseja selecionar e a grade muda para incluir apenas os itens que estavam nesta direção. Esse processo se repete até que sobre apenas a opção desejada. (ALVA et al., 2017)

2.5.3 I-Series

I-Series é um dispositivo de geração de voz controlado por meio do olhar, produzido pela empresa Tobii Dynavox. Possuindo modelos de diferentes tamanhos, são computadores customizados em formato de tablet que rodam o sistema Windows e inclui softwares de acessibilidade pré-instalados. Sua detecção de olhar funciona por meio de luz quase infravermelha, que é refletida pelos olhos e detectada pelas câmeras do dispositivo.

Ele inclui os seguintes softwares:

- TD Snap: software de comunicação baseado em símbolos. Permite que o usuário escolha entre vários tipos de painéis de símbolos que também podem ser editados.
- Communicator 5: software de comunicação baseado em texto. Possui vários tipos de layouts de teclados, frases pré-definidas e previsões de texto.
- TD Control: software para interação com o Windows por meio do olhar. Pode ser customizado para exibir apenas os botões mais utilizados pelo usuário.

(TOBIIDYNAVOX, 2022)

2.5.4 SINA

SINA é um software que substitui o controle do mouse por um controle baseado no olhar capturado pela webcam, incluindo a função de cliques e não possui a necessidade de ser calibrado. Ele tem sua implementação detalhada no artigo ”Hands-free vision-based interface for computer acessibility”.

A arquitetura do sistema de visão computacional é projetada para pessoas que possuem deficiências que afetam o movimento dos membros superiores. Seu único requerimento é que no início do uso do software o usuário posicione a cabeça diretamente para a webcam, sem colocá-la em um ângulo, mas após o reconhecimento inicial ele fica livre para se movimentar.

O sistema funciona por meio da detecção do rosto com um framework feito usando classificadores de cascata e as características do rosto são obtidas usando cálculos de distância que tem como base a ponta do nariz. Com isso também é possível fazer a detecção de gestos que podem ser mapeados para ações. (VARONA; MANRESA-YEE; PERALES, 2008)

2.5.5 GazePointer

GazePointer é um software gratuito de controle do cursor do mouse por meio do olhar para o sistema Windows. Ele controla apenas o movimento, sem possuir uma forma de realizar cliques. Sua implementação é detalhada no artigo ”GazePointer: A Real Time Mouse Pointer Control Implementation Based On Eye Gaze Tracking”.

Seu funcionamento é baseado na captura de imagens pela webcam do computador, onde o usuário precisa estar na mesma altitude da câmera, numa distância entre 20-75 centímetros, realizar uma calibração no início da utilização do programa e deve manter a cabeça em uma posição estável. Também possui a limitação de não funcionar quando o usuário usa óculos e precisa de boas condições de luz no ambiente.

O processo do funcionamento do software inicia com a obtenção da captura da imagem, que é convertida para escala cinza e é normalizada por meio do ajuste de intensidades dos pixeis usando a equalização de histograma. Após esse processo é feita a detecção de rostos e olhos com um algoritmo de Haar-cascade. Com os olhos detectados, é feita a detecção de das pupilas

e cantos dos olhos usando Hough Circle Transform. Esses dados são utilizados para encontrar o ponto de referência do centro do olho, necessário para realizar os cálculos de movimento. (GHANI et al., 2013)

2.5.6 Comparação Entre As Aplicações

Das ferramentas citadas acima, 3 são projetos de artigo e 2 são dispositivos comerciais de acessibilidade. Apenas 1 dos projetos de artigo é focado em desenvolver um sistema de comunicação, os outros 2 tem o foco em substituir o controle do mouse em qualquer tipo de atividade realizada no computador. Nenhum dos projetos de artigo utiliza deep learning, em vez disso tem o seu funcionamento baseado no uso de técnicas de processamento de imagem. Dos dispositivos comerciais, apenas 1 possui informações sobre o uso de deep learning.

Outro ponto sobre esses produtos é a sua disponibilidade: só é possível adquiri-los por meio do contato direto com as empresas e também é necessário uma indicação médica do dispositivo adequado às necessidades do paciente. Com isso, o sistema proposto neste trabalho, chamado de EYEASSiSTIVE, se diferencia pelo uso de deep learning combinado com facilidade de acesso provida por apenas ser necessário um computador com webcam.

Tabela 2.1: Tabela de comparação entre aplicações

Ferramenta	Deep Learning	Plataforma	Preço
EyeOn	Sim	Dispositivo Próprio	Aproximadamente R$116,881 (INCLUSIVE, 2023)
Image Based Eye Controlled Assistive System	Não	Windows	Grátis
I-Series	Desconhecido	Windows	Aproximadamente R$46,590 (ASSISTIVE, 2023)
SINA	Não	Windows	Grátis

GazePointer	Não	Windows	Grátis
EYEASSiSTIVE (Sistema Proposto)	Sim	Linux/Multiplataforma	Grátis

Capítulo 3

Solução Proposta

Neste capítulo será descrita a solução proposta para um sistema de comunicação baseado em visão computacional e deep learning, incluindo detalhes sobre a criação de seu dataset e do modelo da rede neural convolucional.

3.1 Visão Geral do Problema

O sistema proposto tem como objetivo ser um método de comunicação para pessoas que não conseguem realizar movimentos com seus membros superiores. Além disso, deve ser uma solução de baixo custo, precisando apenas de um computador com webcam.

O funcionamento da aplicação se dá por meio da obtenção de imagens por meio de uma webcam, que serão processadas e classificadas de acordo com um modelo de rede neural convolucional. Os movimentos no software são feitos de acordo com a classe da imagem detectada por esse modelo, sem necessidade de uma fase de calibração para cada usuário.

A aplicação possui dois tipos de interface: um painel com imagens do sistema de símbolos similares ao PCS e um teclado QWERTY. No painel de símbolos são exibidos várias imagens que correspondem a possíveis necessidades do usuário, permitindo uma comunicação mais rápida. As imagens são exibidas em cantos da tela, onde o usuário deve movimentar o olhar para a direção correspondente a sua seleção.

Já na interface do teclado, é exibido um teclado na tela que possui um marcador indicando

qual é a tecla que está sendo selecionada. O usuário movimenta o marcador de acordo com seu olhar e seleciona uma tecla mantendo o olhar no centro da tela. Essa interface tem como objetivo permitir que seja feita uma comunicação mais específica que os símbolos, como por exemplo a digitação do nome de uma pessoa.

Para evitar a realização de movimentos indesejados, o sistema detecta e classifica o olhar múltiplas vezes durante alguns segundos e realiza os movimentos de acordo com a classificação mais comum durante esse período de tempo. Visando melhorar a usabilidade, esse período de tempo varia de acordo com a tela, sendo 2 segundo na tela de símbolos e 3.3 segundos no teclado.

3.2 Processo de Desenvolvimento do Software

Para o desenvolvimento deste trabalho, foi usado o método SCRUM com modificações feitas a partir das necessidades do autor e do orientador. O método SCRUM é constituído por um conjunto de boas práticas empregadas no gerenciamento de projetos e tem o foco na redução do tempo de entrega e a realização de mudanças com facilidade.

Tem como base a realização de pequenos ciclos de atividades durante o projeto, conhecidos como sprints. Cada ciclo tem uma duração predeterminada e foca na execução de uma determinada atividade. Como cada sprint é uma pequena parte do projeto, é possível que alterações sejam sugeridas e facilmente executadas, trazendo mais flexibilidade.(ASANA, 2022)

O SCRUM define três papéis para os participantes de um projeto:

- A equipe (SCRUM Team) que irá fazer o desenvolvimento. Composta pelos indivíduos que irão trabalhar juntos para entregar o que foi definido em cada sprint.
- O SCRUM Master, que faz parte da equipe e a mantém focada e de acordo com as regras do SCRUM. Apesar de seu papel parecer similar, não é um gerente da equipe.
- O Product Manager representa o cliente para qual o projeto está sendo desenvolvido. Ele faz a priorização de tarefas e coordena a equipe, dando opiniões e sugerindo mudanças de acordo com os resultados. Tem um papel semelhante ao gerente de uma equipe tradicional.

Antes do início de um sprint, são definidos os objetivos dele em uma reunião chamada de Sprint Planning Meeting. Os objetivos definidos são colocados em uma lista conhecida como Product Backlog. Com isso se inicia um sprint, onde serão realizadas reuniões diárias (Daily Sprint) para o acompanhamento do progresso. No final do sprint os resultados são apresentados na chamada Sprint Review. (ALTVATER, 2017)

Nas adaptações realizadas para este trabalho, o professor orientador assumiu o papel de SCRUM Master, e a SCRUM Team é composta apenas pela autora. As reuniões do sprint foram realizadas de forma semanal, por meio da ferramenta Google Meet. Os prazos e entregas foram feitos conforme o estabelecido na matéria de TCC.

3.3 Desenvolvimento do Modelo de Classificação do Olhar Humano

Nessa seção será detalhado como foi feita a criação do dataset e o desenvolvimento do modelo utilizado.

3.3.1 Conjunto de Dados

O conjunto de dados usado neste trabalho consiste em um dataset feito a mão, composto por imagens retiradas de vídeos de pessoas movimentando seus olhos. Esses vídeos foram obtidos uma parte por meio de contribuições de voluntários, e outra parte gravados pela autora do trabalho.

As imagens foram extraídas de frames dos vídeos usando a ferramenta ffmpeg, e foram separadas manualmente nas seguintes categorias: centro, esquerda, direita, cima, baixo e fechado. Cada imagem foi colocada em uma pasta diferente de acordo com sua categoria.

Figura 3.1: Exemplo de frame extraído

Fonte: De autoria própria. Imagem retrata a própria autora do trabalho.

Após a separação das imagens, foi feito o pré-processamento para que sejam usadas na visão computacional. Por meio de funções da biblioteca Dlib, é realizada a detecção de rosto e a identificação de 68 pontos que representam 68 marcos faciais. Usando esses marcos é possível extrair a região dos olhos, sendo escolhido o uso do olho direito.

Figura 3.2: Pontos identificados pelo Dlib

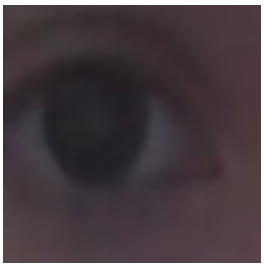
Fonte: De autoria própria. Imagem retrata a própria autora do trabalho.

Com a região do olho direito identificada, é feito um corte na imagem usando a biblio-

teca OpenCV para retirar as partes da imagem que não são relevantes. A imagem cortada é redimensionada para 24 x 24, pronta para o uso no treinamento do modelo.

Assim que o pré-processamento foi concluído, as imagens foram movidas para uma pasta que será lida no momento do treinamento, mantendo a estrutura de classificação feita inicialmente. A figura 3.3 mostra um exemplo de imagem obtida após a conclusão do processo.

Figura 3.3: Imagem após o pré-processamento



Fonte: De autoria própria. Imagem retrata a própria autora do trabalho.

Para complementar as imagens geradas por esse procedimento, foram usadas algumas partes dos seguintes datasets:

- Eye dataset: Contem imagens de pessoas com olhos abertos e fechados, criado apartir de imagens coletadas diretamente da internet e do banco de dados Labeled Face in the Wild. (PATIL, 2021)

- Eye-dataset: Criado para a automatização da movimentação de cadeiras de rodas, é um dataset composto por imagens de olhos classificadas em fechado, olhar para a esquerda, olhar para a direita e olhar para a frente. (SHAH, 2020)

- SynthesEyes: Tem o seu desenvolvimento detalhado no artigo "Rendering of Eyes for Eye-Shape Registration and Gaze Estimation". Foi criado com o objetivo de reduzir o tempo de coleta e classificação de imagens para projetos envolvendo o olhar em visão computacional usando modelos 3D realistas. (WOOD et al., 2015)

- Eyes - Open or Closed: Criado para a detecção de sonolência, é composto por imagens de olhos classificados em abertos e fechados.(MADAN, 2021)

- GI4E - Gaze Interaction for Everybody: Possui imagens de pessoas movimentando os olhos, com 12 imagens para 103 usuários. Foi realizado o mesmo procedimento de detecção de olhar descrito anteriormente para a classificação das imagens deste dataset.(VILLANUEVA et al., 2013)

- GazeCapture: É um dataset de grande escala criado para o treinamento da rede neural convolucional iTracker, capaz de fazer o rastreamento ocular em tempo real em dispositivos móveis comuns, como telefones e tablets, sem a necessidade de sensores adicionais. Devido ao seu grande tamanho, foi usada apenas uma pequena parcela deste dataset.(KRAFKA et al., 2016)

- NeuroEye Pupil Center Dataset: Dataset contendo imagens de movimentação do olhar, criado para o estudo de movimentos oculares. (HASSAN, 2022)

No total, foram obtidas 6242 imagens para o dataset, com sua distribuição detalhada na tabela 3.1.

Tabela 3.1: Distribuição das amostras

Up	788
Down	951
Left	1047
Right	1027
Center	1362
Closed	1067

Foi optado por não realizar o procedimento comum de transformar as imagens para escala cinza, devido aos testes da aplicação feitos com o modelo usando essa técnica trazerem resultados ruins. Para a validação dos resultados foi usado o método de hold-out, com 70% dos dados no grupo de treinamento e 30% no grupo de teste.

3.3.2 Arquitetura da Rede Neural Convolucional Experimentada e Seus Parâmetros e Hiperparâmetros

O tipo de rede escolhida foi uma rede neural convolucional supervisionada simples, criada usando o modelo sequencial da biblioteca Keras. O uso de uma rede convolucional foi feito devido a sua capacidade de lidar com imagens, sendo feito o uso de um modelo simples pelo fato da proposta do trabalho ser o desenvolvimento de um protótipo e não possuir o foco na criação de uma rede otimizada.

O modelo sequencial do Keras é apropriado para uma pilha simples de camadas, onde cada camada possui exatamente um tensor de entrada e um tensor de saída, sendo criado para permitir que o usuário construa modelos rapidamente e sem o uso de muito código.

Foi usado um modelo básico porém capaz de obter mais de 99% de acurácia no dataset MNIST (KERASTEAM, 2020). Ele possui a seguinte estrutura:

- Uma camada de convolução com a função de ativação ReLu.
- Uma camada de pooling por valor máximo.
- Uma camada de convolução com a função de ativação ReLu.
- Uma camada de pooling por valor máximo.
- Uma camada de operação flatten, que torna uma entrada com multiplas dimensões em uma saída de uma dimensão.
- Uma camada de operação dropout, para reduzir o overfitting.
- Por fim, uma camada totalmente conectada, com a função de ativação softmax. Ela é responsável por gerar as predições multiclasse.

Para o treinamento foi utilizado otimizador adam, a função de loss categorical_crossentropy e o melhor modelo foi escolhido pela métrica de menor loss. Foram feitas 40 épocas de treinamento (número de vezes que o modelo aprende usando o dataset) com o batch_size de 16. A quantidade

de épocas escolhida se deve ao tamanho pequeno do dataset, para evitar o treinamento de uma rede que terá dificuldades com a generalização

A biblioteca Keras disponibiliza um método chamado Model.summary(), que exibe a quantidade de parâmetros aprendidos pelo modelo em cada camada. A tabela 3.2 detalha a saída desse método para o modelo trainado.

Tabela 3.2: Saída do metodo Model.summary()

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1	(None, 13, 13, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dropout (Dropout)	(None, 2304)	0
dense (Dense)	(None, 6)	13830
Total params: 33,222 Treinable params: 33,222 Non-treinable params: 0		

3.4 Levantamento de Requisitos

Nesta seção serão apresentados os requisitos funcionais e não funcionais definidos para o programa EYEASSiSTIVE.

3.4.1 Requisitos Funcionais

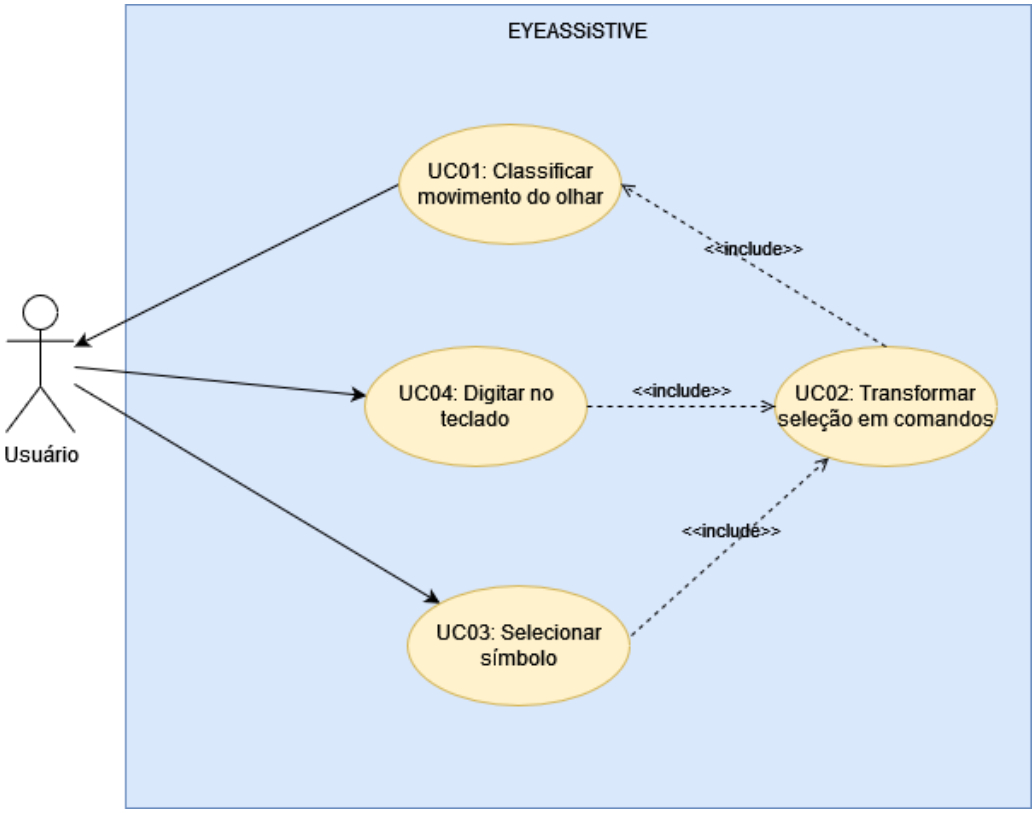
- RF01 - O programa irá realizar a captação de imagens pela webcam do usuário.
- RF02 - O programa irá realizar a detecção de rostos e classificação do olhar com as imagens captadas.
- RF03 - O usuário poderá usar a visão para selecionar símbolos para se expressar.
- RF04 - O usuário poderá usar a visão para selecionar teclas em um teclado virtual.

3.4.2 Requisitos Não-Funcionais

- RNF01 - O programa terá uma interface simples para facilitar a usabilidade.
- RNF02 - O programa irá captar imagens e realizar classificações por pelo menos 1 segundo para que possa realizar o movimento desejado pelo usuário.
- RNF03 - O programa será desenvolvido na linguagem Python.
- RNF04 - O programa necessita que o usuário possua uma webcam conectada ao computador.
- RNF05 - O programa será desenvolvido primariamente para o sistema operacional Linux, porém usará bibliotecas multiplataforma permitindo que seja usando em outros sistemas operacionais.
- RNF06 - O programa será projetado para ser usado em telas com resolução 1920 x 1080.

3.5 Diagrama de Casos de Uso

Figura 3.4: Diagrama de Casos de Usos da Aplicação



3.5.1 Descrição dos Casos de Uso

Tabela 3.3: Descrição do Caso de Uso UC01

Identificação-Nome	UC01: Classificar movimento do olhar
Descrição	O sistema irá fazer detecção de rosto e realizar a classificação da direção do olhar.
Ator	Usuário

Fluxo Principal	1. Inicia automaticamente quando o programa é executado. 2. O sistema captura as imagens da webcam. 3. É feita a detecção de rosto e o pré-processamento. 4. A imagem resultante é passada para o modelo e classificada.
Extensão	Não se aplica.
Inclusão	Não se aplica.
Pós-condição	É gerada uma classificação para o movimento.
Regras de Negócio	Apenas uma pessoa pode estar na captura da webcam. É preciso que o ambiente esteja bem iluminado.

Tabela 3.4: Descrição do Caso de Uso UC02

Identificação-Nome	UC02: Transformar classificação em comandos
Descrição	O sistema usará a classificação para gerar comandos que controlam as telas.
Ator	Sistema

Fluxo Principal	1. Inicia após a coleta de 1.5 segundos de classificações. 2. É feita contagem da classificação mais comum durante esse período. 3. Essa classificação é transformada em um acionamento de tecla correspondente.
Extensão	Não se aplica.
Inclusão	UC01: Classificar movimento do olhar.
Pós-condição	É feito um acionamento de um botão para controle das telas.
Regras de Negócio	Não se aplica.

Tabela 3.5: Descrição do Caso de Uso UC03

Identificação-Nome	UC03: Selecionar símbolo
Descrição	O usuário seleciona um dos símbolos disponíveis.
Ator	Usuário

Fluxo Principal	1. O usuário é apresentado a três grupos de símbolos. 2. Ele irá olhar para a direção que contém o símbolo que deseja. 3. Serão exibidos os símbolos de grupo selecionado e um botão para retornar. 4. Ele irá olhar para a direção que contém o que deseja selecionar. 5. Caso seja um símbolo, ele será exibido em destaque. Caso seja o botão de retornar, é exibida a tela anterior.
Extensão	Não se aplica.
Inclusão	UC03: Transformar classificação em comandos.
Pós-condição	É exibido o símbolo selecionado.
Regras de Negócio	Não se aplica.

Tabela 3.6: Descrição do Caso de Uso UC04

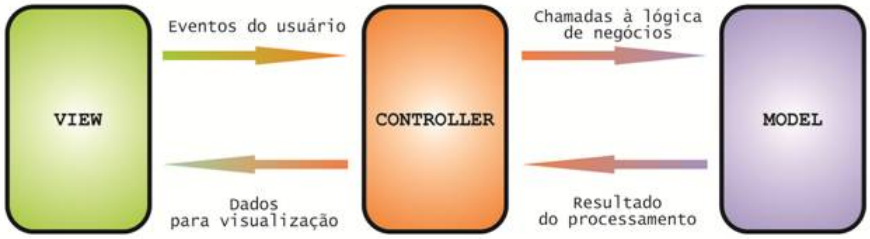
Identificação-Nome	UC04: Digitar no teclado
Descrição	O usuário seleciona um dos símbolos disponíveis.
Ator	Usuário

Fluxo Principal	1. O usuário seleciona a opção do teclado na tela de símbolos. 2. É exibido um teclado com um marcador de posição. 3. O usuário movimenta o marcador pelo olhar. 4. Ele seleciona uma tecla olhando para o centro da tela. 5. A digitação é feita no campo de texto na parte central do programa.
Extensão	Não se aplica.
Inclusão	UC03: Transformar classificação em comandos.
Pós-condição	O texto digitado pelo usuário é exibido.
Regras de Negócio	Não se aplica.

3.6 Arquitetura do Sistema

A arquitetura do programa EYEASSiSTIVE é divida em dois módulos principais: frontend e backend. A organização dos componentes foi baseada no modelo MVC (Model-View-Controller), tradicionalmente utilizada para o desenvolvimento de interfaces gráficas para desktops. Este padrão de projeto enfatiza a separação entre a lógica de negócio de um software e a sua exibição. Essa separação traz uma divisão de tarefas melhor e facilita a manutenção. Ele é separado em três camadas básicas, cada uma com funções bem definidas.(DEVMEDIA, 2011)

Figura 3.5: Padrão MVC



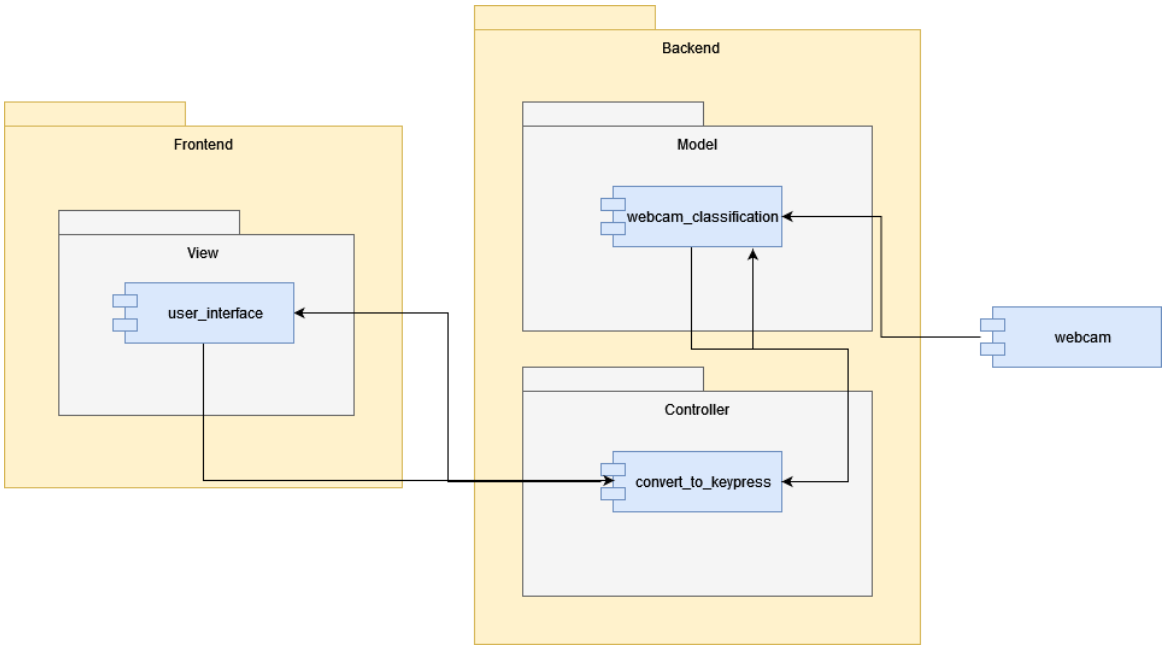
Fonte: (DEVMEDIA, 2011)

A camada Model (modelo) define que dado a aplicação deve conter, possuindo a lógica da aplicação. Caso esse dado mude, o modelo irá notificar a View, para que a exibição mude e às vezes o Controller, caso a mudança necessite de uma lógica diferente. A camada View (visão) define como os dados devem ser exibidos, representando a parte do sistema que interage com o usuário. Ela não contém a lógica de negócios, todo o processamento é feito na camada Model.

Já a camada Controller (controlador) contém a lógica que atualiza o Model e/ou o View de acordo com a entrada do usuário, agindo como uma intermediária entre as duas camadas. (CODEACADEMY, 2022)

A figura 3.6 retrata como ocorre a interação dos componentes do sistema seguindo o padrão de projeto MVC.

Figura 3.6: Diagrama de componentes



3.6.1 Frontend

O módulo frontend do projeto possui o pacote View com o componente UI, que contém a interface do sistema, desenvolvida usando a biblioteca tkinter. Este componente é responsável pela visualização e interação com o usuário. Ele também é responsável por manter a captura da webcam ligada durante o funcionamento do programa e envia estes dados para o Controller.

3.6.2 Backend

O módulo backend envolve os elementos que recebem dados e os transformam em comandos que são enviados para controlar o módulo de frontend. Nele estão contidos os pacotes de Model e Controller.

O pacote Model consiste no componente webcam_classification, onde é feito o uso do modelo de deep learning criado para gerar previsões por meio das imagens capturadas pela webcam. Ele obtém as imagens capturadas, faz a detecção de rostos e isola o olho direito, que por fim é classificado. O pacote Controller usa essa classificação no componente convert_to_keypress como entrada para transformá-la no pressionamento do botão corresponde por meio da biblioteca Pynput.

3.7 Projeto de Telas

Nas próximas seções será feita a exibição das telas do programa, apresentando suas funcionalidades.

3.7.1 Tela de Símbolos

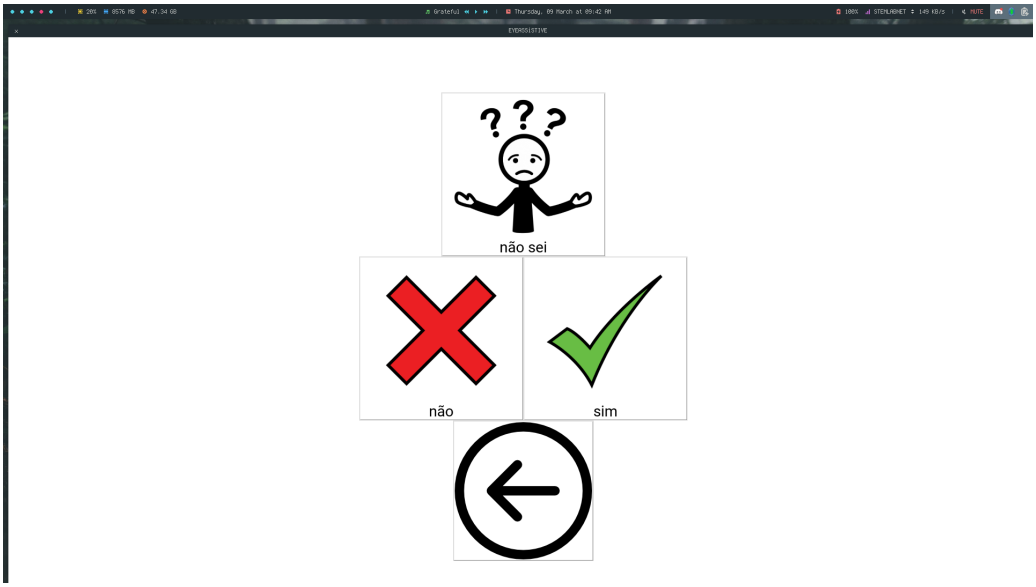
É a tela inicial do programa, sendo o seu foco principal. Nela o usuário pode selecionar um conjunto de símbolos por meio da direção que olha ou ir para a tela do teclado. As imagens dos símbolos foram obtidas no site ARAASAC, que oferece pictogramas para a criação de pranchas sob a licença Creative Commons.(ARASAAC, 2022)

Figura 3.7: Tela de Símbolos

3.7.2 Seleção de Símbolos

Após o usuário selecionar um conjunto, ele é levado para esta tela que contém um símbolo em cada direção e o botão de voltar para a tela principal. O usuário também pode retornar para a tela anterior mantendo os olhos fechados por alguns segundo.

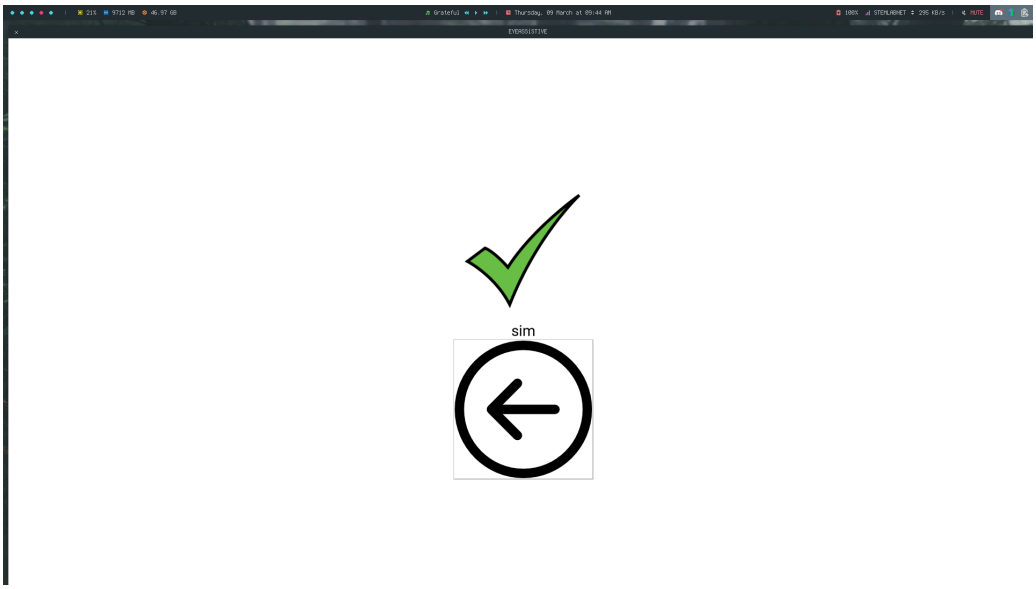
Figura 3.8: Seleção de Símbolos



3.7.3 Símbolo Selecionado

O símbolo selecionado na tela anterior é exibido em destaque, junto com um botão para retornar. Da mesma forma que a tela anterior, outra forma de retornar pode ser feita fechando os olhos durante alguns segundos.

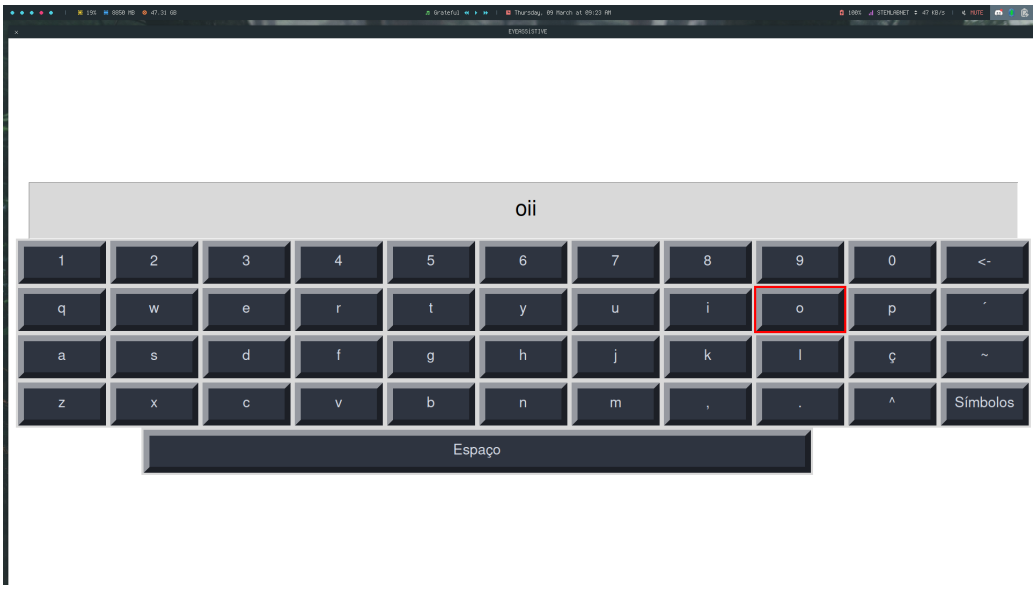
Figura 3.9: Símbolo Selecionado



3.7.4 Tela do Teclado

Nesta tela é exibido um teclado virtual com um marcador de seleção e uma caixa de texto. O marcador é movimentado de acordo com a direção do olhar e o pressionar da tecla ocorre quando o usuário olha para o centro da tela. Manter os olhos fechados tem a função de excluir o ultimo caractere inserido. O teclado também inclui um botão que leva o usuário para a tela de símbolos.

Figura 3.10: Tela do Teclado



3.8 Tecnologias Utilizadas

Nas próximas sessões serão introduzidas as principais tecnologias usadas no desenvolvimento deste projeto.

3.8.1 Jupyter

Jupyter é um projeto open-source que permite a combinação de texto em Markdown e code Python executável em um único arquivo chamado de notebook. Surgiu do projeto IPython em 2014, por Fernando Pérez e Brian Granger, que continua a existir como uma shell python e kernel para o Jupyter.

Ele foi criado para uso em computação interativa: o usuário pode criar células contendo código e pode executar apenas o código contido em uma única célula, sem necessidade de executar o código inteiro, facilitando a realização de testes.Além disso o usuário pode criar células que usam Markdown, permitindo a exibição de resultados e observações junto ao código. (PEREZ, 2014)

Jupyter Notebooks podem ser usados online por meio de ferramentas como Google Collaboratory e Amazon SageMaker, e o uso local pode ser feito usando extensões para integração

em ambientes de desenvolvimento como Visual Studio Code e PyCharm. (VSCODE, 2021) (PYCHARM, 2022)

3.8.2 Keras

Keras é uma biblioteca de código aberto que disponibiliza uma interface para o desenvolvimento de redes neurais artificiais. Tem como objetivo implementar uma API de alto nível que facilite o uso da biblioteca TensorFlow, auxiliando na criação de modelos protótipos. (KERASTEAM, 2022b)

O Keras possui suporte para redes neurais padrões, convolucionais e recorrentes, além de possuir camadas de utilidade comumente utilizadas como dropout, pooling e normalização em lote.

Também traz a possibilidade de escalonamento e multiplataforma: pode ser usado tanto em unidades de processamento de tensor quanto em unidades de processamento gráfico, e permite a exportação de modelos para serem executados em navegadores ou em dispositivos móveis.

Suas estruturas de dados mais básicas são as camadas e os modelos, com o tipo mais simples de modelo sendo o sequencial, construído por uma pilha linear de camadas. Para a criação de arquiteturas complexas, deve ser usado a API funcional do Keras, que permite a criação de grafos arbitrários de camadas e a criação de modelos do zero pelo uso de subclasssing. (KERASTEAM, 2022a)

3.8.3 OpenCV

OpenCV é uma biblioteca de código aberto disponível para várias linguagens de programação que tem como objetivo implementar diversas funções de visão computacional. Inicialmente foi desenvolvida pela Intel, atualmente tem seu suporte mantido por uma organização sem fins lucrativos chamada Open Source Vision Foundation. (OSVF, 2022)

A biblioteca reúne mais de algoritmos otimizados , incluindo ambos os modernos e clássicos nas áreas de visão computacional e aprendizagem de máquinas. Esses algoritmos podem ser usados para detectar e reconhecer faces, identificar objetos, classificar ações humanas, rastrear

movimento de câmera, rastrear movimento de objetos, entre outros. Ela possui interfaces para C++, Python, Java e MATLAB e suporta Windows, Linux, Android e MacOS e foi escrita nativamente em C++. (OPENCV, 2022)

No desenvolvimento do EYEASSiSTIVE, esta biblioteca foi usada para a recortagem, carregamento e redimensionamento de imagens do dataset. Também foi usada para o lidar com a utilização do vídeo capturado pela webcam.

3.8.4 Dlib

Dlib é uma biblioteca multiplataforma que contém algoritmos de aprendizagem de máquina e ferramentas para criação de softwares complexos.Teve o início de seu desenvolvimento em 2022 e foi criada usando a linguagem C++.

Atualmente Dlib possui a implementação de funções que lidam com aprendizagem de máquina, processamento de imagem, gerenciamento de threads, algoritmos numéricos, algoritmos de modelos de inferência gráficos, redes, interfaces gráficas, compressão de dados e integridade, testes e algoritmos de utilidade geral. (DLIB, 2022) A função de detecção de rostos e mapeamento de pontos faciais facilitou o isolamento da área dos olhos para a execução da classificação do olhar.

3.8.5 Tkinter

O pacote tkinter é a interface padrão da linguagem Python para conjunto de ferramentas de desenvolvimento de interfaces gráficas Tcl/Tk. Está disponível para a maior parte das plataformas Unix, macOS e também para sistemas Windows. (PYTHON, 2022) A criação da interface do EYEASSiSTIVE foi feita por meio desta biblioteca.

Seu funcionamento é baseado em widgets, implementam vários tipos de controles para interfaces como botões, caixas de texto e menus. O widget chamado Frame permite a organização de outros widgets, agindo como um contêiner. Cada widget possui atributos que podem ser editados para customização da aparência da interface, como dimensões, cores e fontes. (DEVMEDIA, 2016)

3.8.6 Pynput

Pynput é uma biblioteca para Python que permite o controle e o monitoramento de dispositivos de entrada. Atualmente possui suporte para mouses e teclados, separados em subpacotes para cada tipo. (PALMéR, 2015a)

Por meio do uso desta biblioteca é possível fazer com que certas teclas sejam pressionadas de acordo com a realização de determinadas ações, a digitação de texto sem o uso do teclado físico e monitoramento de teclas pressionadas. Para o mouse, é possível ser feita a movimentação para coordenadas específicas, relativas e o controle do clique e rolagem. (PALMéR, 2015b)

No desenvolvimento deste trabalho foi utilizado para que certas teclas sejam pressionadas de acordo com a classificação gerada pelo modelo treinado, assim permitindo que o usuário selecione elementos na interface.

Capítulo 4

Testes e Resultados Parciais

A plataforma usada no treinamento do modelo e execução do sistema proposto consiste em um notebook com o sistema operacional Arch Linux, 64-bit, com o processador Intel Core i7-9750H de 6 cores, 32 GB de memória RAM 2666 MHz e a placa gráfica Nvidia GeForce GTX 1660 Ti com 6 GB de memória. A webcam utilizada possui a resolução de 1280 x 720 pixels.

Para o desenvolvimento foram usadas a versão 3.9.12 da linguagem Python e a versão 2.9.0 da biblioteca Keras.

4.1 Resultados referentes ao desempenho do modelo

4.1.1 Tabela com as Métricas do Modelo

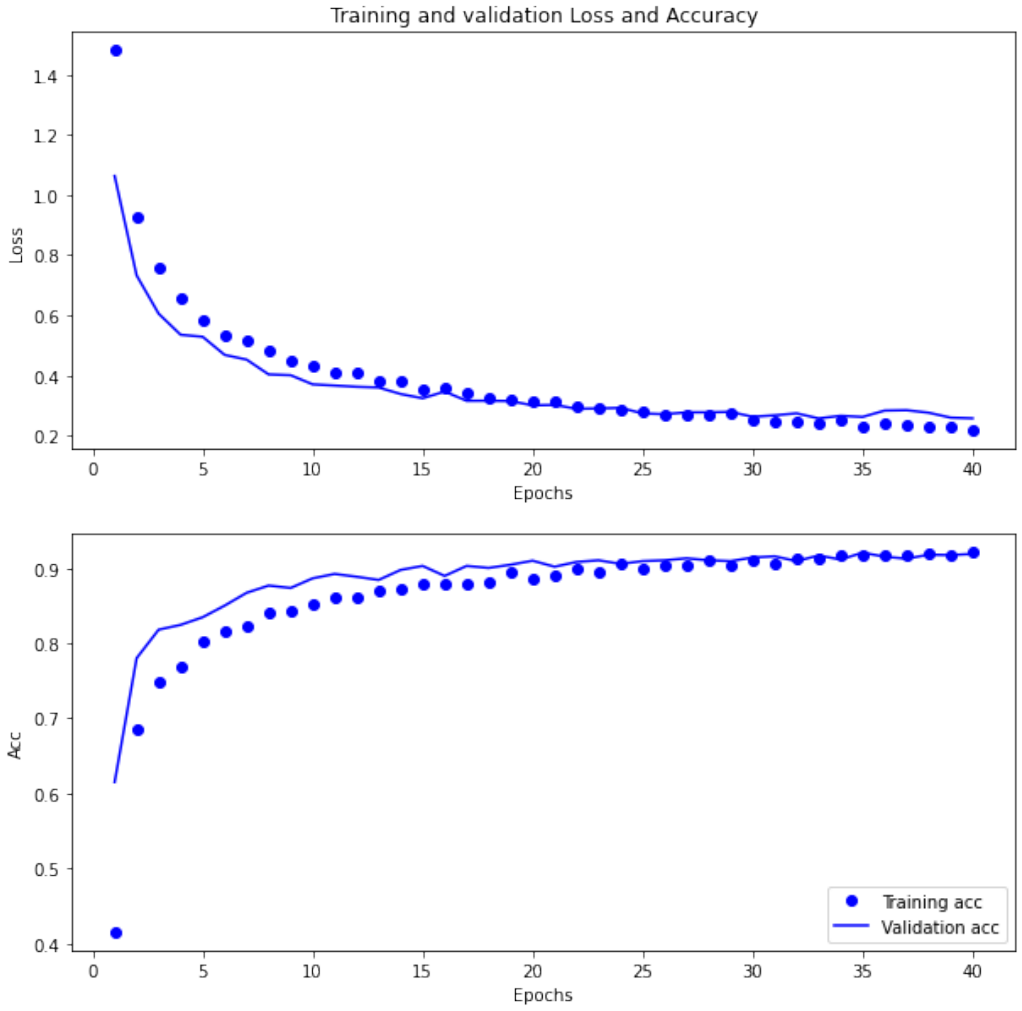
A tabela 4.1 apresenta o resultado das métricas obtidas pelo modelo e figura 4.1 detalha as mudanças do loss e da acurácia durante as épocas de treinamento.

Tabela 4.1: Saída da função classfication_report()

	precision	recall	f1-score	support (n. amostras)
Center	0.85	0.89	0.87	414
Up	0.95	0.91	0.93	249

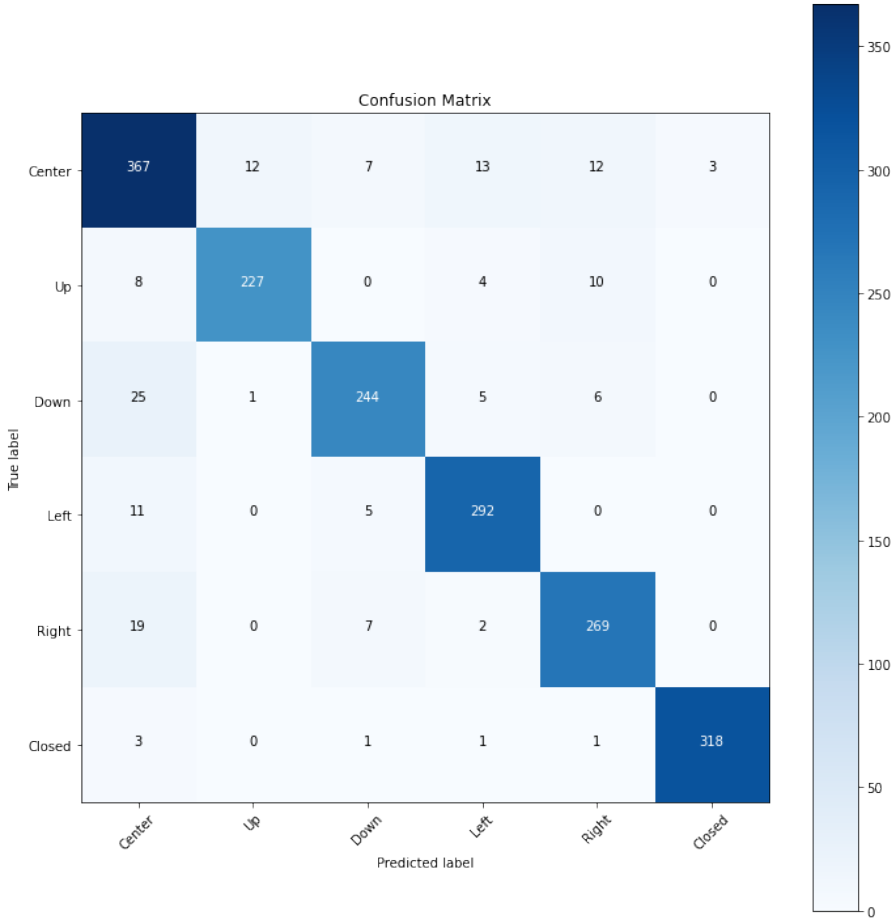
Down	0.92	0.87	0.90	281
Left	0.92	0.95	0.93	308
Right	0.90	0.91	0.90	297
Closed	0.99	0.98	0.99	324
accuracy			0.92	1873
macro avg	0.92	0.92	0.92	1873
weighted avg	0.92	0.92	0.92	1873

Figura 4.1: Histórico de Loss e Acurácia durante o treinamento



4.1.2 Matriz de Confusão

Figura 4.2: Matriz de Confusão



4.2 Resultados refentes ao teste do sistema

4.2.1 Teste com a câmera

Durante a etapa de desenvolvimento do dataset foram realizados testes iniciais de classificação das imagens da webcam. Inicialmente o dataset havia sido feito usando a técnica de usar imagens em escala de cinza para melhorar a performance da rede neural. Apesar dos testes de validação mostrarem resultados normais, os testes com a câmera obtiveram resultados ruins, com grandes erros de classificação.

Também foram realizados testes para verificar o funcionamento e a detecção usando a webcam. Com isso foi possível notar que o sistema necessita de um local que possua boa iluminação,

caso contrário a biblioteca de detecção facial não conseguirá reconhecer rostos.

Outro ponto notado foi a distância de webcam para o usuário. Ela precisa que o usuário esteja uma distância não muito perto e não muito longe, de aproximadamente 30 centímetros, para que o sistema consiga realizar as classificações corretamente.

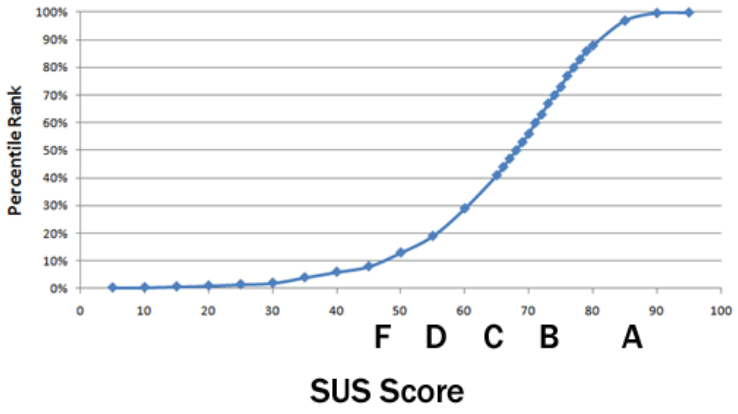
4.2.2 Teste de usabilidade do sistema

Foi usado o método SUS (System Usability Scale) para a criação do questionário de avaliação do sistema e a análise dos resultados. Ele é uma ferramenta padronizada amplamente utilizada para avaliar a usabilidade de sistemas, produtos ou serviços digitais. Foi criado por John Brooke em 1986 e é composto por um questionário que avalia a satisfação do usuário em relação a diferentes aspectos da usabilidade, como facilidade de uso, eficiência e aprendizagem. (BROOKE, 1995)

O questionário é composto por dez itens, cada um avaliado em uma escala de cinco pontos, que variam de ”discordo totalmente”a ”concordo totalmente”. As questões são projetadas para avaliar a percepção do usuário em relação à usabilidade geral do sistema, em vez de avaliar aspectos específicos de usabilidade, como a eficácia ou a eficiência. (BROOKE, 1995)

Termino de coleta das respostas, deve ser feitos alguns cálculos com a pontuação de cada pergunta para obter a nota final da aplicação. A pontuação média de SUS entre 500 estudos é 68. Uma pontuação da SUS acima de 68 seria considerada acima da média e qualquer pontuação abaixo de 68 é abaixo da média. (SAURO, 2011)

Figura 4.3: Gráfico de notas SUS



Fonte: (SAURO, 2011)

4.2.2.1 Descrição do teste

Foram convidadas sete voluntários para usar o sistema em um notebook. O objetivo e o funcionamento básico foram explicados para cada usuário antes do início do teste. Os usuários puderam testar a aplicação até se sentirem confortáveis para responder o questionário e puderam relatar suas opiniões durante e depois dos testes.

4.2.3 Avaliação dos resultados

Após o término do teste, os usuários responderam o questionário de usabilidade de acordo com sua experiência. A tabela 4.2 apresenta a distribuição dos resultados para cada pergunta, onde 1 corresponde a discordo completamente e 5 a concordo completamente.

Tabela 4.2: Resultado das avaliações SUS

Eu acho que o sistema tem um propósito útil.	5	5	5	5	5	5	5
Eu acho o sistema desnecessariamente complexo.	1	1	1	1	1	1	2
Eu achei o sistema fácil de usar.	2	5	5	5	4	5	4
Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.	1	1	4	1	1	5	1
Eu acho que as várias funções do sistema estão muito bem integradas.	4	5	5	5	5	3	4
Eu acho que o sistema apresenta muita inconsistência.	1	2	1	1	2	1	1
Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.	3	5	5	5	5	5	5
Eu achei o sistema atrapalhado de usar.	1	1	3	1	1	1	1
Eu me senti confiante ao usar o sistema.	3	4	5	5	4	5	3
Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.	1	1	1	1	1	1	1

Para obter a pontuação final do sistema com o método SUS, é necessário realizar alguns cálculos com o resultado das perguntas:

1. Para questões ímpares (positivas): subtraia 1 da resposta do usuário.

2. Para questões pares (negativas): subtraia as respostas do usuário de 5.

3. Some as respostas convertidas para cada usuário e multiplique esse total por 2,5. Isso converte a faixa de valores possíveis de 0 a 100, em vez de 0 a 40.

(BROOKE, 1995)

Com isso, o sistema obteve as seguintes pontuações: 80, 95, 87.5, 100, 92.5, 85, 87.5. Tirando a média dessas pontuações, o valor final atingido foi 89.6, classificando o sistema como acima da média.

Durante o teste e ao final do questionário, os usuários puderam relatar livremente as suas experiências durante o uso do EYEASSiSTIVE. Alguns falaram sobre a importância da criação de sistemas para a acessibilidade e citaram a possibilidade da utilização por outros grupos, como pessoas idosas.

Como críticas e sugestões, foi apontado um pouco de dificuldade inicial da movimentação no teclado e foi sugerida a possibilidade de permitir que o usuário customize o tempo de detecção para a geração de classificações.

Capítulo 5

Conclusão

Este trabalho descreveu o projeto de um sistema de comunicação baseado na detecção do olhar utilizando deep learning. Esse sistema irá permitir que pessoas com certos tipos de deficiências se comuniquem de forma rápida, com o painel de símbolos, ou uma comunicação mais específica, usando o painel de teclado.

Para o desenvolvimento do projeto, foi feita uma pesquisa inicial, para verificar se existem projetos ou produtos similares. Em seguida, foi realizado um levantamento bibliográfico a respeito de conceitos sobre acessibilidade, deep learning e visão computacional.

Com isso, os requisitos foram elaborados e diagramas foram criados para abstrair as funcionalidades do programa. Além disso, foram feitas pesquisas e estudos sobre as tecnologias empregadas no desenvolvimento da aplicação. Desta forma foi desenvolvido, sendo realizadas reuniões semanais com o professor orientador para o acompanhamento do progresso e sugestão de mudanças.

Após o desenvolvimento do programa foram realizados testes para encontrar e corrigir bugs, verificar se o funcionamento está correto e determinar em que áreas é possível realizar melhorias. Para os testes de usabilidade, convidados sete voluntários usar o sistema, relatar sua experiência e responder um questionário de avaliação. Foram recebidas boas avaliações e algumas sugestões de melhora.

5.1 Trabalhos Futuros

Um dos principais objetivos do desenvolvimento deste sistema é aumentar a disponibilidade de métodos de comunicação alternativos. Na versão atual é necessário o uso de um notebook ou desktop com webcam para que ele seja executado, mas isso pode dificultar o acesso para pessoas que não possuem esses dispositivos.

Com isso, será importante o desenvolvimento de versões que possam ser usadas em plataformas mais comuns, como celulares ou tablets Android. Para a criação dessas versões, é necessário a adaptação do modelo de deep learning criado para que possa funcionar em sistemas mais leves. Além disso, outra funcionalidade adicional seria permitir que o usuário customize os símbolos selecionáveis da aplicação.

Referências Bibliográficas

ALTEXSOFT. *Key Machine Learning Metrics to Evaluate Model Performance | AltexSoft*. 2022. Disponível em: <https://www.altexsoft.com/blog/machine-learning-metrics/>.

ALTVATER, A. *What is Scrum Software Development? How It Works, Best Practices & More*. 2017. Disponível em: <https://stackify.com/what-is-scrum/>.

ALVA, M. et al. An image based eye controlled assistive system for paralytic patients. In: *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*. [S.l.: s.n.], 2017. p. 178–183.

ARASAAC. *Biblioteca de símbolos e recursos para Comunicação Aumentativa e Alternativa (CAA)*. 2022. Disponível em: <https://arasaac.org/>.

ASANA. *Ainda não conhece o Scrum? Descubra o que é e por que funciona tão bem • Asana*. 2022. Disponível em: <https://asana.com/pt/resources/what-is-scrum>.

ASSISTIVE, L. *I-Series Gaze Interaction*. 2023. Disponível em: <https://www.linkassistive.com/product/i-series-gaze-interaction/>.

BABICH, N. *What is Computer Vision & How Does it Work? An Introduction*. 2020. Disponível em: <https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/>.

BAHETI, P. *Activation Functions in Neural Networks [12 Types & Use Cases]*. 2022. Disponível em: <https://www.v7labs.com/blog/neural-networks-activation-functions,https://www.v7labs.com/blog/neural-networks-activation-functions>.

BAJAJ, A. *Performance Metrics in Machine Learning [Complete Guide]*. 2021. Disponível em: <https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide>.

BARELLI, F. *Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV*. [S.l.]: Casa do Código, 2018.

BERSCH, R. *Introdução à Tecnologia Assistiva*. 2017. Disponível em: <http://inf.ufes.br/~zegonc/material/Comp_Sociedade/ZEGONC_Tecnologias_Assistivas_Livro_Introducao_TA.pdf>.

BOUVRIE, J. Notes on convolutional neural networks. Citeseer, 2006.

BROOKE, J. SUS: A quick and dirty usability scale. *Usability Eval. Ind.*, v. 189, nov. 1995.

BROWNLEE, J. *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*. 2019. Disponível em: <https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/>.

C3.AI. *Holdout Data*. 2022. Disponível em: <https://c3.ai/glossary/data-science/holdout-data/>.

CERON, R. *A Inteligência Artificial hoje: dados, treinamento e inferência*. 2020. Disponível em: <https://www.ibm.com/blogs/systems/br-pt/2020/01/a-inteligencia-artificial-hoje-dados-treinamento-e-inferencia/>.

CODEACADEMY. *MVC: Model, View, Controller*. 2022. Disponível em: <https://www.codecademy.com/article/mvc>.

DETTMERS, T. *Deep Learning in a Nutshell: Core Concepts*. 2015. Disponível em: <https://developer.nvidia.com/blog/deep-learning-nutshell-core-concepts/>.

DEVMEDIA. *Padrão MVC (Model-View-Controller) Tutorial*. 2011. Disponível em: <https://www.devmedia.com.br/padrao-mvc-java-magazine/21995>.

DEVMEDIA. *Tkinter: Interfaces gráficas em Python*. 2016. Disponível em: <https://www.devmedia.com.br/tkinter-interfaces-graficas-em-python/33956>.

DLIB. *dlib C++ Library*. 2022. Disponível em: <http://dlib.net/>.

FIRMIN, S. *Holdouts and Cross Validation: Why the Data Used to Evaluate your Model Matters*. 2019. Disponível em: <https://community.alteryx.com/t5/Data-Science/Holdouts-and-Cross-Validation-Why-the-Data-Used-to-Evaluate-your/ba-p/448982>.

GHANI, M. U. et al. GazePointer: A real time mouse pointer control implementation based on eye gaze tracking. In: *INMIC*. Lahore, Pakistan: IEEE, 2013. p. 154–159. ISBN 9781479930432. Disponível em: <http://ieeexplore.ieee.org/document/6731342/>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. Google-Books-ID: Np9SDQAAQBAJ. ISBN 9780262035613.

HASSAN, M. A. Digital Camera-based Eye Movement Assessment Method for NeuroEye Examination. TechRxiv, dez. 2022. Disponível em: <https://www.techrxiv.org/articles/preprint/Digital_Camera-based_Eye_Movement_Assessment_Method_for_NeuroEye_Examination/21767825/1>.

HERFF, C. et al. Brain-to-text: decoding spoken phrases from phone representations in the brain. *Frontiers in Neuroscience*, v. 9, 2015. ISSN 1662-453X. Disponível em: <https://www.frontiersin.org/articles/10.3389/fnins.2015.00217>.

IBGE. *PNS 2019: país tem 17,3 milhões de pessoas com algum tipo de deficiência | IBGE*. 2021. Disponível em: <https://censos.ibge.gov.br/2013-agencia-de-noticias/releases/31445-pns-2019-pais-tem-17-3-milhoes-de-pessoas-com-algum-tipo-de-deficiencia.html>.

INCLUSIVE. *EyeOn Elite for AAC*. 2023. Disponível em: <https://www.inclusive.com/uk/eyeon-elite-for-aac.html>.

KARIDIS, A. New tool allows paralyzed to speak — using their eyes. *Washington Post*, abr. 2016. ISSN 0190-8286. Disponível em: <https://www.washingtonpost.com/national/health-science/new-tool-allows-paralyzed-to-speak--using-their-eyes/2016/04/04/a1dc1d98-da50-11e5-925f-1d10062cc82d_story.html>.

KELLEHER, J. *Deep Learning*. [S.l.]: MIT Press, 2019. ISBN 9780262354899.

KERASTEAM. *Keras documentation: Simple MNIST convnet*. 2020. Disponível em: <https://keras.io/examples/vision/mnist_convnet/>.

KERASTEAM. *Keras: Deep Learning for humans*. 2022. Original-date: 2015-03-28T00:35:42Z. Disponível em: <https://github.com/keras-team/keras>.

KERASTEAM. *Keras documentation: About Keras*. 2022. Disponível em: <https://keras.io/about/>.

KRAFKA, K. et al. *Eye Tracking for Everyone*. [S.l.], 2016. ArXiv:1606.05814 [cs] type: article. Disponível em: <http://arxiv.org/abs/1606.05814>.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, maio 2015. ISSN 0028-0836, 1476-4687. Disponível em: <http://www.nature.com/articles/nature14539>.

MADAN, A. *Eyes- Open or Closed*. 2021. Disponível em: <https://www.kaggle.com/datasets/akshitmadan/eyes-open-or-closed>.

MCDERMOTT, J. *Convolutional Neural Networks — Image Classification w. Keras*. 2022. Disponível em: <https://www.learndatasci.com/tutorials/convolutional-neural-networks-image-classification/>.

MISHRA, A. *Metrics to Evaluate your Machine Learning Algorithm | by Aditya Mishra | Towards Data Science*. 2018. Disponível em: <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>.

OMPI, O. M. d. P. I. *Mais de 1 bilhão de pessoas precisam de apoio de tecnologias em sua vida diária*. 2021. Disponível em: <https://news.un.org/pt/story/2021/03/1745282>.

OPENCV. *About*. 2022. Disponível em: <https://opencv.org/about/>.

OSVF. *The Open Source Vision Foundation*. 2022. Disponível em: <https://osvf.org/>.

PALMéR, M. *pynput: Monitor and control user input devices*. 2015. Disponível em: <https://github.com/moses-palmer/pynput>.

PALMéR, M. *pynput Package Documentation*. 2015. Disponível em: <https://pynput.readthedocs.io/en/latest/index.html>.

PATIL, P. V. *Eye Dataset*. 2021. Disponível em: <https://www.kaggle.com/datasets/prasadvpatil/eye-dataset>.

PEREZ, F. *Project Jupyter*. 2014. Disponível em: <https://speakerdeck.com/fperez/project-jupyter>.

PUBLICATIONS, M. *Computer Vision Pipeline, Part 1: the big picture*. 2019. Disponível em: <https://freecontent.manning.com/computer-vision-pipeline-part-1-the-big-picture/>.

PYCHARM. *Jupyter notebook support*. 2022. Disponível em: <https://www.jetbrains.com/help/pycharm/jupyter-notebook-support.html>.

PYTHON. *tkinter — Python interface to Tcl/Tk*. 2022. Disponível em: <https://docs.python.org/3/library/tkinter.html>.

ROCHA, M. d. C. C. E. F. Reflexões sobre recursos tecnológicos: ajudas técnicas, tecnologia assistiva, tecnologia de assistência e tecnologia de apoio. *Revista de Terapia Ocupacional da Universidade de São Paulo*, v. 16, n. 3, p. 97–104, set. 2005. ISSN 2238-6149. Disponível em: <https://www.revistas.usp.br/rto/article/view/13968>.

ROSEBROCK, A. *Convolutional Neural Networks (CNNs) and Layer Types*. 2021. Disponível em: <https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/>.

SARTORETTO, R. B. M. L. *O que é Tecnologia Assistiva?* 2022. Disponível em: <https://www.assistiva.com.br/tassistiva.html>.

SAURO, J. *Measuring Usability with the System Usability Scale (SUS) – MeasuringU*. 2011. Disponível em: <https://measuringu.com/sus/>.

SHAH, K. *Eye-dataset*. [S.l.]: Kaggle, 2020.

SRIVASTAVA, N. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014. ISSN 1533-7928. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>.

SYSTEMS, E. D. *EyeOn Eye Tracking Platform*. 2022. Disponível em: <https://eyetechds.com/eye-tracking-products/the-eyeon-platform/>.

TOBIIDYNAVOX. *I-Series AP*. 2022. Disponível em: <https://us.tobiidynavox.com/pages/i-series-ap>.

VARONA, J.; MANRESA-YEE, C.; PERALES, F. J. Hands-free vision-based interface for computer accessibility. *Journal of Network and Computer Applications*, v. 31, n. 4, p. 357–374, nov. 2008. ISSN 10848045. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1084804508000210>.

VILLANUEVA, A. et al. Hybrid method based on topography for robust detection of iris center and eye corners. *ACM Transactions on Multimedia Computing, Communications, and Applications*, v. 9, n. 4, p. 25:1–25:20, ago. 2013. ISSN 1551-6857. Disponível em: <https://doi.org/10.1145/2501643.2501647>.

VOULODIMOS, A. et al. Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, Hindawi Limited, v. 2018, p. 1–13, 2018. Disponível em: <https://doi.org/10.1155/2018/7068349>.

VSCODE. *Working with Jupyter Notebooks in Visual Studio Code*. 2021. Disponível em: <https://code.visualstudio.com/docs/datascience/jupyter-notebooks>.

WOOD, E. et al. Rendering of eyes for eye-shape registration and gaze estimation. In: *Proc. of the IEEE International Conference on Computer Vision (ICCV 2015)*. [S.l.: s.n.], 2015.

WOOD, T. *Activation Function*. 2020. Disponível em: <https://deepai.org/machine-learning-glossary-and-terms/activation-function>.