UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUCAS ANDRÉ WALTER

LEITORA DE TESTES RÁPIDOS DO TIPO FLUXO LATERAL UTILIZANDO MACHINE LEARNING

CURITIBA

2023

LUCAS ANDRÉ WALTER

LEITORA DE TESTES RÁPIDOS DO TIPO FLUXO LATERAL UTILIZANDO MACHINE LEARNING

Lateral flow immunoassays test reader using machine learning

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador(a): Rafael Eleodoro de Goes

CURITIBA

2023

LUCAS ANDRÉ WALTER

LEITORA DE TESTES RÁPIDOS DO TIPO FLUXO LATERAL UTILIZANDO MACHINE LEARNING

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 31/maio/2023

_____

Rafael Eleodoro de Goes
Doutor
Universidade Tecnológica Federal do Paraná

_____

André Eugenio Lazzaretti
Doutor
Universidade Tecnológica Federal do Paraná

_____

Luiz Fernando Copetti
Mestre
Universidade Tecnológica Federal do Paraná

CURITIBA

2023

Dedico este trabalho àqueles que participaram
dessa jornada.

AGRADECIMENTOS

Uso este parágrafo para agradecer a todos que de alguma forma contribuíram nessa jornada, em especial aos amigos e familiares. Agradeço também ao meu orientador Prof. Dr. Rafael Eleodoro de Goes, pela ajuda e prestatividade durante o desenvolvimento desse trabalho e por ter compartilhado seus conhecimentos e sabedoria durante essa etapa.

Agradeço também aos pesquisadores do LaCTAS que auxiliaram no desenvolvimento desse projeto com dados e material e que sem eles não teria sido possível.

Eu não temo os computadores.
Eu temo a ausência deles.
(ASIMOV; ISAAC, 1980).

RESUMO

Os imunoensaios de fluxo lateral são testes simples e de resultado rápido que permitem aplicações em larga escala sem a necessidade de infraestruturas complexas ou processos laboratoriais. Esse tipo de teste entrou em evidência com a recente pandemia de Covid-19, mas sua área de aplicação é muito mais ampla e pode ir de detecção de contaminantes em alimentos à medicina veterinária. Caracterizados pelo resultado apontado através de listras que surgem ao depositar uma amostra no teste, o resultado é dado de forma quantitativa (onde a intensidade das listras aponta a concentração de um antígeno ou contaminante na amostra sendo analisada) ou qualitativa (onde se estabelece um limiar de corte e o resultado pode ser reagente ou não reagente). A análise desse tipo de teste a olho nu pode ser desafiadora, em alguns casos podendo levar a erros humanos, além de ser uma tarefa dispendiosa de tempo para profissionais da saúde quando utilizados em larga escala. Neste trabalho, foi desenvolvida uma ferramenta capaz de exibir o resultado de tais testes sem a necessidade de interferência humana, através de uma leitora que se utiliza de uma câmera para gerar imagens e através de uma rede neural artificial previamente treinada, classifica os testes rápidos conforme sua respectiva doença de ensaio e exibe os resultados através de uma interface gráfica de usuário.

Palavras-chave: Classificadora de Imunoensaios de fluxo lateral; Redes neurais artificiais; Processamento de imagens; *Machine learning*.

ABSTRACT

Lateral flow immunoassays are simple, fast-resulting tests that allow large-scale applications without the need for complex infrastructure or laboratory processes. This type of test came to prominence with the recent Covid-19 pandemic, but its scope is much broader and can range from detecting contaminants in food to veterinary medicine. Characterized by the result indicated through stripes that appear when depositing a sample in the immunoassays, the result is given in a quantitative way (where the intensity of the stripes indicates the concentration of an antigen or contaminant in the sample being analyzed) or qualitatively (where a threshold is established, and the result can be reactive or non-reactive). Analyzing this type of test with the naked eye can be challenging, in some cases leading to human errors, in addition to being a time-consuming task for health professionals when used on a large scale. In this undergraduate thesis, a tool was developed capable of displaying the results of such tests without the need for human interference, through a reader that uses a camera to generate images and, through a previously trained artificial neural network, classifies the rapid immunoassays according to their respective test disease and displays the results through a graphical user interface.

Keywords: Lateral flow immunoassays classifier; Artificial neural networks; Image processing; Machine learning.

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D	2 Dimensões
3D	3 Dimensões
API	Interface de Programação de Aplicação
ANN	Rede Neural Artificial
CNN	Rede Neural Convolucional
FC	*Fully Connected*
FN	Falso Negativo
FP	Falso Positivo
FPS	*Frames* per Second
GPIO	General Purpose Input/Output
GPU	Unidade de Processamento Gráfico
GUI	Graphical User Interface
hCG	Gonadotrofina Coriônica Humana
HIV	Vírus da Imunodeficiência Humana
IA	Inteligência Artificial
IoT	Internet das Coisas
LaCTAS	Laboratório de Ciências e Tecnologias Aplicadas em Saúde
LCD	Liquid Crystal Display
LED	Diodo Emissor de Luz
LFIA	Imunoensaios de Fluxo Lateral
MP	Megapixels
NC	Nitrocelulose
OS	Sistema Operacional
PWM	Pulse Width Modulation
SDRAM	Memória Dinâmica de Acesso Aleatório Sincronizado
SMD	Surface Mount Device
SUS	Sistema Único de Saúde
TF	TensorFlow
TK	Tkinter
TN	Verdadeiro Negativo
TP	Verdadeiro Positivo

LISTA DE SÍMBOLOS

Ω	Ohm
$g(i,j)$	Função de duas variáveis
m	Linhas
n	Colunas
H	Matriz
T	Limiar
Hz	Hertz
Mm	milímetros
p	Pixel
V	Volt

SUMÁRIO

1 INTRODUÇÃO

Os testes rápidos de fluxo lateral são amplamente utilizados para aplicações clínicas, veterinárias, segurança alimentar e diversas outras áreas onde diagnósticos rápidos e precisos da presença de antígenos podem ser necessários. Na área clínica, tem grande importância na detecção de doenças infecciosas como o HIV, malária e mais recentemente o Covid-19.

Os testes rápidos tiveram um grande protagonismo na recente pandemia de Covid-19, onde possibilitaram testar a população em larga escala e oferecendo resultados imediatos que ajudaram a diminuir novos contágios da doença ao possibilitar que pessoas infectadas pudessem ser identificadas e tomassem providencias como o isolamento social. Esse grau de efetividade dificilmente poderia ser obtido com outros tipos de testes como os moleculares, onde o resultado pode levar até 48 horas para ser divulgado.

1.1 Motivação

Uma classificadora de testes rápidos permite automatizar a interpretação desse tipo de exame, aumentando a confiabilidade na leitura dos resultados e minimizando diagnósticos incorretos decorrente de erros humanos e discrepância entre as leituras realizados por diferentes profissionais da saúde. Além disso, uma leitora também possibilita um aumento na agilidade para aplicação de testes em grande número de pessoas, como no caso de pandemias, e integrar a leitora a outros sistemas que permitam automatizar o processo de registro e divulgação de resultados aos pacientes.

1.2 Objetivos gerais

Desenvolver um sistema capaz de realizar leitura de testes rápidos de fluxo lateral utilizando redes neurais artificiais para a classificação de imagens, gerando resultados sem interferência humana.

1.3 Objetivos específicos

- Obtenção de banco de imagens classificadas por profissionais da saúde.
- Tratamento das imagens do banco removendo ruídos e realizando a extração de características a serem utilizadas.

- Particionamento do banco de imagens para o treinamento, validação e testes de uma rede neural artificial.
- Integração ao sistema de uma câmera fotográfica para obtenção de imagens de testes rápidos inseridos na leitora.
- Montagem de um protótipo de uma leitora de testes rápidos.
- Definição da melhor forma de iluminação do interior da leitora, otimizando a capacidade de diagnóstico.
- Desenvolvimento de interface com o usuário para a leitora.
- Salvar resultados obtidos possibilitando que sejam exportados e utilizados em outras aplicações.

2 FUNDAMENTAÇÃO TEÓRICA

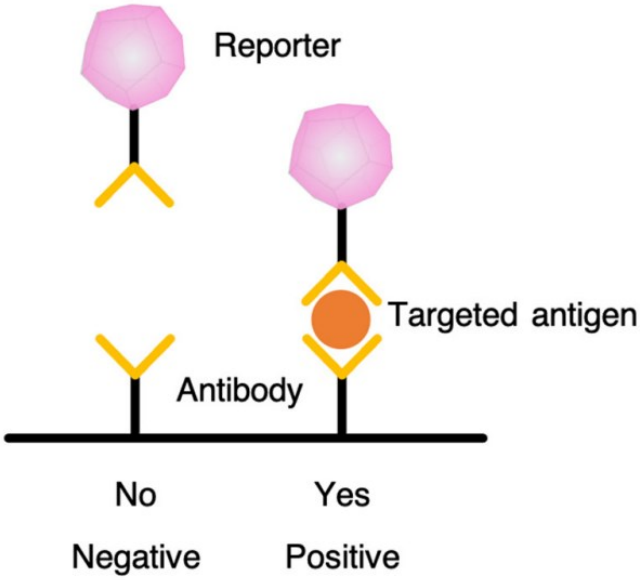
2.1 Imunoensaios de fluxo lateral

Os imunoensaios de fluxo lateral, do inglês *Lateral Flow Immunoassays* (LFIA) são uma tecnologia estabelecida e amplamente utilizada para detecção qualitativa e semiquantitativa de analitos como anticorpos, antígenos, parasitas, bactérias etc. Como apontam os autores [Hsiao et al., 2021], os LFIA apareceram pela primeira vez no início de 1980 e passaram a ser comercializados em 1984 com o primeiro produto para um teste de gravidez baseado em urina através da detecção de coriônica humana gonadotrofina (hCG). Desde então, milhares de dispositivos LFIA foram desenvolvidos e aplicados para o diagnóstico e prognóstico de várias doenças, tumores, patógenos, pesticidas, toxinas e íons metálicos.

Nos LFIA, uma amostra que pode ser de saliva, sangue ou outros materiais, é depositada na membrana de amostra do cassete, e então flui através de difusão capilar pela estrutura, passando por diferentes camadas com propriedades químicas e mecânicas. A detecção é feita através de um material chamado de marcador ou *reporter*, que são partículas que são misturadas ao analito e geram uma ligação entre as moléculas, chamada de conjugado. O acúmulo dessas partículas forma as características linhas de resultado em um teste rápido de fluxo lateral.

O método de detecção dos LFIA utilizados para esse trabalho, são do tipo anticorpo-antígeno, no qual a amostra é composta por material biológico que pode conter o antígeno de interesse. Anticorpos como os gerados pelo corpo humano para combater uma infecção podem ligar-se a antígenos específicos, e essa propriedade é explorada nos testes rápidos com a presença de anticorpos em regiões do teste que fixam o conjugado de antígeno e material marcador presente no analito. Na figura 1, podemos visualizar esse esquema, onde o antígeno, material marcador e anticorpo presente nas listras do teste de fluxo lateral se unem e indicam a presença da doença na amostra coletada.

Figura 1 - Processo de detecção LFIA



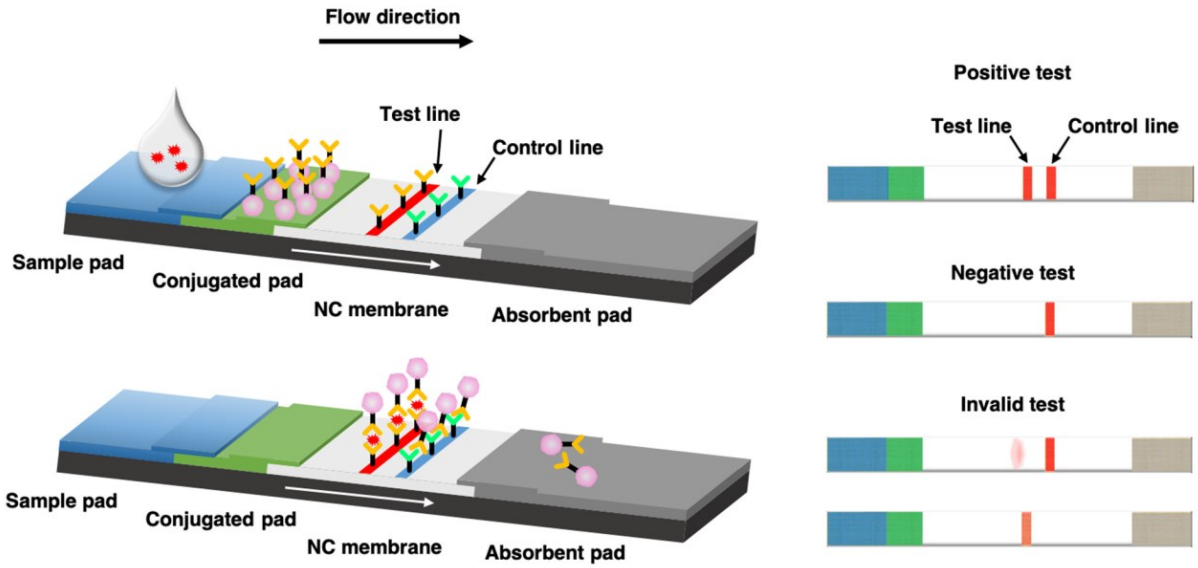
Fonte: Adaptado de Hsiao et al., (2021)

As partes que compõem um imunoensaios de fluxo lateral e suas funções dentro do teste estão descritas a seguir:

- Membrana de amostra: Parte onde a amostra é depositada, geralmente feita de materiais secos como fibras de celulose onde há a absorção e filtragem do material depositado;
- Membrana do conjugado: Região por onde o analito flui e reage com material do marcador;
- Membrana de nitrocelulose: Região onde as bandas de teste e de controle ficam contidas;
- Linha de teste: Região onde há a presença de anticorpos que fixam a solução de analito e marcador, fazendo com que haja concentração do material na região, possibilitando a visualização do resultado;
- Linha de controle: Segunda linha disposta alguns milímetros após a linha de teste, onde o excesso de material é fixado e atesta o correto fluxo de fluido no teste;
- Membrana de absorção: Absorve o excedente de fluxo que alcançam o final do teste;

Na figura 2 podemos visualizar o esquema de um teste rápido de fluxo lateral contendo as partes anteriormente descritas.

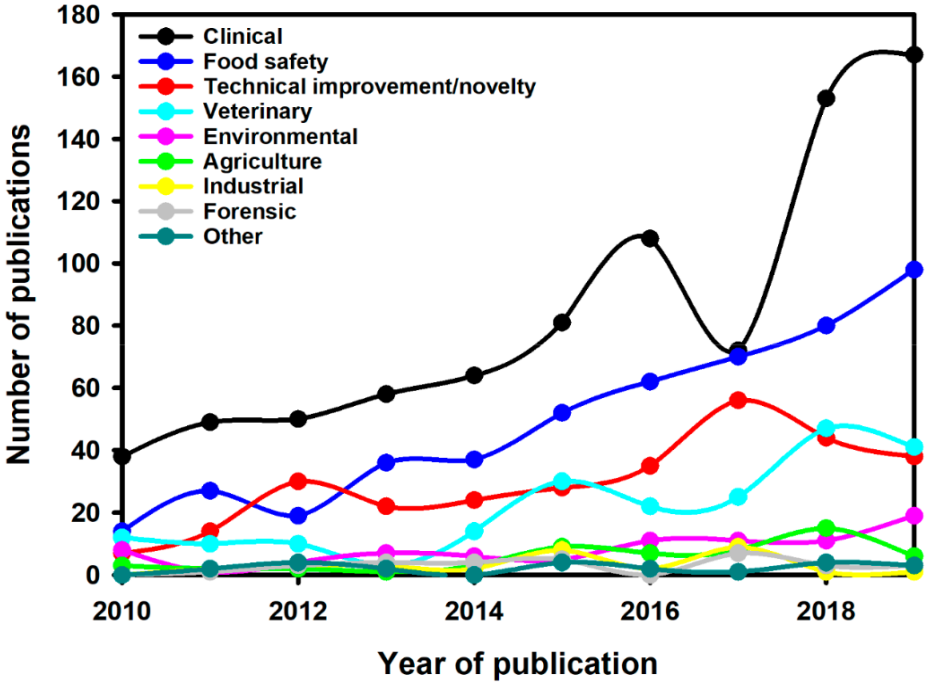
Figura 2 - Funcionamento de um LFIA



Fonte: Hsiao et al., (2021)

Os imunoensaios de fluxo lateral têm aplicação em diversas áreas. Em um recente estudo, [Di Nardo et al.,2021], os autores utilizaram o banco de dados de resumos e citações de artigos “Scopus”, para analisar como os artigos relacionados a aplicações de LFIA e como eles se distribuem nas principais áreas de uso da tecnologia. Os autores encontraram 2078 artigos publicados entre 2010 e 2019 e esses artigos foram classificados nas seguintes áreas: aprimoramento tecnológico, clínico, segurança alimentar, medicina veterinária, meio ambiente, agricultura, industrial, forense e outros. Na figura 3 são apresentados números de estudos e suas respectivas áreas conforme classificados pelos autores, exibindo o crescimento de importância dada a esse tipo de teste e sua ampla gama de atuação.

Figura 3 - Publicações por ano



Fonte: Di Nardo et al (2021)

2.1.1 Leitoras comerciais de imunoensaios de fluxo lateral

As leitoras de imunoensaios de fluxo lateral são equipamentos utilizados para ler e classificar os resultados de testes rápidos como os de gravides, Covid, HIV etc. São encontradas em formas que vão de equipamentos complexos usados em laboratórios a leitoras compactas de fácil transporte e uso.

As leitoras de testes rápidos de fluxo lateral comercialmente disponíveis são capazes de realizar diagnósticos rápidos e precisos sendo de acordo com Park (2022), classificadas em 6 categorias dependendo da partícula marcadora e da tecnologia de medição utilizada: as colorimétricas, que medem o contraste ou a mudança de cor de partículas de ouro coloidal ou de látex monodisperso colorido; as de fluorescência, que excitam partículas de látex monodispersas fluorescentes usando fontes de luz, como LEDs, e medem sua intensidade de luminescência com um sensor óptico; as magnéticas, que medem a intensidade do campo magnético de partículas de látex monodispersas paramagnéticas ou de óxido de ferro usando sensores de campo magnético; as fototérmicas, que medem o calor das partículas por excitação de luz; as eletroquímicas, que medem as mudanças nas propriedades elétricas, como voltagem, corrente e impedância que aparecem nas partículas e as de sinal duplo que

usam dois ou mais simultaneamente dos princípios de medição mencionados anteriormente.

Na figura 4, temos um exemplo de uma leitora de testes rápidos de fluxo lateral capaz de fazer análises do tipo colorimétrica e fluorescência.

Figura 4 - Leitora de testes rápidos Axxin AX-2X-S

Fonte: Axxin (2023)

2.2 Visão computacional

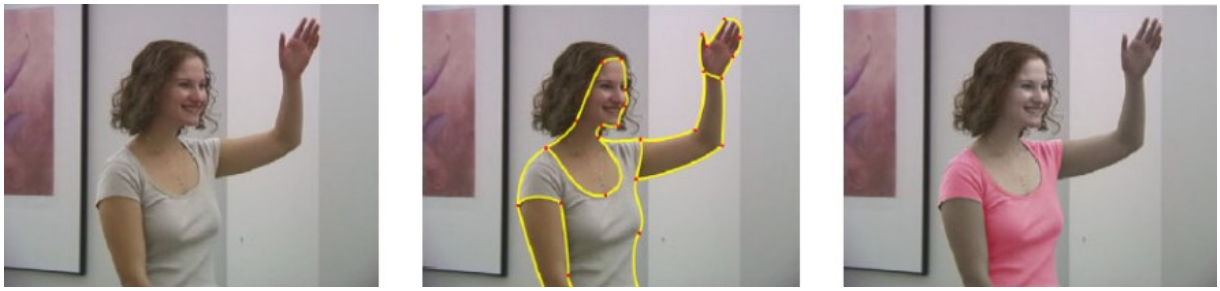
Visão computacional ou *Computer vision* é um campo ds inteligência artificial (IA) que permite que computadores e sistemas obtenham informações significativas a partir de imagens digitais, vídeos e outras entradas visuais. Com base nessas informações, é possível tomar ações ou fazer recomendações. Se a IA permite que os computadores pensem, o Computer Vision permite que eles vejam, observem e compreendam. (IBM, 2022).

A visão computacional é um campo da inteligência artificial que treina computadores para interpretar e compreender o mundo visual. Usando imagens digitais de câmeras e vídeos e modelos de aprendizado profundo, as máquinas podem identificar e classificar objetos com precisão – e então reagir ao que “veem”. (SAS, 2022).

Os avanços no campo da visão computacional geraram um grande aumento na capacidade de identificação e classificação de objetos e junto com a crescente

capacidade de processamento dos computadores, as aplicações para esse segmento ganharam destaque cada vez maior e presença crescente em diversas funções do cotidiano, estando já presente em carros autônomos, reconhecimento de escrita, captura de movimentos etc. Na figura 5, podemos ver um exemplo de visão computacional sendo utilizada para a detecção de bordas e recoloração de parte da imagem, técnicas comumente utilizadas para aplicações como segmentação de imagens, detecção e reconhecimento de padrões.

Figura 5 - Exemplo de aplicação de visão computacional



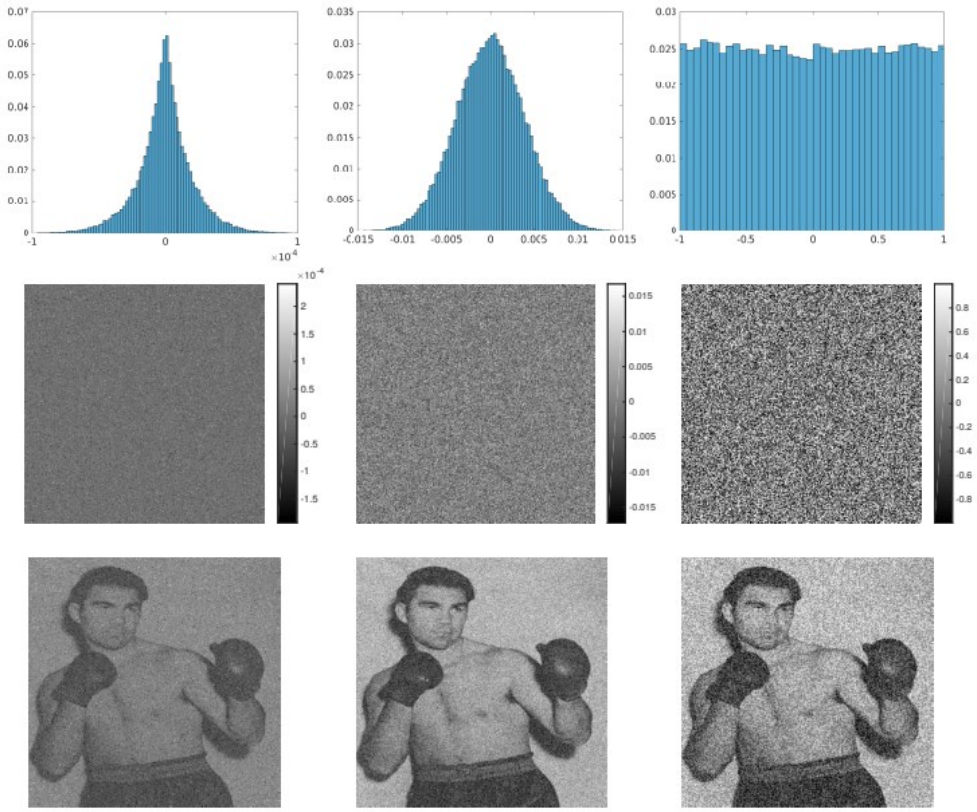
Fonte: Adaptado de Szeliski, (2011)

2.3 Processamento de imagens

2.3.1 Filtro Gaussiano

As imagens feitas por câmeras fotográficas são normalmente corrompidas por erros aleatórios chamados de ruído, que podem ocorrer em qualquer ponto da geração da imagem como a captura, transmissão ou processamento. Esse ruído é gerado de forma aleatória e pode ser descrito através de uma função densidade de probabilidade, sendo um ruído ideal chamado de ruído branco, no qual todas as frequências de ruído estão presentes e com mesma intensidade. Um caso especial de ruído branco é o ruído gaussiano, no qual a sua função densidade segue a distribuição Gaussiana. Na figura 6, podemos observar diferentes tipos de ruídos e seus efeitos em uma imagem.

Figura 6 - Efeito do ruído em uma imagem



Fonte: Franceschi et al, (2018)

O filtro gaussiano é um filtro capaz de filtrar altas frequências da imagem, causando também a perda de detalhes através da atenuação de bordas, gerando um efeito “borrado”. O efeito pode ser observado na figura seguir:

Figura 7 - Imagem original e com filtro gaussiano

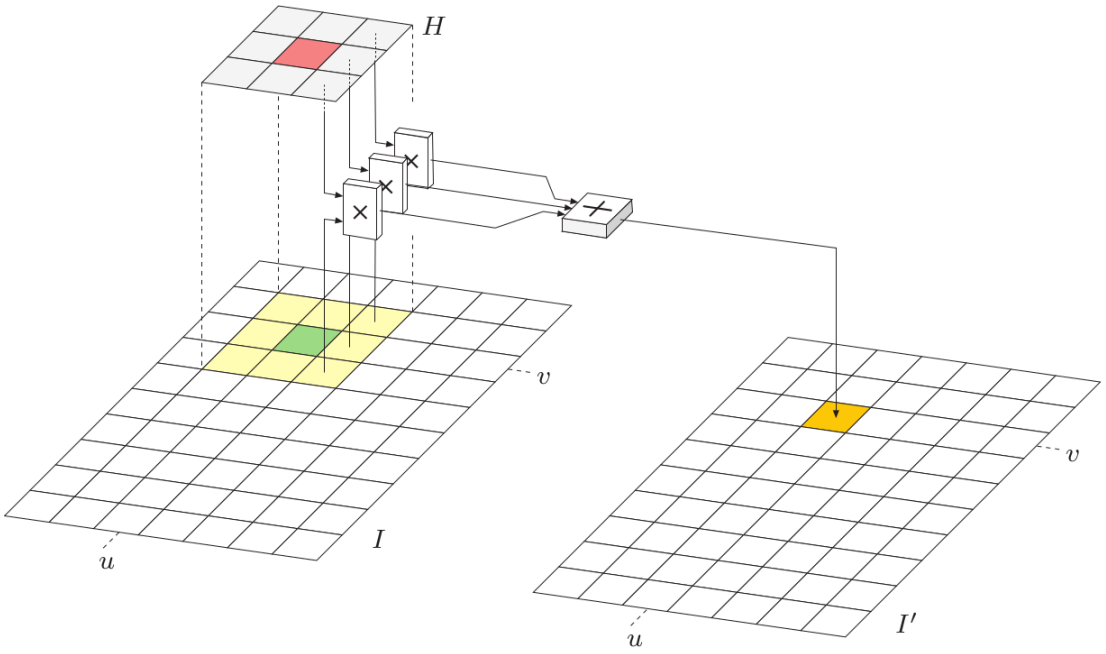
Fonte: Adaptado de Szeliski (2011)

2.3.2 Filtro 2D

A filtragem 2D ou filtragem espacial é um processo em cada pixel de uma imagem passa ter um valor que não mais depende apenas de si, mas também do valor dos pixels na sua vizinhança. O número de pixels utilizado no processo é definido através de uma matriz $m \times n$, chama de kernel ou máscara.

O processo consiste em mover o centro da máscara de filtro de ponto a ponto em uma imagem. Em cada ponto (x, y), a resposta do filtro nesse ponto é a soma dos produtos dos coeficientes do filtro e os pixels de vizinhança correspondentes na área abrangida pela máscara do filtro. [Gonzalez et al, 2009].

Figura 8 - Filtragem 2D



Fonte: Burger et al (2009)

2.3.3 Limiarização

Limiarização ou *thresholding*, é um tipo especial de quantização que separa os valores em duas classes, dependendo de um determinado valor de limiar que geralmente é constante. A função de limiarização mapeia todos os pixels para um de dois valores fixos. [Burger et a, 2009].

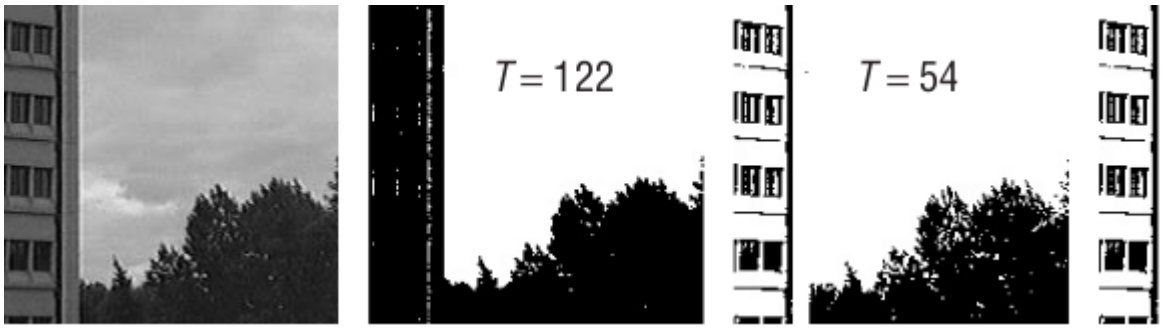
A limiarização é definida segundo a equação abaixo, e em caso dos valores de a e b forem 0 e 1 respectivamente, o processo é chamado de binarização.

$$g(i,j) = a, se\ g(i,j) > T,$$

$$= b, se\ g(i,j) \leq T$$

A limiarização com diferentes valores para T pode ser visualizada na figura 9, onde temos a imagem original e após limiarização com os valores de 122 e 54 para T.

Figura 9 - Exemplos de limiarização



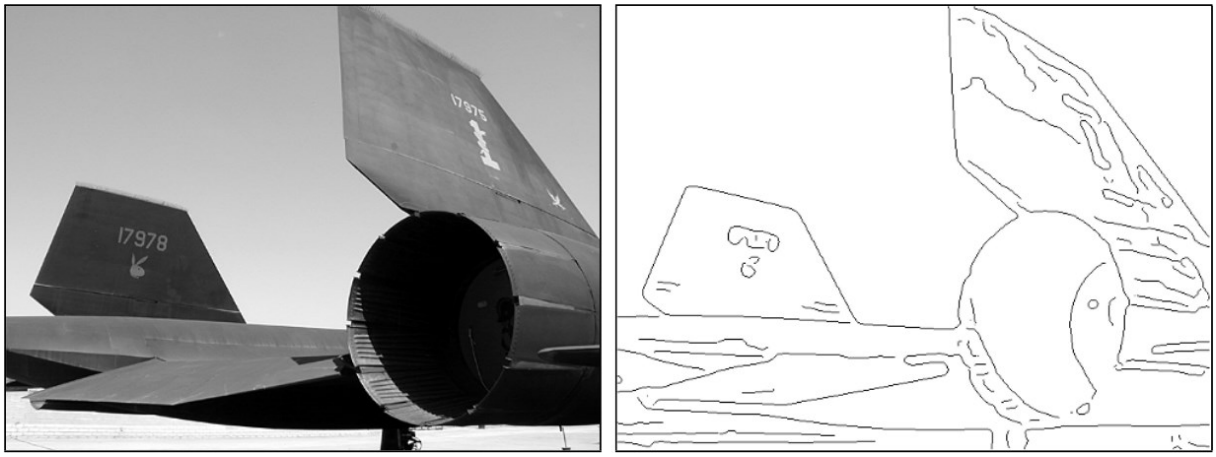
Fonte: Adaptado de Parker (2011)

2.3.4 Identificação de bordas e contornos

De acordo com [Burger et a, 2009]., bordas e contornos são de grande importância para a percepção visual e interpretação de imagens. A quantidade de informação percebida parece estar diretamente relacionada à distinção das estruturas e descontinuidades contidas, onde estruturas e contornos semelhantes a bordas parecem ser tão importantes para o sistema visual humano que algumas linhas em uma ilustração são muitas vezes suficientes para descrever inequivocamente um objeto ou uma cena.

Uma borda em uma imagem é a união de pontos onde a intensidade luminosa muda rapidamente em relação a sua vizinhança. Esses pontos são agrupados gerando segmentos de linha que podem formar linhas retas, curvas e contornos, como apresentado na figura 10**.**

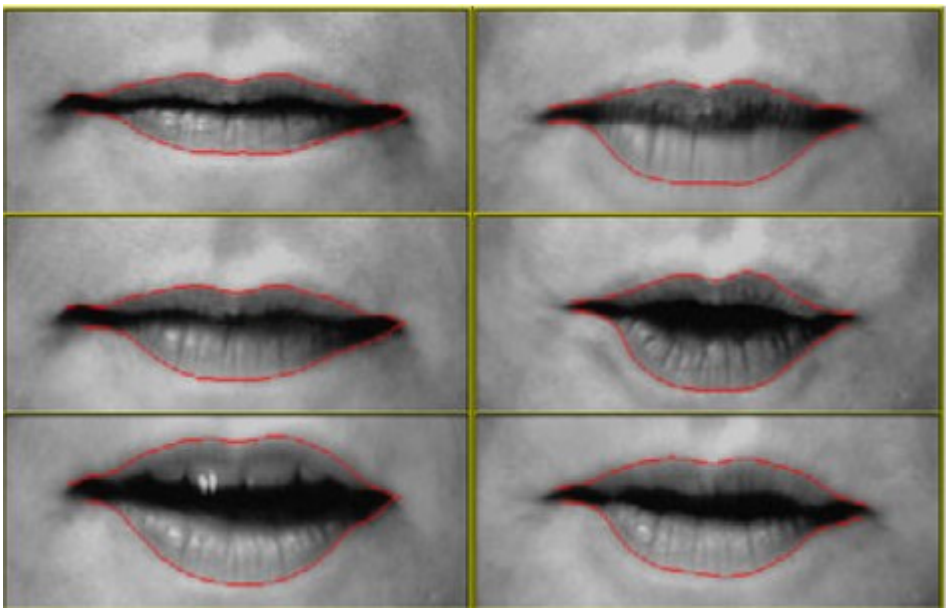
Figura 10 - Detecção de bordas



Fonte: Burger (2009)

Um contorno é traçado seguindo uma borda em ambas as direções até os traços se encontrarem, assim formando um caminho fechado. Na figura 11 podemos visualizar exemplos de detecção de contornos.

Figura 11 - Detecção de contornos



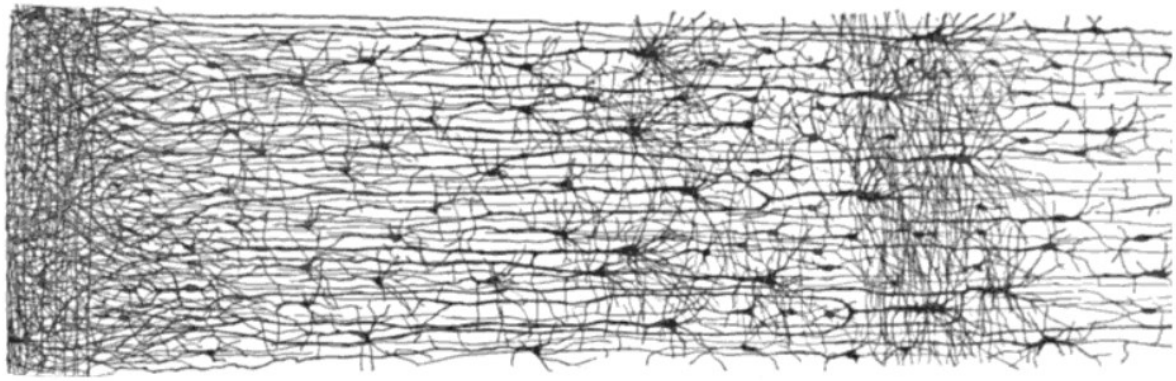
Fonte: Szeliski (2011)

2.4 Redes neurais

As redes neurais artificiais (ANNs) são modelos computacionais que surgiram através da inspiração nas redes neurais biológicas encontradas nos cérebros de seres humanos e de diversos outros animais. No cérebro humano, as redes neurais são formadas por células chamadas de neurônios, que fazem complexas cadeias de conexões entre si e com que transmitem impulsos nervosos.

Na figura 12 podemos visualizar uma representação de uma rede neural humana, ponto de partida para o entendimento de uma rede neural artificial.

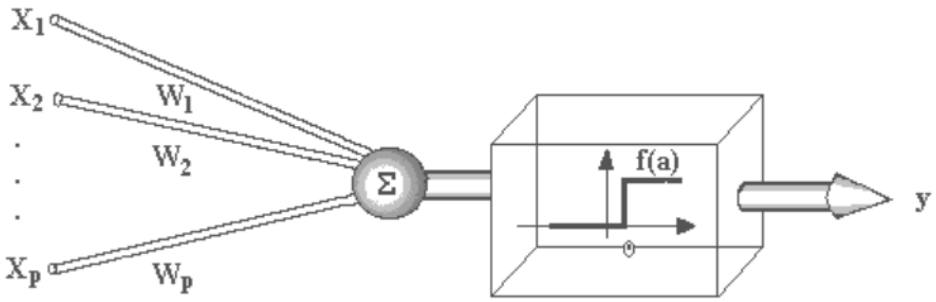
Figura 12 - Múltiplas camadas em uma rede neural biológica



Fonte: Gerón (2019)

As redes neurais artificiais possuem diversas unidades de processamento que fazem operações matemáticas básicas, chamadas de neurônios artificiais. Essas unidades recebem dados de entrada que são multiplicados por um peso w e então são somados, por fim passando por uma função de ativação $f(a)$ e gerando uma saída y.

Figura 13 - Esquema de unidade McCulloch-Pitts

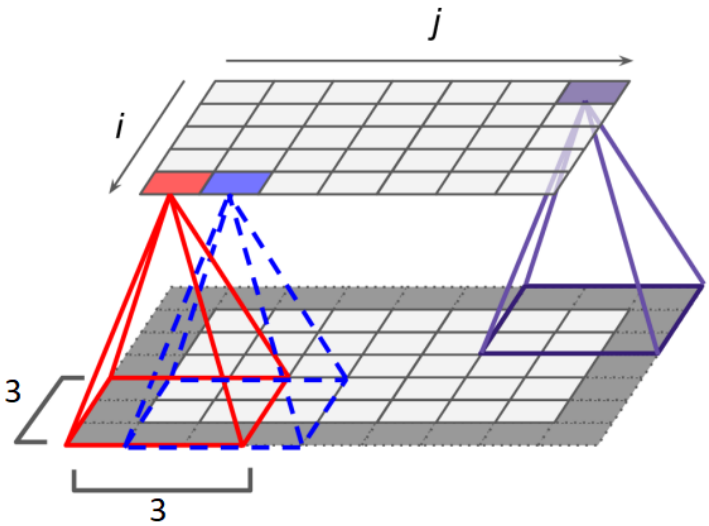


Fonte: Carvalho (2022)

2.4.1 Redes neurais convolucionais

As redes neurais convolucionais (CNNs) se diferenciam de outros tipos de redes neurais artificiais por usarem operações de convolução na extração de características dos dados de entrada, normalmente uma imagem. Essas operações são feitas através de uma matriz $m \times n$, chamada *kernel*, que percorre toda a imagem aplicando sobre diferentes segmentos uma operação linear de convolução, gerando uma saída que mantém ou enfatiza as características de desejo e descarta ou suprime características que são sejam importantes para aplicação da CNN. Segundo Ajit (2020), a camada de convolução é a camada mais básica, mas ao mesmo tempo mais importante na CNN. Basicamente, ela convoluciona a matriz de pixels gerada para a imagem ou objeto fornecido para produzir um mapa de ativação para a imagem fornecida, armazenando todas as características distintivas de uma imagem dada, ao mesmo tempo em que reduz a quantidade de dados a ser processada.

Figura 14 - Convolução com um kernel 3x3



Fonte: Adaptado de Gerón (2019)

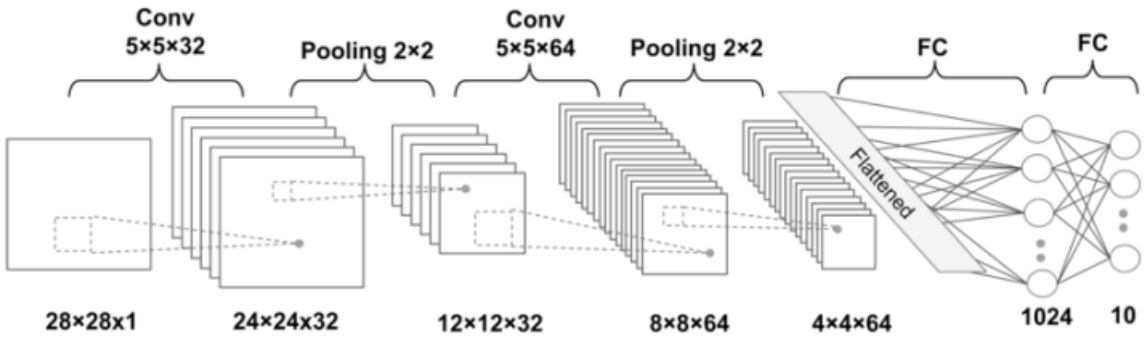
Uma CNN é composta por camadas de convolução seguidas por uma camada de *pooling*, onde as camadas de *pooling* servem para reduzir as dimensões dos dados recebidos da camada anterior. Ajit (2020), diz que as camadas de pooling “reduzem as dimensões, mantendo apenas as características importantes. Isso, por sua vez, reduz o número de características aprendíveis para o modelo”. Ou ainda, de acordo com Li (2021) , a camada de pooling “utiliza o princípio da correlação local da imagem para reduzir a amostragem de uma imagem, o que pode reduzir a quantidade de

dados enquanto retém informações úteis". Sendo a correlação local de uma imagem, o princípio de que pixels próximos tem informações semelhantes.

Esse arranjo de camadas de convolução seguidas por camadas de *pooling* pode ser repetido diversas vezes em uma CNN. Essas camadas são normalmente seguidas por uma ou mais camadas *Fully Connected* (FC), que são as camadas que conectam os nós de uma camada a camada seguinte, como diz Albawi (2017), "cada nó em uma camada *fully connected* está diretamente conectado a todos os nós tanto na camada anterior quanto na próxima camada".

Na figura 15, temos a arquitetura de uma CNN exemplificando o posicionamento e ligações entre as camadas anteriormente descritas.

Figura 15 - Arquitetura de uma CNN



Fonte: Raschka et al (2017)

O treinamento de uma rede neural é feito através de diversas iterações sobre uma base de dados, ao longo das quais os pesos das conexões entre os neurônios e o viés da rede é ajustado. Esse ajuste é feito para minimizar a diferença entre os resultados obtidos na saída da rede neural e os resultados esperados.

Nas redes neurais utilizadas para esse trabalho, o algoritmo utilizado no treinamento foi a retropropagação, onde os ajustes são feitos propagando o erro na saída para as camadas anteriores. Para realizar o ajuste, é calculada a função de custo (ou função de perda) da rede neural artificial, que é utilizada para avaliar o erro no treinamento comparando a classificação dada pela rede ao rótulo dos dados. Existem diferentes tipos de função de custo, sendo a utilizada para esse trabalho a entropia cruzada. O erro calculado é repassado às camadas anteriores, onde é usado para corrigir os pesos das conexões entre os neurônios através de algoritmos de otimização para encontrar valores mínimos para a função de custo. Esse processo é

feito a cada iteração sobre a base de dados, convergindo os resultados obtidos na saída com os resultados esperados.

2.4.2 Medidas de desempenho de redes neurais

Existem diversas métricas para medir o desempenho de uma rede neural artificial, e de acordo com Twomey (1995), a validação é um aspecto crítico de qualquer construção de modelo. Embora não exista uma metodologia bem formulada ou teórica para a validação de modelos de redes neurais artificiais, a prática usual é basear a validação em alguma medida de desempenho de rede especificada por dados que não foram usados na construção do modelo.

Algumas das métricas comumente utilizadas para medir o desempenho e realizar a validação de uma rede neural artificial, e que serão utilizadas neste trabalho, são a acurácia, precisão e matriz confusão.

A acurácia é utilizada para indicar a porcentagem de classificações corretas feita por uma rede neural, sendo definida pela proporção de amostras classificadas corretamente sobre o total de amostras classificadas. Podemos visualizar a acurácia conforme a seguinte equação:

$$acurácia = \frac{TP + TN}{TP + TN + FP + FN}$$

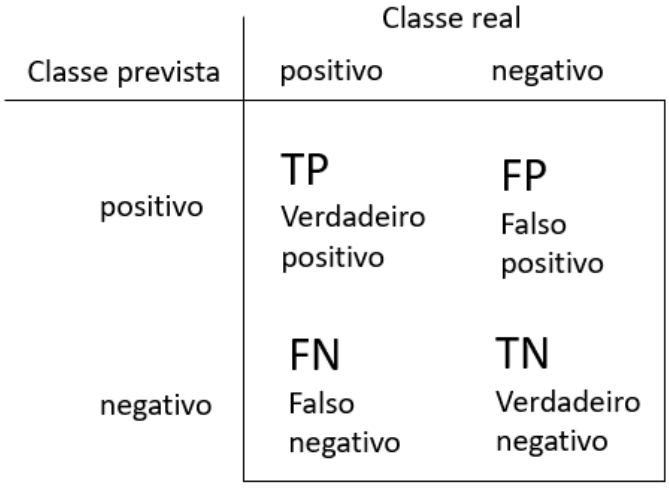
Sendo TP as amostras positivas corretamente classificadas como positivas, TN as amostras negativas corretamente classificadas como negativas, FP as amostras negativas erroneamente classificadas como positivas e FN as amostras positivas erroneamente classificadas como negativas.

A precisão é uma métrica definida pelo número de amostras positivas corretamente classificadas como positivas, muito importante para casos em que o número de falso positivos deve ser minimizado, como na detecção de doenças. A precisão pode ser definida conforme a equação:

$$precisão = \frac{TP}{TP + TN + FP + FN}$$

Uma matriz confusão pode ser utilizada para fácil visualização do desempenho de uma rede neural artificial, e pode ser utilizada para o cálculo de outras métricas como a acurácia e a precisão. Na imagem a seguir podemos ver a estrutura de uma matriz confusão para uma classificação binárias de dados.

Figura 16 - Matriz confusão



Fonte: Adaptado de Lovell (2022)

2.5 Tecnologias utilizadas

2.5.1 OpenCV (Computer Vision)

OpenCV (*Open Source Computer Vision Library*) é um *framework* de código aberto para criação de aplicações em visão computacional e aprendizado de máquina. Lançado em 2000 e desde então mantido pela Intel Corporation, o *framework* tem suporte para os diversos sistemas operacionais como Windows, Mac OS e GNU/Linux.

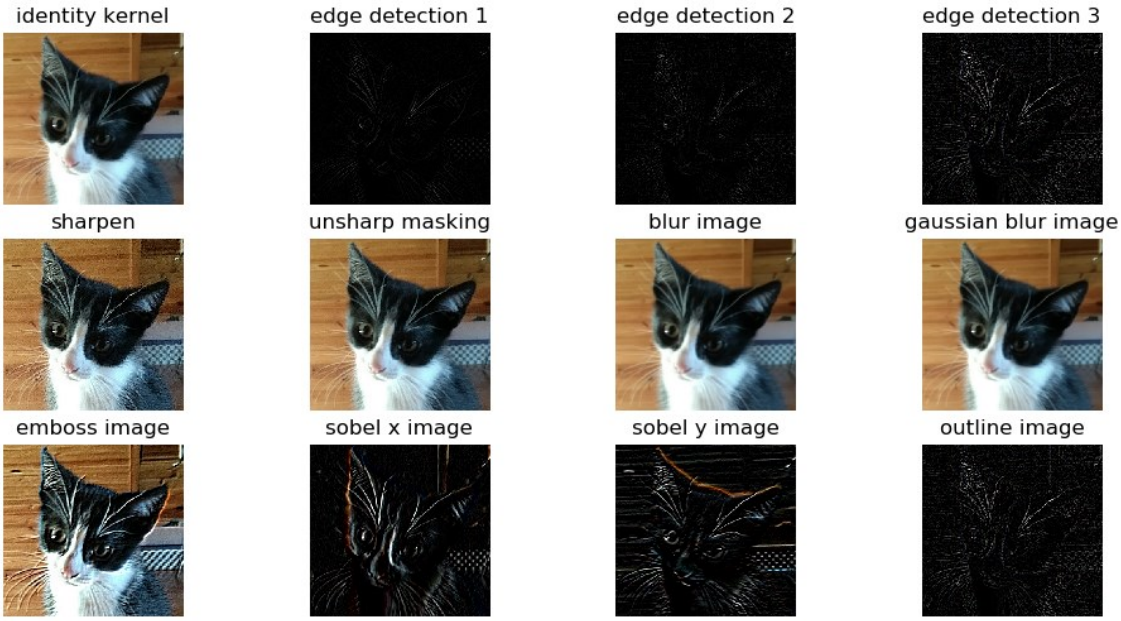
De acordo com Villán (2019), a biblioteca ganhou popularidade entre cientistas e acadêmicos porque muitos algoritmos de visão computacional de última geração são fornecidos por essa biblioteca. Além disso, é frequentemente usado como uma ferramenta de ensino para visão computacional e aprendizado de máquina. Deve-se levar em consideração que o OpenCV é robusto o suficiente para suportar aplicativos do mundo real. É por isso que o OpenCV pode ser usado para produtos não comerciais e comerciais. Por exemplo, é usado por empresas como Google, Microsoft, Intel, IBM, Sony e Honda. Institutos de pesquisa nas principais universidades, como MIT, CMU ou Stanford, fornecem suporte para a biblioteca. O OpenCV foi adotado em

todo o mundo. Tem mais de 14 milhões de downloads e mais de 47.000 pessoas em sua comunidade

O OpenCV é amplamente utilizado em um vasto conjunto de aplicações em visão computacional e processamento de imagens como reconhecimento facial, rastreamento de movimentos, identificação de objetos e veículos autônomos, tendo notoriamente sido utilizado no veículo Stanley, desenvolvido pela universidade de Stanford e que venceu a competição de carros autônomos *DARPA Grand Challenge* em 2005 na Califórnia.

Na figura 17, podemos observar exemplos do openCV sendo utilizado para aplicação de filtros e detecção de bordas.

Figura 17 - Exemplo de aplicação do OpenCV



Fonte: Adaptado de Villán (2019)

2.5.2 TensorFlow

TensorFlow é uma plataforma *open source* para o desenvolvimento de modelos para *machine learning*. Desenvolvido pelo Google LLC em 2015, o *framework* suporta diversos sistemas operacionais como Windows, Mac OS e GNU/Linux e suporta diversas linguagens de programação como Python e C++. É uma plataforma que oferece solução *end-to-end*, amplamente utilizada em projetos de *machine learning* de larga escala, “foi desenvolvido pelo Google Brain Team e potencializa muitos dos

serviços de grande escala do Google, como o Google Cloud Speech, Google Photos e Google Search". (GÉRON, 2019, p. 368).

Sendo utilizado para as mais diversas aplicações de *machine learning*, como classificação de imagens, identificação de escrita, identificação facial entre outros, o TensorFlow destaca-se como "a mais popular biblioteca de *deep learning* em termos de citações em artigos, adoção em empresas, estrelas no GitHub, etc." (GÉRON, 2019, p. 368).

O TensorFlow é amplamente utilizado no desenvolvimento de aplicações em *deep learning,* sendo comumente combinado com o Keras, uma sub biblioteca do TensorFlow que permite maior grau de abstração tornando a ferramenta bastante amigável ao usuário e facilitando o processo de criar, treinar, avaliar e executar redes neurais artificiais. O uso conjunto dessas ferramentas torna muito simples manipular grandes conjuntos de dados, podendo-se gerar facilmente a partir do *dataset* original os subconjuntos de treinamento e validação, além de possibilitar modelar redes neurais, definindo parâmetros como números de camadas e função de ativação de forma bastante simplificada.

O TensorFlow também permite exportar os modelos treinados, sendo possível utilizar esse modelo em uma nova aplicação sem a necessidade de realizar todo o processo novamente. Isso permite criar aplicações em que o as etapas que exigem maior poder de processamento computacional como a manipulação dos *datasets* e treinamento sejam feitos por computadores de maior capacidade, e que esses modelos gerados possam ser usados em dispositivos muito mais modestos como celulares e microcontroladores, onde apenas a inferência será realizada.

2.5.2.1 TensorFlow Lite

O TensorFlow Lite é uma versão da biblioteca TensorFlow otimizada para execução em dispositivos com menor capacidade de processamento, como dispositivos móveis e sistemas embarcados. Um modelo do TF Lite é um arquivo de extensão ".tflite", que se utiliza da biblioteca de serialização Flatbuffers, que foi desenvolvida pelo Google LLC para aplicação em jogos. "Isso oferece diversas vantagens sobre o formato do modelo buffer de protocolo do TensorFlow, como um tamanho reduzido e uma inferência mais rápida, o que permite ao TensorFlow Lite

uma execução eficiente nos dispositivos com recursos limitados de computação e memória” [Documentação TensorFlow, 2023].

Um modelo TF Lite pode ser criado a partir de um modelo padrão do TensorFlow, possibilitando o treinamento do modelo em computadores de maior performance e exportando o modelo pré-treinado para os dispositivos com limitação de recursos, onde poderá ser usado para classificação de novos dados.

2.5.3 Tkinter

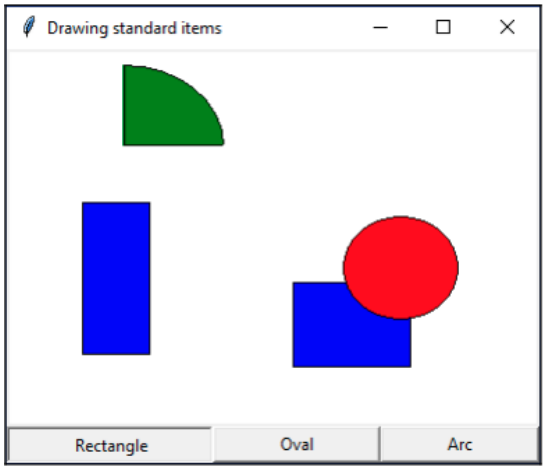
Tkinter é um *framework cross-platform* para o desenvolvimento de GUI’s em Python, sendo incluído na instalação padrão do Python e a interface mais utilizada para criar GUI com Tcl/Tk *toolkit* na linguagem.

O Tkinter aparece em aplicações gráficas e pode ser utilizado para exibir textos e imagens, criar entradas de valores, exibir itens em formato de listas, tratamento de eventos de mouse etc. O desenvolvimento de uma GUI com o Tkinter é feito utilizando orientação a objetos, com classes representando as diferentes telas da aplicação, sendo os atributos da classe os seus respectivos componentes como botões e textos, e os métodos classe o que permite a interatividade, como chamar um *script* ao clicar em botão que modifique a tela atual ou que leve o usuário a uma tela diferente.

O *framework* oferece muitas vantagens, como cita o autor “Tkinter também é muito rápido e fácil de aprender. O código pode ser escrito tanto procedimentalmente quanto usando práticas orientadas a objetos e funciona perfeitamente em qualquer sistema operacional que suporte Python, incluindo Windows, MacOS e Linux.” (Love, 2018).

Na figura 18 podemos visualizar um exemplo de aplicação criada com o Tkinter, onde objetos podem ser posicionados na tela e cliques de mouse podem iniciar eventos na interface.

Figura 18 - Aplicação gráfica utilizando o Tkinter



Fonte: PAZ (2018)

3 DESENVOLVIMENTO

Neste capítulo serão exibidos os materiais e as técnicas utilizados para desenvolvimento de um protótipo de uma leitora para testes rápidos de fluxo lateral. Serão enumerados os passos para a criação da rede neural artificial por trás da classificação dos testes e da criação da estrutura do protótipo.

3.1 Base de dados

As imagens utilizadas para o treinamento da rede neural foram concedidas pelo Laboratório de Ciências e Tecnologias Aplicadas em Saúde (LaCTAS) do Instituto Carlos Chagas - Fiocruz Paraná. Foram cedidas 2697 imagens de cassetes já rotuladas conforme a doença a ser examinada e o resultado apontado. Na figura 19 podemos visualizar exemplos de imagens que fazem parte desse conjunto de imagens.

Figura 19 - Exemplos de imagens pertencentes a base de dados.

Fonte: Autoria própria (2023)

A base de dados é composta por imagens de diferentes cassetes de fluxo lateral, usados para os testes de Covid-19, HIV e Hepatites A e B. Essas imagens foram inicialmente separadas em categorias referentes aos seus respectivos testes. Na tabela 1, estão representados o número de imagens pertencentes a cada categoria.

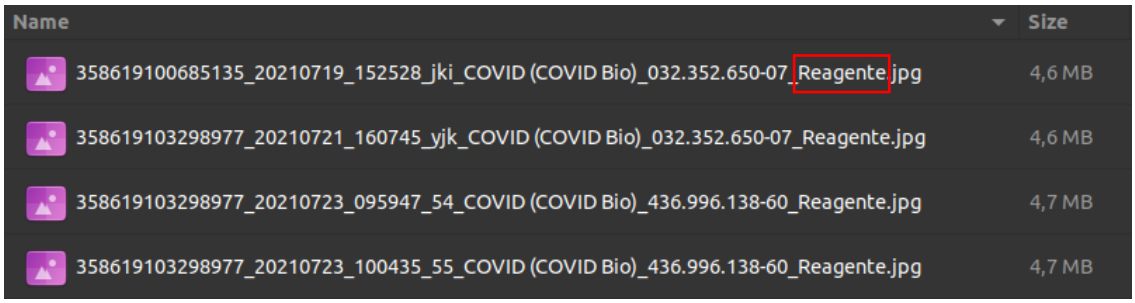
Tabela 1 - Base de dados para cada tipo de cassete

Cassete	Total de imagens
Covid-19	830
HIV	732
Hepatites A e B	686

Fonte: Autoria própria (2023)

As imagens da base de dados foram previamente rotuladas conforme o resultado apontado pelo exame, esse resultado poderia ser “Reagente”, “Não Reagente” ou “Inválido”. Essa rotulação foi feita através do nome do arquivo. Na figura 20, uma captura de tela é utilizada para mostrar como as imagens foram rotuladas. Pode-se observar que o resultado do exame está presente no nome de cada arquivo.

Figura 20 - Exemplo de arquivos rotulados da base de dados



Fonte: Autoria própria (2023)

3.2 Processamento de imagens

Toda a base de dados passou por uma fase de pré-processamento, com a intenção de padronizar as dimensões das imagens e aplicação de filtros para a remoção de ruídos. Com as imagens preparadas, deu-se início à etapa de segmentação, onde o objetivo era extrair as características relevantes presentes nas imagens e remover componentes que pudessem causar distorções nos resultados.

As imagens passaram inicialmente por uma função de *thresholding*, onde o valor de cada pixel que estivesse abaixo de um limiar foi definido como zero, e os que estivessem acima foram definidos como o valor máximo, gerando assim uma imagem em tons de cinza. Na figura 21 podemos ver uma imagem antes e depois de passar por esse processo.

Figura 21 - Imagem original e após passar pela função de thresholding

Fonte: Autoria própria (2023)

Com as imagens obtidas na etapa anterior, foram traçadas as bordas do cassete e de outros elementos que se destacassem em relação ao seu entorno. Na figura 22 podemos ver esse processo de detecção de bordas.

Figura 22 - Detecção das bordas do cassete

Fonte: Autoria própria (2023)

Com os elementos delimitados, foram obtidas as coordenadas de cada objeto e então calculada a área de seu contorno. As áreas de cada objeto foram comparadas com área esperada para o contorno do cassete e a que mais se aproximava foi definida como o provável contorno de interesse.

As coordenadas do contorno obtido para o cassete foram utilizadas para recortar a imagem deixando apenas a região onde o teste se encontra, removendo

assim todo o contorno visto nas imagens anteriores e que poderiam carregar características que seriam processadas pela rede neural e levar a resultados inesperados. Na figura 23 temos um exemplo de imagem obtidas após o processo de recortar o cassete de uma imagem.

Figura 23 - Imagem gerada a partir da detecção de bordas

Fonte: Autoria própria (2023)

As imagens geradas correspondentes apenas à região do cassete possuem marcações realizadas pelos profissionais da saúde no momento de sua utilização. Essas marcações seriam processadas na CNN e poderiam impactar de maneira negativa os resultados. Para solucionar esse problema, a base passou por mais uma etapa de segmentação, com a intenção de gerar uma nova imagem correspondente apenas à região onde as bandas estão presentes. Na figura 24 pode ser visualizada uma imagem gerada através dessa segmentação, onde apenas a banda de controle e teste foram mantidas.

Figura 24 - Imagem gerada após identificar a região das bandas

Fonte: Autoria própria (2023)

Com este processo, o objetivo foi obter as coordenadas na imagem onde as bandas estariam representadas, e assim possibilitar aplicar essas coordenadas no arquivo original para recortar apenas o fragmento de interesse. Optou-se por continuar o processamento da imagem a partir do recorte na imagem original e não com o *output* gerado na etapa anteriormente apresentada, porque os filtros aplicados para obter os contornos em etapas anteriores não se mostraram os de melhor desempenho para visualização das bandas, voltando assim para a imagem original sem recortes ou

filtros e então recortando nela as listras de teste com base nas coordenadas obtidas nos processos mencionados anteriormente.

As coordenadas anteriormente encontradas e aplicadas no arquivo original geraram como imagens como a apresentada na figura 25.

Figura 25 - Bandas na imagem original

Fonte: Autoria própria (2023)

Foi então realizada uma última etapa de processamento. As imagens obtidas no passo anterior passaram por um filtro 2D de *kernel* 5x5 e novamente por uma função de *thresholding* adaptativo para gerar uma nova imagem onde as bandas do cassete pudessem ser mais bem visualizadas, possibilitando um melhor resultado quando utilizadas para o treinamento de uma CNN. Na figura 26, temos o resultado obtido após aplicar o filtro 2D e função de *thresholding* na imagem apresentada na figura 25.

Figura 26 - Imagem gerada ao fim do algoritmo de processamento

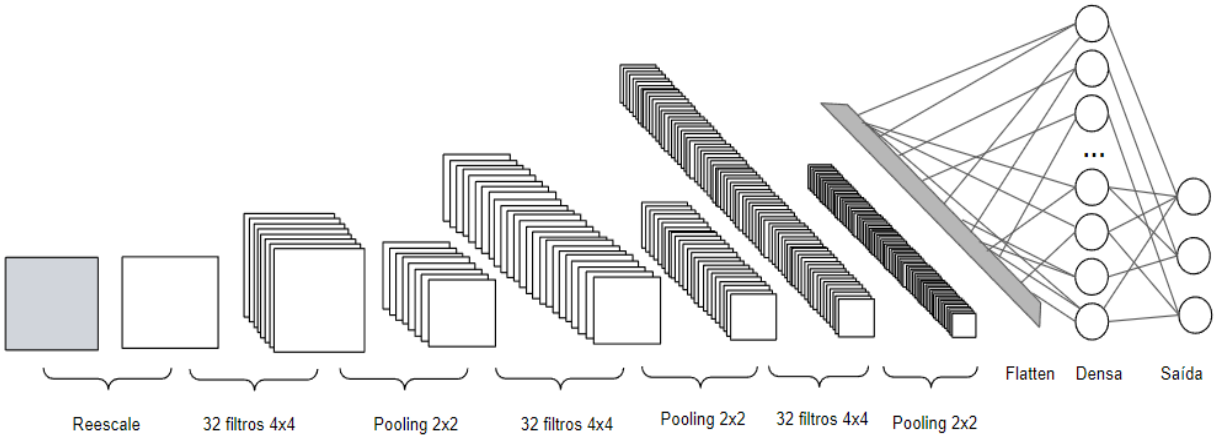
Fonte: Autoria própria (2023)

3.3 Rede neural

Para que a leitora fosse capaz de identificar qual o tipo de teste foi inserido para análise e classificar os cassetes segundo o seu resultado, foram utilizadas diferentes redes neurais para atuar em cada situação. Para identificar o tipo de cassete, foi criada uma rede neural utilizando as imagens dos diferentes tipos de teste disponíveis para esse trabalho, com o resultado advindo de sua classificação sendo utilizado para selecionar uma rede neural específica para obter o resultado do teste para cada um dos diferentes cassetes. Para isso foram criadas outras três redes neurais, para avaliar cada cassete com as suas respectivas características de forma individualizada. Essas redes neurais avaliam as fotos tiradas das bandas de teste e controle do cassete e tem como saída o resultado “reagente” ou “não reagente”.

Todas as redes neurais desenvolvidas para esse trabalho utilizaram uma mesma topologia para a classificação de imagens em uma rede neural convolucional. As camadas utilizadas na respectiva ordem podem ser visualizadas na figura 27.

Figura 27 - Arquitetura da CNN desenvolvida



Fonte: Autoria própria (2023)

- A camada Reescale foi utilizada para normalizar os valores dos pixels na imagem para um intervalo [0,1].
- A camada de convolução 2D foi aplicada por três vezes na rede neural, onde em cada uma delas foram aplicados 32 filtros de tamanho 4x4.
- A camada de Pooling foi aplicada por três vezes, após cada camada de convolução. Nessa camada é utilizado *max pooling* para efetuar *down-sampling* das características recém extraídas na camada anterior.

- A camada Flatten converte a saída da camada anterior em um vetor unidimensional.
- A camada Densa é uma camada com 128 neurônios.
- A camada Saída possui número de neurônios igual ao número de classes utilizadas para o treinamento da rede, sendo neste trabalho 2 ou 3.

3.3.1 Identificação de testes

A rede neural para reconhecer o modelo de teste rápido de fluxo lateral inserido na leitora foi criada para classificar os cassetes segundo suas respectivas doenças de aferição (Covid-19, HIV e hepatite). Nessa rede neural, o resultado não foi levado em consideração, sendo as imagens agrupadas somente segundo o modelo do teste. Os conjuntos de imagens referentes a cada doença foram colocados em diferentes pastas e o caminho para elas utilizado como parâmetro para o TensoFlow para dar início ao treinamento da rede neural. As imagens foram pré-processadas, padronizando as imagens para terem tamanho de 450x150 pixels. Por fim, foi definido o *batch size* para a rede neural, sendo esse o parâmetro que define o número de imagens que serão tratadas por vez no treinamento da rede neural. Essa etapa pode ser visualizada no trecho de código a seguir.

Figura 28 - Definindo classes para cassetes

```
data_dir = pathlib.Path(data_dir)
image_count = len(list(data_dir.glob('*.jpg')))
COVID = list(data_dir.glob('COVID*'))
len_COVID=len(list(data_dir.glob('COVID*')))
HIV = list(data_dir.glob('HIV*'))
len_HIV=len(list(data_dir.glob('HIV*')))
HEPATITE = list(data_dir.glob('HEPATITE*'))
len_HEPATITE=len(list(data_dir.glob('HEPATITE*')))
batch_size = 5
img_height = 150
img_width = 450
```

Fonte: Autoria própria (2023)

As imagens utilizadas foram separadas em dois grupos, um com setenta por cento do total de imagens e outro com os trinta por cento restantes. O conjunto maior

foi utilizado para o treinamento da rede neural e o menor para a validação. A separação foi feita de forma aleatória, conforme trecho de código a seguir.

Figura 29 - Conjuntos de treinamento e validação (cassetes)

```
train_ds = tf.keras.utils.image_dataset_from_directory(
        data_dir,
        validation_split=0.3,
        subset="training",
        seed=123,
        image_size=(img_height, img_width),
        batch_size=batch_size
)
val_ds = tf.keras.utils.image_dataset_from_directory(
        data_dir,
        validation_split=0.30,
        subset="validation",
        seed=123,
        image_size=(img_height, img_width),
        batch_size=batch_size
)
```

Fonte: Autoria própria (2023)

Para a criação da rede neural, foi necessário definir o número e o tipo das camadas utilizadas. Foram criadas dez camadas sequenciais, sendo a primeira camada utilizada apenas para definir a escala para os valores de entrada e então três sequências de uma camada de convolução e uma de *pooling*. Em seguida, temos uma camada que modela os dados em um vetor (*flatten)* e então duas camadas *Dense* para gerar a saída. A criação das camadas mencionadas pode ser visualizada no trecho de código da figura 30.

Figura 30 - Camadas da CNN (cassetes)

```
num_classes = 3
model = tf.keras.Sequential([
        tf.keras.layers.Rescaling(1./255),
        tf.keras.layers.Conv2D(32, 4, activation='relu'),
        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Conv2D(32, 4, activation='relu'),
        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Conv2D(32, 4, activation='relu'),
        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(num_classes)
])
```

Fonte: Autoria própria (2023)

Para compilar a rede neural, foram necessários alguns parâmetros, o otimizador, a função de perda e as métricas para análise, conforme o trecho de código da figura 31.

Figura 31 - Trecho de código para compilar CNN (cassetes)

```
model.compile(
 optimizer='adam',
 loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),
 metrics=['accuracy'])
```

Fonte: Autoria própria (2023)

Para iniciar o treinamento, são passados como parâmetros os conjuntos de imagem de treino e validação e o número de épocas (número de iterações sobre todo o conjunto de imagens). Essa etapa pode ser visualizada no trecho de código da figura 32.

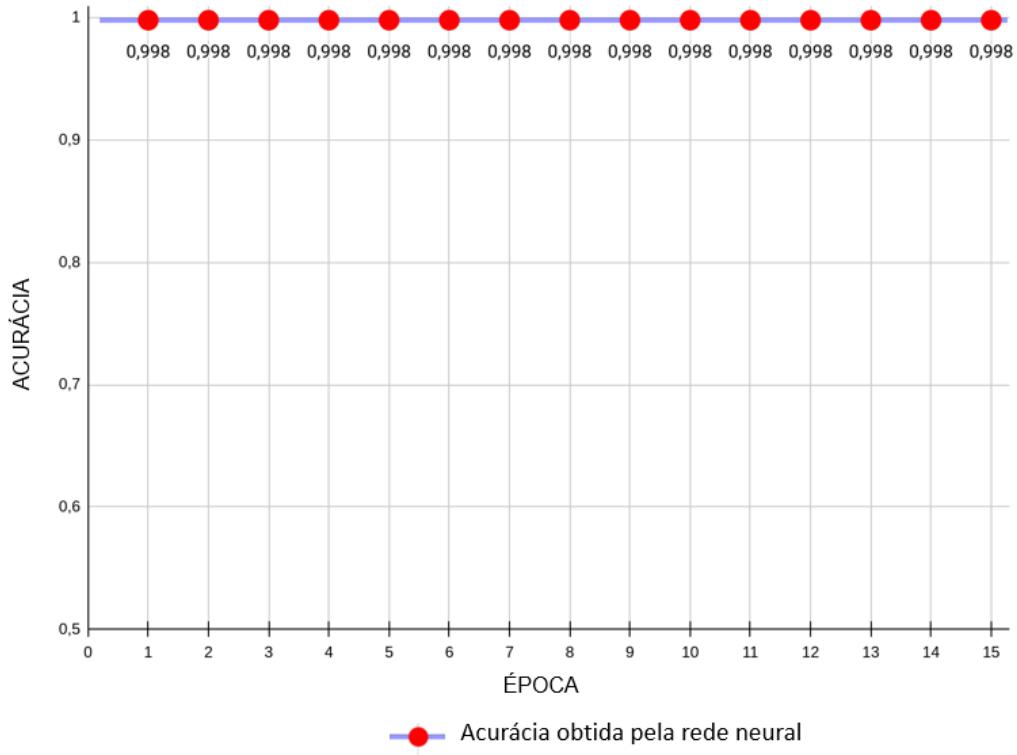
Figura 32 - Treinamento CNN (cassetes)

```
history = model.fit(
        train_ds,
        validation_data=val_ds,
        epochs=15,
        shuffle=True,
        verbose=2
)
```

Fonte: Autoria própria (2023)

Para o treinamento dessa rede neural, foram utilizadas 1846 imagens, das quais 553 foram utilizadas para validação. No gráfico 1, podemos visualizar a acurácia obtida pela rede neural durante a validação, essa sendo a porcentagem de imagens corretamente classificadas pela rede neural dentro do conjunto de 553 imagens para validação. A CNN foi treinada em 15 épocas, sendo esse o número de vezes que a rede neural passou por todo o conjunto de dados. O resultado obtido pode ser visualizado abaixo, onde podemos visualizar que a rede neural classificou 552 das 553 imagens de forma correta em cada uma das 15 épocas de treinamento.

Gráfico 1 - Acurácia por época (cassetes)



Fonte: Autoria própria (2023)

Os valores de saída gerados pelo TensorFlow durante o treinamento foram capturados e estão exibidos na figura 33.

Figura 33 - Outputs no terminal (cassetes)

```
['COVID', 'HEPATITE', 'HIV']
Found 1846 files belonging to 3 classes.
Using 553 files for validation.
['COVID', 'HEPATITE', 'HIV']
Epoch 1/15
259/259 - 55s - loss: 0.1741 - accuracy: 0.9358 - val_loss: 0.0088 - val_accuracy: 0.9982 - 55
Epoch 2/15
259/259 - 56s - loss: 8.2258e-04 - accuracy: 0.9992 - val_loss: 0.0023 - val_accuracy: 0.9982
Epoch 3/15
259/259 - 57s - loss: 1.4628e-06 - accuracy: 1.0000 - val_loss: 0.0027 - val_accuracy: 0.9982
Epoch 4/15
259/259 - 56s - loss: 5.5916e-07 - accuracy: 1.0000 - val_loss: 0.0030 - val_accuracy: 0.9982
Epoch 5/15
259/259 - 57s - loss: 3.0941e-07 - accuracy: 1.0000 - val_loss: 0.0033 - val_accuracy: 0.9982
Epoch 6/15
259/259 - 57s - loss: 1.9573e-07 - accuracy: 1.0000 - val_loss: 0.0034 - val_accuracy: 0.9982
Epoch 7/15
259/259 - 56s - loss: 1.3396e-07 - accuracy: 1.0000 - val_loss: 0.0036 - val_accuracy: 0.9982
Epoch 8/15
259/259 - 56s - loss: 9.6805e-08 - accuracy: 1.0000 - val_loss: 0.0037 - val_accuracy: 0.9982
Epoch 9/15
259/259 - 56s - loss: 7.2742e-08 - accuracy: 1.0000 - val_loss: 0.0038 - val_accuracy: 0.9982
Epoch 10/15
259/259 - 57s - loss: 5.5963e-08 - accuracy: 1.0000 - val_loss: 0.0039 - val_accuracy: 0.9982
```

Fonte: Autoria própria (2023)

Por fim, o modelo criado foi salvo e convertido para o TensorFlow Lite, estando assim pronto para ser utilizado pela leitora. O trecho de código da figura 34 mostra como essa etapa foi realizada.

Figura 34 - Conversão para TFLite (cassetes)

```
model.save(MODEL_DIR)
converter = tf.lite.TFLiteConverter.from_saved_model(MODEL_DIR)
converter.target_spec.supported_ops = [
        tf.lite.OpsSet.TFLITE_BUILTINS,
        tf.lite.OpsSet.SELECT_TF_OPS
]
tflite_model = converter.convert()
```

Fonte: Autoria própria (2023)

3.3.2 Classificação dos resultados

Para classificar resultados, foram utilizadas redes neurais convolucionais diferentes para cada tipo de teste rápido de fluxo lateral. Cada rede neural foi treinada utilizando as imagens de um único tipo de teste. Mesmo gerando 3 redes neurais para a classificação dos resultados dos testes para Covid-19, HIV e hepatite, a única diferença entre essas CNNs foram as imagens utilizadas para seu treinamento e validação, mantendo uma única arquitetura para as redes.

Para o treinamento de cada rede neural, os conjuntos de imagens de cada tipo de teste foi dividido entre “REAGENTE” e “NAO_REAGENTE”, e então utilizados pelo TensorFlow para a criação de cada CNN. As imagens passadas foram previamente pré-processadas para que todas tivessem o mesmo tamanho, sendo definido o tamanho padrão de 65 pixels de altura por 375 pixels de largura, definidos como “img_height” e “img_width” no trecho de código da figura 35. Foi então escolhido o *batch size* para a rede neural, sendo esse o parâmetro que define o número de imagens que serão tratadas por vez durante o processo de treinamento.

Figura 35 - Trecho de código definindo classes para a rede neural

```
data_dir = pathlib.Path(data_dir)
image_count = len(list(data_dir.glob('*.jpg')))
print(image_count)
print(data_dir)
REAGENTE = list(data_dir.glob('REAGENTE*'))
len_REAGENTE=len(list(data_dir.glob('REAGENTE*')))
print (len_REAGENTE)
NAO_REAGENTE = list(data_dir.glob('NAO_REAGENTE*'))
len_REAGENTE=len(list(data_dir.glob('NAO_REAGENTE*')))
print (len_REAGENTE)
batch_size = 50
img_height = 65
img_width = 375
```

Fonte: Autoria própria (2023)

Também foi necessário criar dois subconjuntos contendo imagens selecionadas de forma aleatória para serem usados para treinamento e validação da rede neural. O conjunto de validação tem o tamanho de trinta por cento do tamanho total do banco de imagens. Os setenta por cento restantes das imagens é usado para

o treinamento. A separação das imagens nesses dois subconjuntos pode ser visualizada no trecho de código da figura 36.

Figura 36 - Trecho de código criando conjunto de validação e treino

```
train_ds = tf.keras.utils.image_dataset_from_directory(
 data_dir,
 validation_split=0.3,
 subset="training",
 seed=123,
 image_size=(img_height, img_width),
 batch_size=batch_size)
```

Fonte: Autoria própria (2023)

Foram criadas 10 camadas sequenciais, sendo a primeira camada apenas utilizada para definir a escala para os valores de entrada, seguida por 3 sequências de uma camada de convolução e uma camada de *pooling*. Essas camadas são seguidas de uma camada *flatten*, que modela os dados em um vetor e então duas camadas *Dense* para gerar a saída. A criação das camadas mencionadas pode ser visualizada no trecho de código na figura 37.

Figura 37 - Camadas da CNN

```
model = tf.keras.Sequential([
 tf.keras.layers.Rescaling(1./255),
 tf.keras.layers.Conv2D(32, 4, activation='relu'),
 tf.keras.layers.MaxPooling2D(),
 tf.keras.layers.Conv2D(32, 4, activation='relu'),
 tf.keras.layers.MaxPooling2D(),
 tf.keras.layers.Conv2D(32, 4, activation='relu'),
 tf.keras.layers.MaxPooling2D(),
 tf.keras.layers.Flatten(),
 tf.keras.layers.Dense(128, activation='relu'),
 tf.keras.layers.Dense(num_classes)])
```

Fonte: Autoria própria (2023)

Com a rede definida, foi necessário compilar. Isso foi feito passando como parâmetros o otimizador, a função de perda e as métricas a serem avaliadas. Essa etapa pode ser visualizada no trecho de código a seguir.

Figura 38 - Trecho de código para compilar a CNN

```
model.compile(
 optimizer='adam',
 loss='mse',
 metrics=['accuracy',
   tf.keras.metrics.TruePositives(),
   tf.keras.metrics.TrueNegatives(),
   tf.keras.metrics.FalsePositives(),
   tf.keras.metrics.FalseNegatives()])
```

Fonte: Autoria própria (2023)

Para iniciar o treinamento, são passados como parâmetros os conjuntos de imagem de treino e validação e o número de épocas (número de iterações sobre todo o conjunto de imagens). O trecho de código da figura 39 exibe essa etapa.

Figura 39 - Trecho de código da etapa de treinamento da CNN

```
history = model.fit(
 train_ds,
 validation_data=val_ds,
 epochs=15,
 shuffle=True,
 verbose=2)
```

Fonte: Autoria própria (2023)

Por fim, o modelo criado para cada doença foi salvo e convertido para o TensorFlow Lite, estando assim pronto para ser utilizado pela leitora. O código utilizado para esse procedimento pode ser visualizado na figura 40.

Figura 40 - Convertendo o modelo para TFLite

```
model.save(MODEL_DIR)
converter = tf.lite.TFLiteConverter.from_saved_model(MODEL_DIR)
converter.target_spec.supported_ops = [
        tf.lite.OpsSet.TFLITE_BUILTINS,
        tf.lite.OpsSet.SELECT_TF_OPS
]
tflite_model = converter.convert()
```

Fonte: Autoria própria (2023)

3.3.3 Resultados obtidos pela rede neural

Após o treinamento da rede neural, o conjunto de imagens de validação foi utilizado para gerar algumas métricas de desempenho. Para este trabalho foram analisados os valores obtidos de “Falso negativo”, “Falso positivo”, “Verdadeiro negativo”, e “Verdadeiro positivo”. Esses valores foram utilizados para criar a matriz confusão para cada uma das categorias dos cassetes de diferentes doenças utilizadas no trabalho.

3.3.3.1 Resultados do treinamento da rede neural para Covid-19

A CNN de classificação de resultados de cassetes de Covid-19 foi treinada com 707 imagens de cassetes, das quais 212 foram utilizadas para validação. A acurácia obtida pela CNN na etapa de validação pode ser observada na matriz confusão da tabela 2.

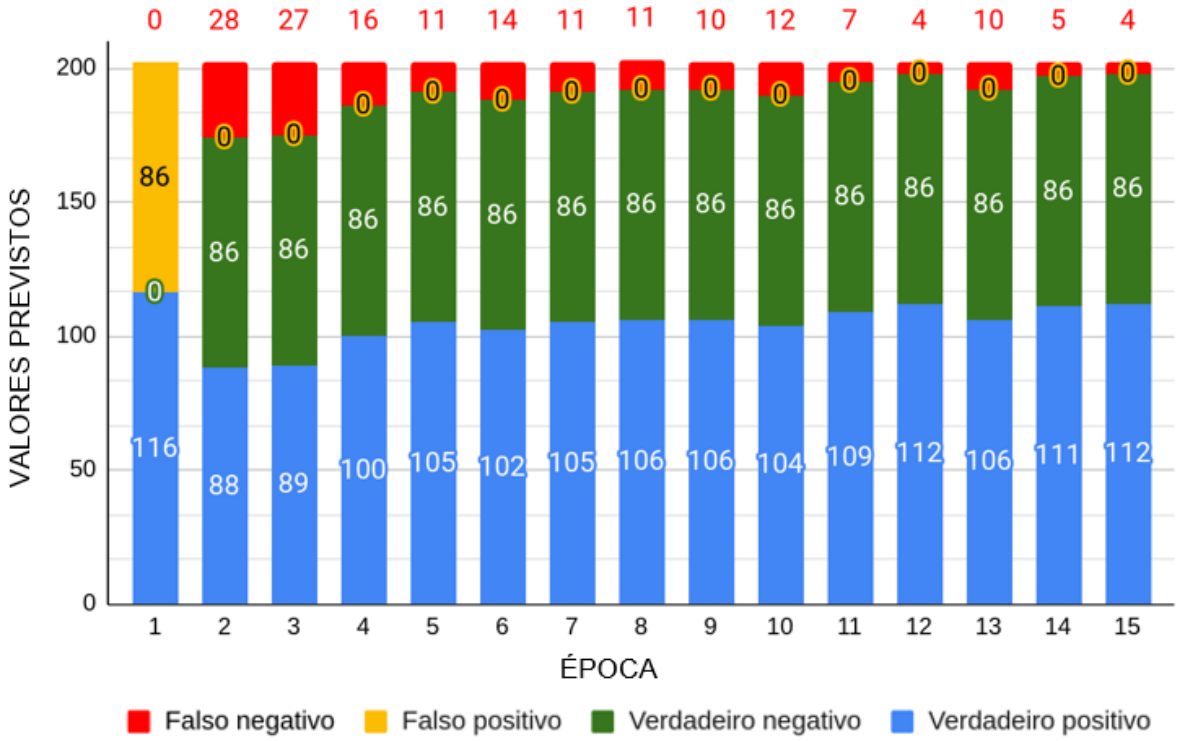
Tabela 2 - Matriz confusão (Covid-19)

		VALOR PREVISTO	
VALOR		Reagente	Não reagente
	Reagente	112	4
	Não reagente	0	86

Fonte: Autoria própria (2023)

Esse resultado foi obtido utilizando 12 épocas para o treinamento da CNN, sendo 12 o valor definido ao observar que não houve ganho de acurácia com o aumento no número de épocas no treinamento. No gráfico a seguir podemos observar como os resultados das classificações das imagens do conjunto de validação, estando exibidos o número de imagens correta e erroneamente classificadas ao longo do treinamento até chegar a época de número 15, onde o treinamento foi terminado. Podemos verificar 4 erros, sendo todos eles do tipo “Falso negativo”.

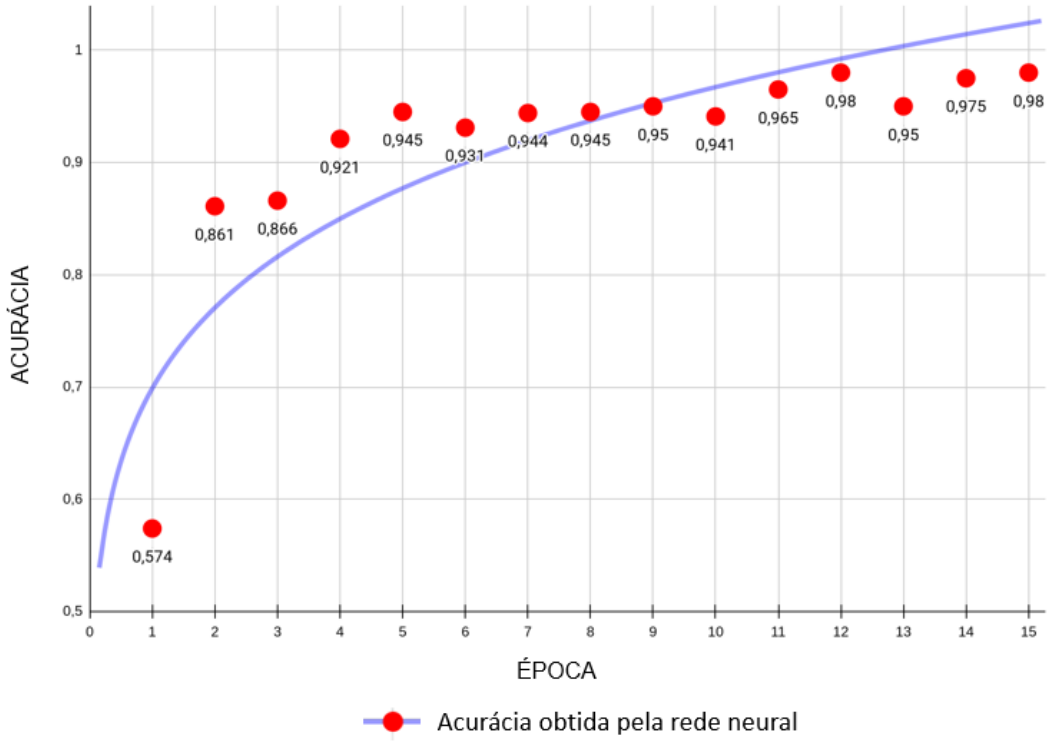
Gráfico 2 - Valores previstos por época (Covid-19)



Fonte: Autoria própria (2023)

A acurácia também foi observada para avaliar a CNN, sendo obtida a partir dos valores dos resultados da classificação do conjunto de validação. A evolução da acurácia por época no treinamento pode ser observada no gráfico 3.

Gráfico 3 - Acurácia por época (Covid-19)



Fonte: Autoria própria (2023)

3.3.3.2 Resultados do treinamento da rede neural para hepatite

A CNN de classificação de resultados de cassetes de hepatite foi treinada com 454 imagens de cassetes, das quais 136 foram utilizadas para validação. A acurácia obtida pela CNN na etapa de validação pode ser observada na matriz confusão da tabela 3.

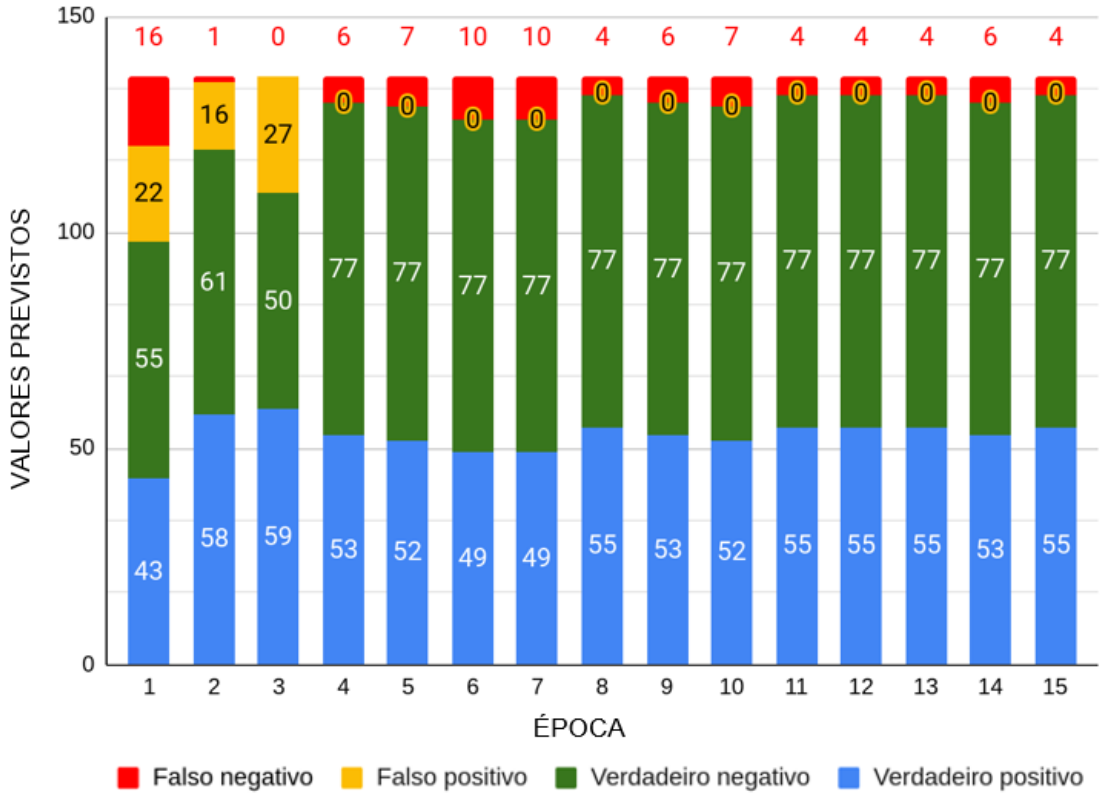
Tabela 3 - Matriz confusão (Hepatite)

		VALOR PREVISTO	
VALOR		Reagente	Não reagente
	Reagente	55	4
	Não reagente	0	77

Fonte: Autoria própria (2023)

A classificação dada pela CNN durante o treinamento pode ser observado no gráfico 4, onde ao final do treinamento (época de número 15) podemos observar que apenas 4 imagens foram erroneamente classificadas, todas elas como “Falso negativo”.

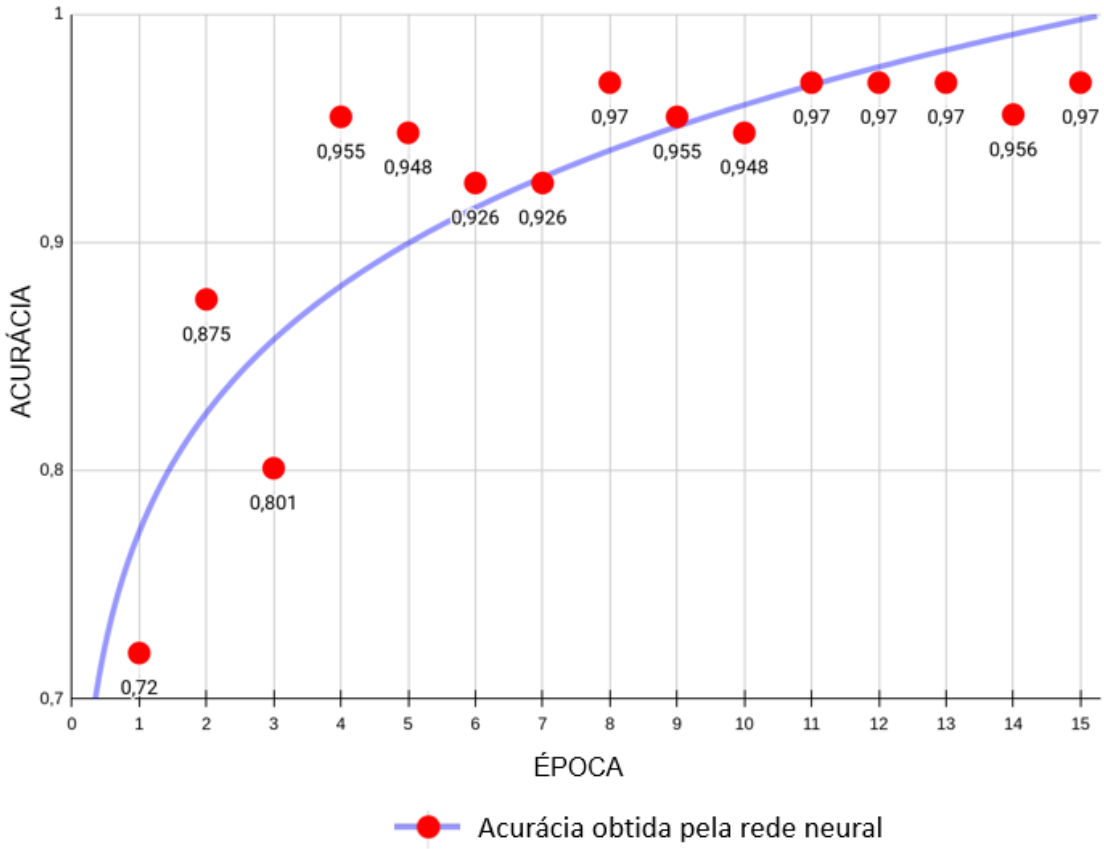
Gráfico 4 - Valores previstos por época (Hepatite)



Fonte: Autoria própria (2023)

Por fim, foram verificados os valores de acurácia obtidos pela rede neural durante a validação. Os valores obtidos por época podem ser visualizados no gráfico a seguir.

Gráfico 5 - Acurácia por época (Hepatite)



Fonte: Autoria própria (2023)

3.3.3.3 Resultados do treinamento da rede neural para HIV

A CNN de classificação de resultados de cassetes de HIV foi treinada com 367 imagens de cassetes, das quais 191 foram utilizadas para validação. A acurácia obtida pela CNN na etapa de validação pode ser observada na matriz confusão da tabela 4.

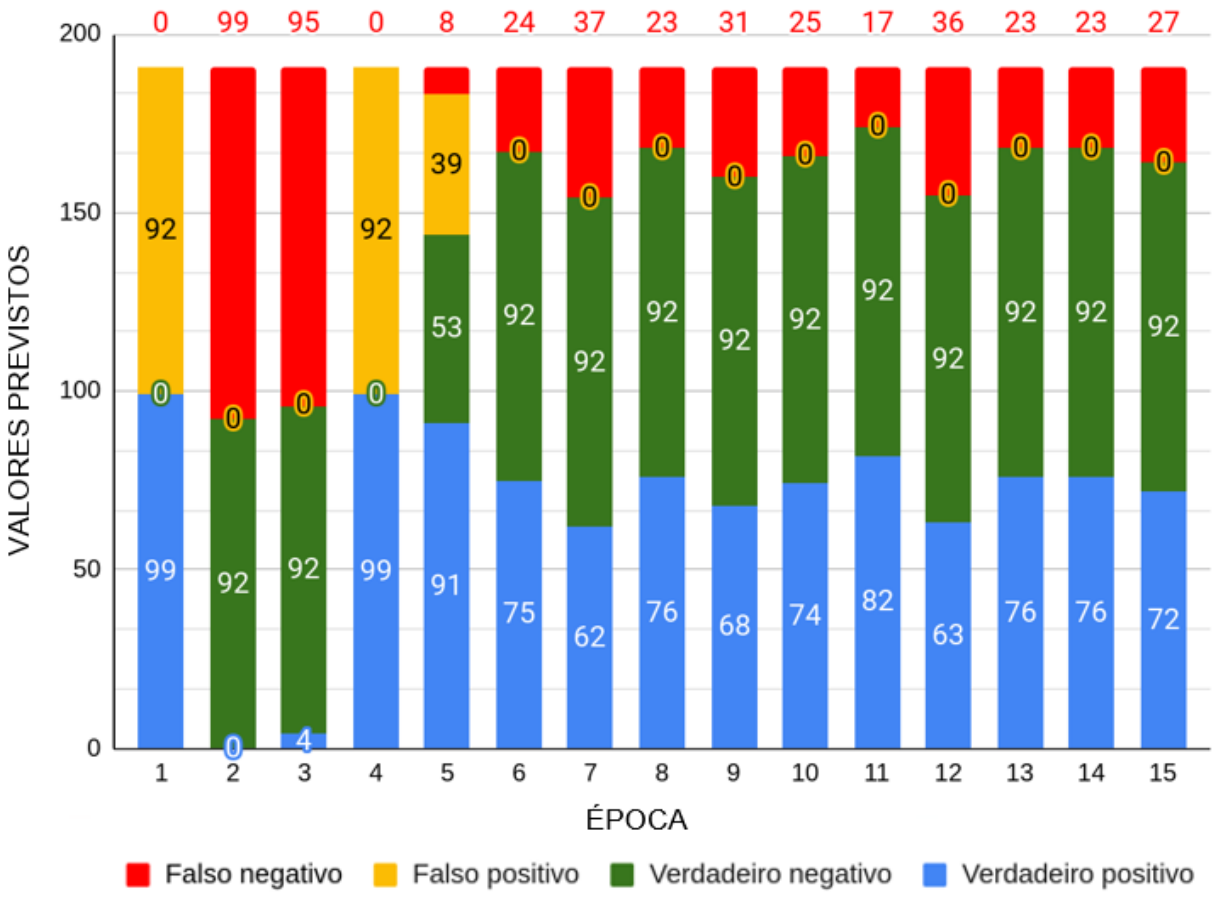
Tabela 4 - Matriz confusão (HIV)

		VALOR PREVISTO	
VALOR		Reagente	Não reagente
	Reagente	72	27
	Não reagente	0	92

Fonte: Autoria própria (2023)

No gráfico 6, podemos observar que ao final do treinamento (época de número 15), 27 das imagens do conjunto de validação ainda estavam sendo erroneamente classificadas, todas elas na categoria de “Falso negativo”. Sendo essa a CNN que obteve o pior resultado na sua capacidade de classificação, obtendo uma acurácia de 85,86%.

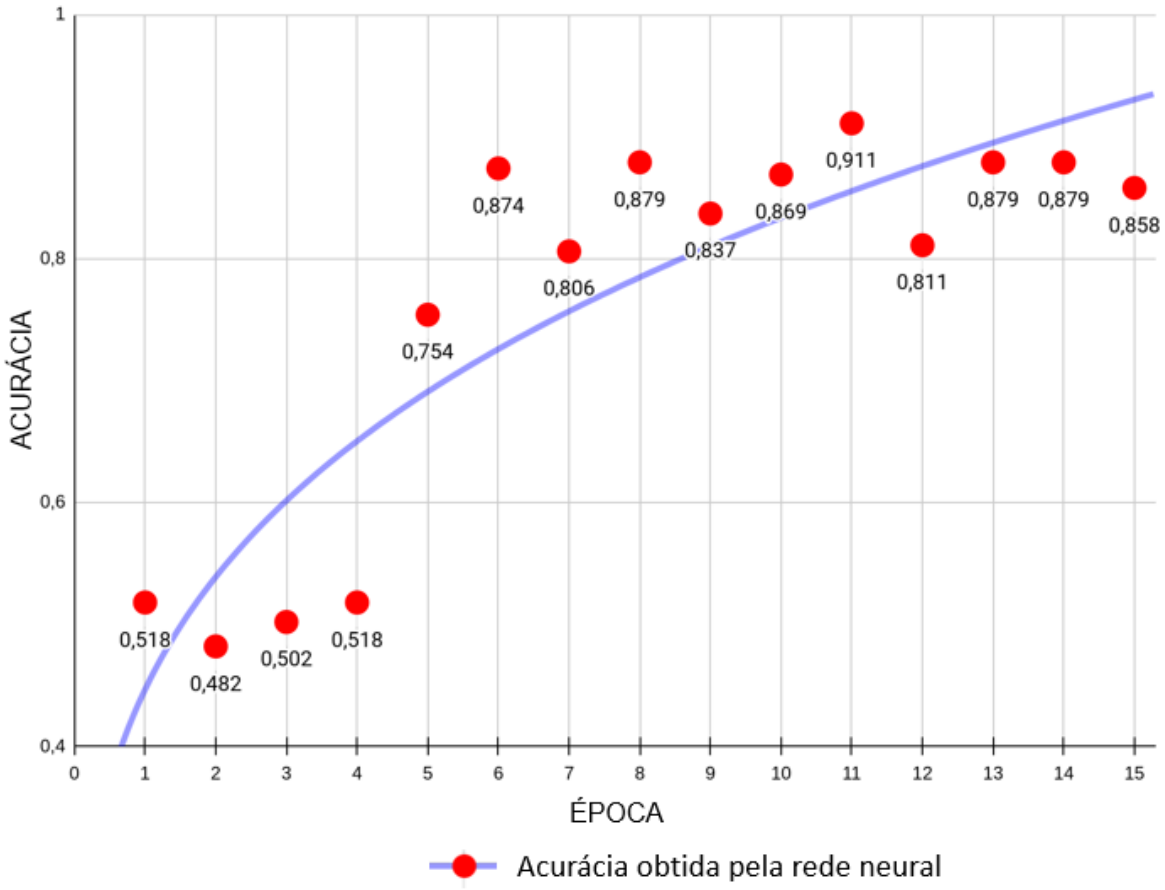
Gráfico 6 - Valores previstos por época (HIV)



Fonte: Autoria própria (2023)

O gráfico 7 permite observar a acurácia obtida pela CNN com maior clareza, onde o valor 85,86% de imagens corretamente classificadas pode ser notada da época de número 15.

Gráfico 7 - Acurácia por época (HIV)



Fonte: Autoria própria (2023)

3.4 Salvando resultados obtidos

A leitora é capaz de gravar os resultados obtidos nas classificações em um arquivo de texto, ficando salvo o tipo de teste lido, o resultado e o horário em que a leitura foi realizada. No trecho de código da figura 41, podemos visualizar o processo de abrir o arquivo e adicionar uma nova linha com as informações referentes à última leitura realizada.

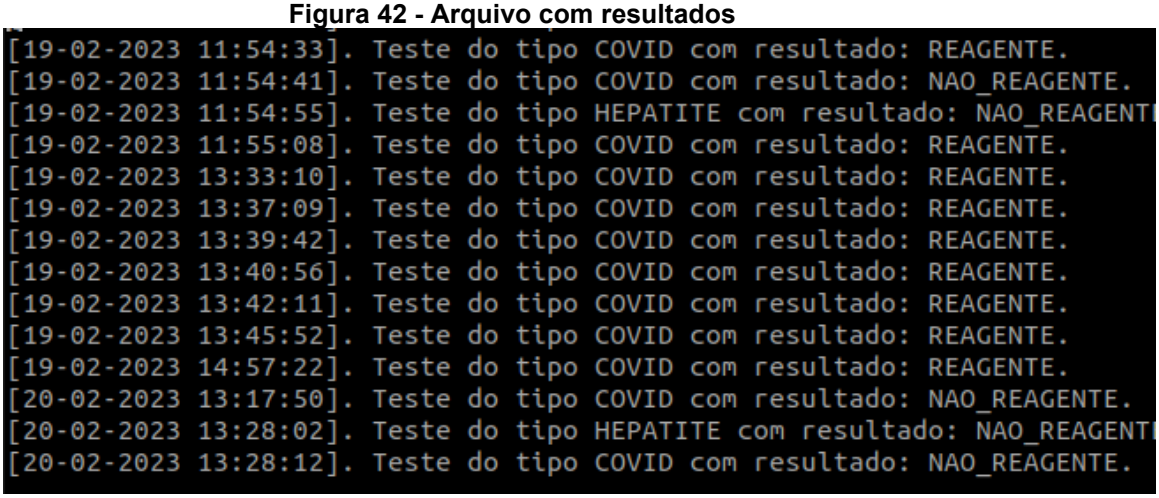
Figura 41 - Salvando resultados em arquivo de texto

```
timeNow = datetime.datetime.now()
time = timeNow.strftime('%d-%m-%Y %H:%M:%S')
with open(logpath, mode='a', encoding='utf-8') as arquivo:
        arquivo.write(f'[{time}]. Teste do tipo {resultCassete} com      resultado: {result}.\n')
        arquivo.close()
```

Fonte: Autoria própria (2023)

Na figura 42 temos uma captura referente ao arquivo em que os resultados são salvos. Nele podemos visualizar o horário da leitora, o tipo de teste e o resultado segundo a classificação da leitora.

Figura 42 - Arquivo com resultados



Fonte: Autoria própria (2023)

3.5 Hardware

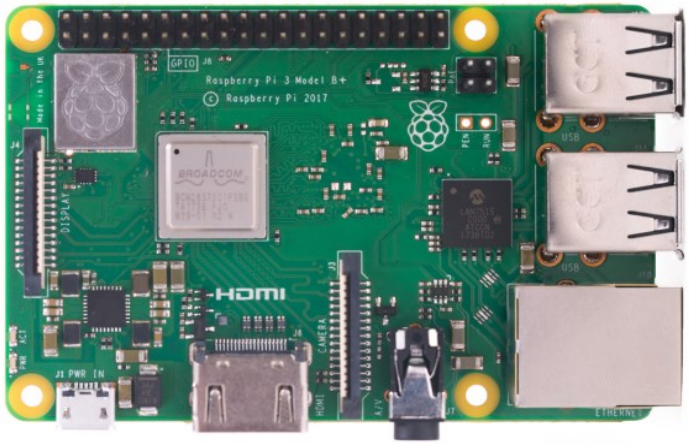
Nesta seção iremos abordar os materiais e equipamentos utilizados para construir o protótipo da leitora de testes rápidos de fluxo lateral.

3.5.1 Raspberry Pi 3 B+

O Raspberry Pi 3 B+ é um minicomputador desenvolvido pela Raspberry Pi Foundation, uma fundação sem fins lucrativos sediada no Reino Unido. Pertencente à família de minicomputadores Raspberry Pi, esses dispositivos foram inicialmente desenvolvidos para fins educacionais, porém com o aumento de popularidade e ganho de robustez, hoje podem ser encontrados até mesmo em ambientes industriais ou áreas como a IoT.

O modelo Raspberry Pi 3 B+, conta com um processador *quad core* 64-*bit* de 1.4 GHz Broadcom BCM2837B0 com GPU integrada VideoCore IV de 400MHz, e 1 GB de memória LPDDR2 SDRAM. Esse minicomputador também conta com 40 pinos GPIO, 4 portas USB 2.0, *wireless* de 2.4GHz e 5GHz, saídas de áudio e vídeo, conector para câmera etc.

Figura 43 - Top view Raspberry Pi 3 B+



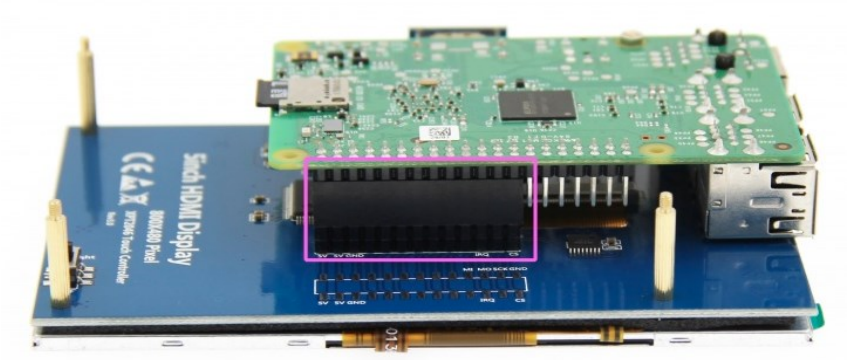
Fonte: Raspberry Pi 3 B+ Product Overview (2022)

Os minicomputadores da família Raspberry Pi contam com um sistema operacional dedicado, o Raspberry Pi OS. Esse sistema fica armazenado em um cartão de memória e possui todas as funcionalidades básicas de um sistema operacional GNU/Linux e interface gráfica.

3.5.2 Display LCD

O XPT2046 é um display de 5 polegadas com resolução de 800x400 e alimentação 5V. Possui interface HDMI e tem suporte para o sistema operacional Raspberry Pi OS.

Figura 44 - Display XPT2046 conectado a um Raspberry Pi



Fonte: LCD Wiki (2022)

3.5.3 Câmera

A câmera utilizada foi a Raspberry Pi Camera Board v1.3 de 5MP em conjunto com uma lente de 2,1mm. A câmera é capaz de gerar imagens com resolução de 2592x1944 e vídeo de 1080p a 30FPS, sendo totalmente compatível com os minicomputadores da família Raspberry Pi e o sistema operacional Raspberry Pi OS.

A lente utilizada possui ajuste de foco e foi escolhida de forma empírica entre um conjunto de lentes, sendo a que apresentou um melhor resultado considerando as limitações de luminosidade e distância entre a câmera e cassete posicionado no interior da estrutura do protótipo.

A lente e câmera utilizadas nesse trabalho foram fornecidas pelo LaCTAS para o desenvolvimento desse projeto e podem ser visualizadas na figura 45.

Figura 45 - Câmera com lente 2,1mm

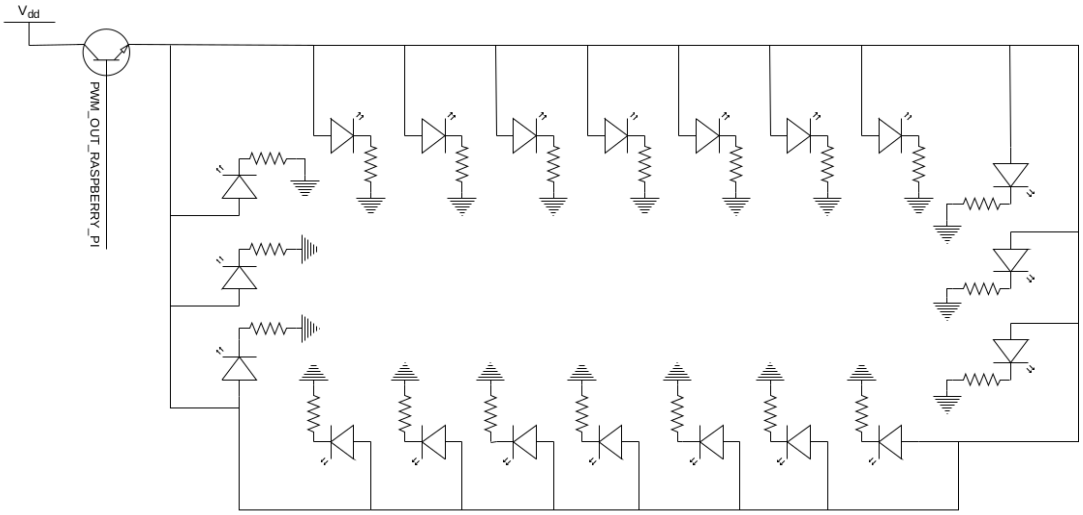
Fonte: Autoria própria (2023)

3.5.4 Iluminação

O protótipo de leitora conta com uma estrutura fechada, sendo assim necessário um sistema de iluminação para otimizar a captura de imagens dos cassetes com a câmera. Para esse propósito, foi desenvolvido um *hardware* de iluminação para a leitora. O circuito conta com 20 LEDs SMD de alto brilho em paralelo, alimentados por uma fonte de 5V. Cada LED está em série com um resistor de 100Ω e intensidade de iluminação é controlada por um PWM gerado pelo Raspberry Pi e aplicado na base de um transistor. A intensidade de iluminação foi definida de forma empírica, ajustando o valor do PWM de forma que se obtivesse a melhor qualidade de imagem.

O circuito utilizado para iluminação pode ser visualizado na figura 46.

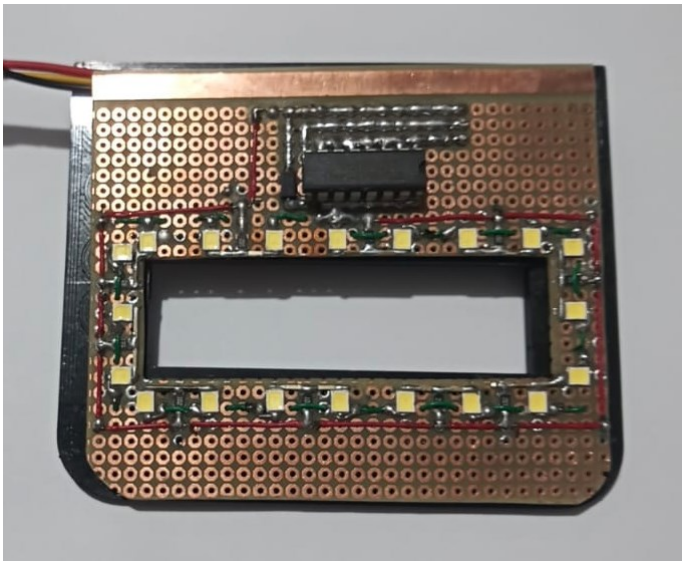
Figura 46 - Circuito de iluminação



Fonte: Autoria própria (2023)

O circuito foi montado de forma a ficar em volta da câmera, mas sem gerar sombras na imagem, direcionando a iluminação para direção onde o cassete estará posicionado no interior da leitora. Na figura 47 podemos visualizar o circuito montado.

Figura 47 - Hardware de iluminação



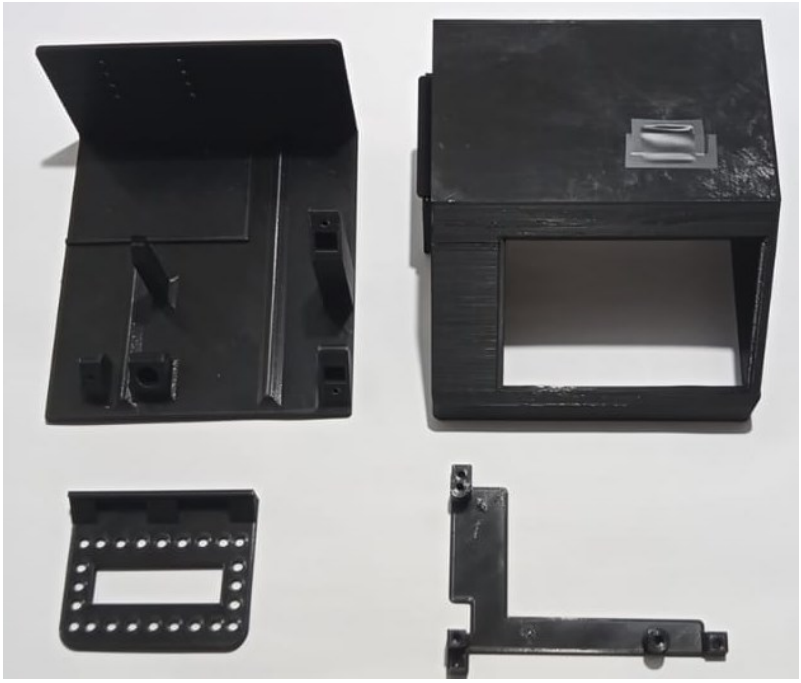
Fonte: Autoria própria (2023)

3.5.5 Gabinete

O gabinete foi desenvolvido em parceria com o LaCTAS, onde ele foi desenhado em *software* de modelagem 3D e construído via manufatura aditiva. O

gabinete foi estruturado de forma a permitir o posicionamento e fixação de cada componente do protótipo e é composto por 4 partes montáveis. Na figura 48 estão expostas a partes que compões essa estrutura.

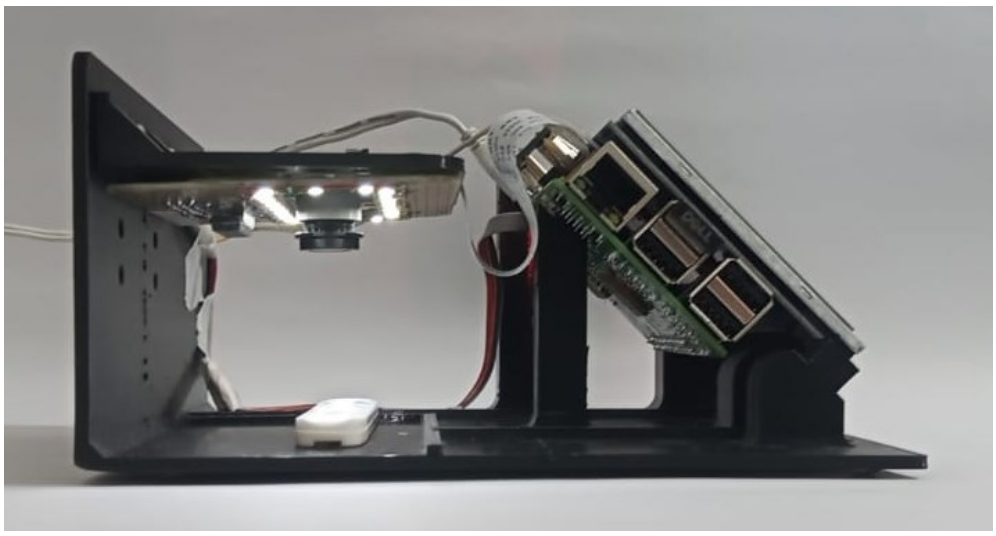
Figura 48 - Partes que compõem o gabinete da leitora



Fonte: Autoria própria (2023)

Essa estrutura em conjunto com os componentes anteriormente citados foram uma leitora de testes rápidos de fluxo lateral, onde um cassete de testes é inserido no seu interior, é iluminado pelo sistema de iluminação e permite que o usuário interaja com a leitora através do display. Todo o sistema é controlado pelo Raspberry Pi, onde também é processada a rede neural responsável por identificar o cassete inserido na leitora e indicar o seu resultado. Na figura 49 podemos observar a leitora montada.

Figura 49 - Vista lateral da leitora

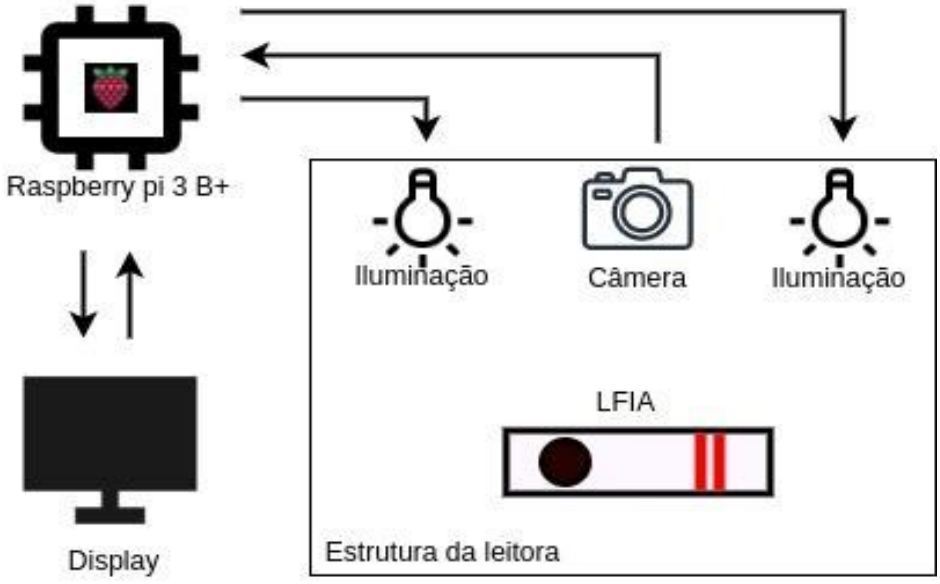


Fonte: Autoria própria (2023)

3.6 Protótipo

O protótipo construído e seus componentes pode ser visualizado no diagrama da figura 50, onde podemos observar como se dá o funcionamento conjunto dos componentes já mencionados que fazem parte da leitora.

Figura 50 - Diagrama da leitora



Fonte: Autoria própria (2023)

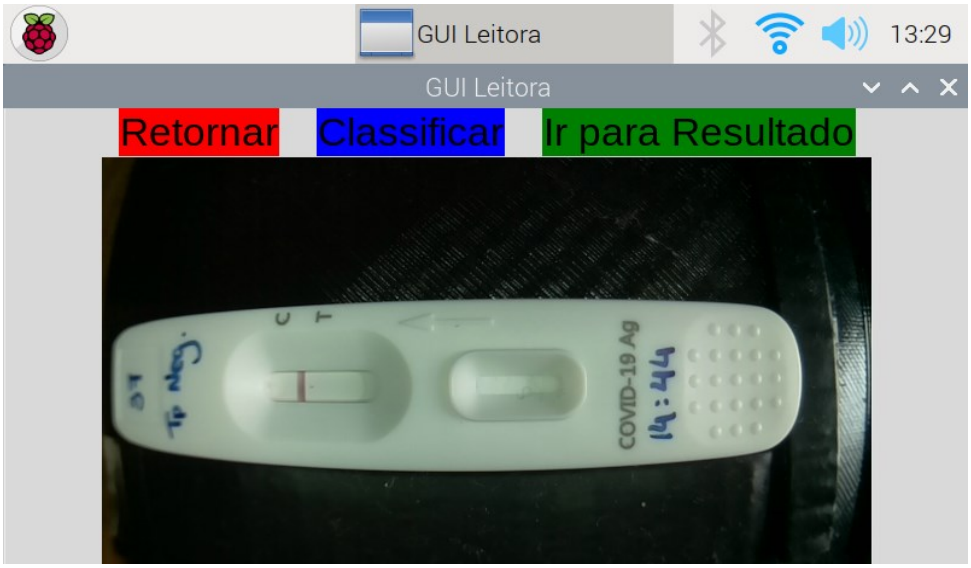
O protótipo foi desenvolvido de forma que pudesse ser facilmente manuseado e transportado, com o objetivo de dar origem a uma ferramenta portátil e que possa ser utilizada em diversos ambientes. O protótipo montado e em funcionamento pode ser visualizado na figura 51.

Figura 51 - Leitora montada

Fonte: Autoria própria (2023)

Uma interface gráfica foi desenvolvida para o protótipo, onde o usuário necessita apenas inserir o cassete na leitora e alguns cliques para obter uma classificação gerada pela leitora. A interface pode ser visualizada na figura 52, onde temos uma captura de tela feita na leitora durante o seu uso.

Figura 52 - Interface da leitora



Fonte: Autoria própria (2023)

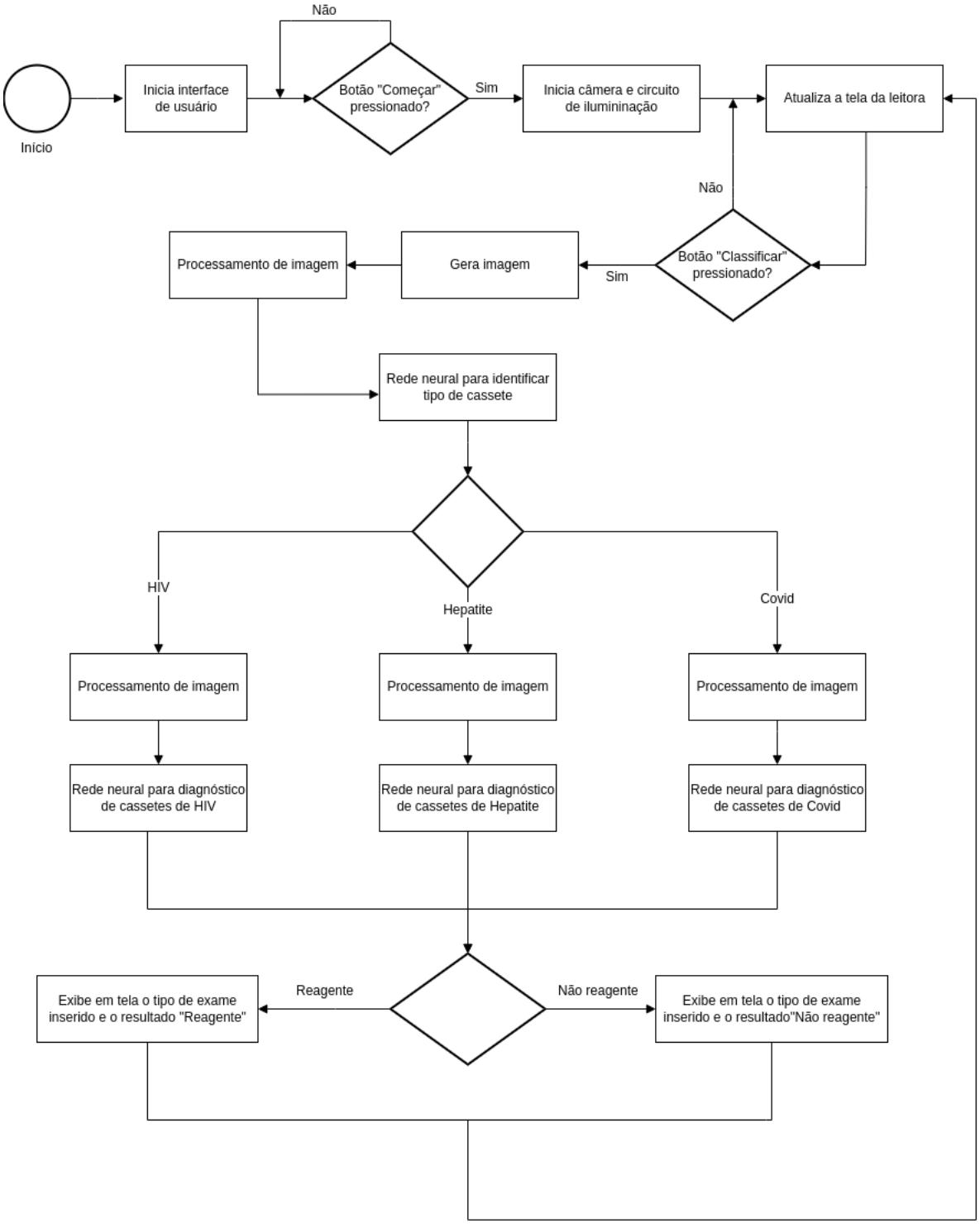
Na interface é possível visualizar o resultado gerado pela leitora para a classificação do cassete, sendo esse exibido em uma nova tela através dos rótulos “REAGENTE” ou “NÃO REAGENTE”. Na imagem a seguir temos um exemplo de um cassete inserido no interior da leitora e o resultado exibido.

Figura 53 - Tela de exibição do cassete e tela de resultado

Fonte: Autoria própria (2023)

Na figura 54, podemos verificar o funcionamento da leitora através do diagrama de atividades.

Figura 54 - Diagrama de atividades da leitora



Fonte: Autoria própria (2023)

4 TESTES E RESULTADOS

Os testes para validar a leitora e suas funcionalidades foram feitos de forma a testar a sua capacidade de diferenciar diferentes tipos de cassetes e classificar os exames conforme resultado obtido. Para tal validação foram realizados dois tipos de teste de forma separada, um apenas para validar a correta identificação do tipo de cassete e um para validar a correta classificação conforme o resultado positivo ou negativo.

Os testes foram feitos a partir das imagens geradas pela própria leitora através da sua câmera e com a iluminação via LEDs presentes na sua estrutura de iluminação. Assim sendo, as imagens geradas para a leitora para realizar a classificação e as imagens utilizadas para o treinamento da rede neural foram obtidas com câmeras diferentes e com diferente iluminação. Os cassetes utilizados para o treinamento são dos exatos mesmos modelos dos cassetes presentes nas imagens que foram usadas para o treinamento da rede neural.

Os testes se resumiram a inserir cassetes de testes rápidos de fluxo lateral de diferentes doenças no protótipo da leitora e obter uma classificação do modelo de teste (Covid-19, HIV ou hepatite) e resultado apontado (reagente ou não reagente).

4.1 Classificação de diferentes tipos de cassetes

Para esse teste, foram utilizados exemplares de testes rápidos de fluxo lateral que passaram por uso em laboratório e foram doados para o propósito desse trabalho. Estavam disponíveis testes de 3 diferentes doenças em diferentes números, conforme a tabela 5.

Tabela 5 - Cassetes utilizados no teste da leitora

Tipo de cassete	Total disponível
Covid-19	77
Hepatite	10
HIV	10

Fonte: Autoria própria (2023)

Os cassetes utilizados para os testes são semelhantes aos encontrados nos bancos de imagens. Na figura a seguir podemos visualizar exemplos de testes de Hepatite, HIV e Covid-19 respectivamente.

Figura 55 - Exemplos de exames utilizados para testes

Fonte: Autoria própria (2023)

Para o teste, foram utilizados 10 cassetes de cada um dos modelos como os apresentados na figura 55, esse número foi definido em função da restrição no número de exemplares disponíveis para os testes de HIV e hepatite. Para os exemplares referentes a Covid-19, foram selecionados 10 exemplares de forma aleatória.

Os testes foram misturados e um a um foram selecionados de forma aleatória e inseridos na leitora para a classificação.

Para esse teste, apenas foi observado qual a classificação dada pela leitora referente ao **modelo do cassete**, não sendo levado em consideração qual o resultado apontado pelo teste. Buscou-se apenas observar a capacidade da leitora classificar um cassete para teste de Covid-19 como um cassete de Covid-19, um de HIV como cassete para teste de HIV etc.

Na tabela 6, estão sendo exibidos os resultados desse teste, onde temos os 10 cassetes referentes a cada tipo de doença e qual foi a classificação dada pela leitora.

Tabela 6 - Classificando diferentes tipos de cassetes

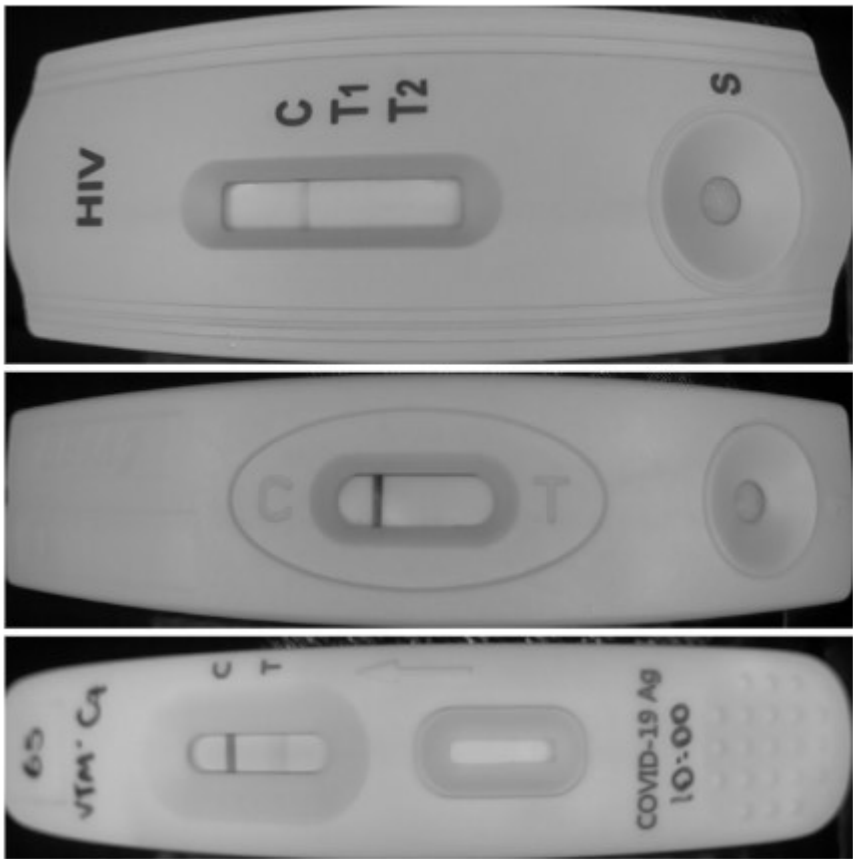
	CLASSIFICAÇÃO DADA PELA LEITORA			
CASSETE	Covid-19	Hepatite	HIV	**TOTAL**
Covid-19	10	0	0	10
Hepatite	0	10	0	10
HIV	0	0	10	10

Fonte: Autoria própria (2023)

Podemos observar, que dos 30 cassetes testados, todos eles foram corretamente classificados pela leitora. Mostrando a capacidade da leitora de diferenciar entre os modelos de cassete os quais ela se propõe a classificar.

Na figura 56 estão alguns exemplos de imagens obtidas pela leitora durante esse teste. Essas imagens foram geradas pela câmera da leitora e mostram os diferentes modelos de cassetes inseridos para a classificação. Temos na figura uma foto tirada de um teste de HIV, hepatite e Covid-19 respectivamente.

Figura 56 - Exemplos de imagens geradas pela leitora para classificação



Fonte: Autoria própria (2023)

4.2 Classificação conforme resultado obtido

Para validar a capacidade da leitora de classificar imunoensaios de fluxo lateral conforme seu resultado (reagente ou não reagente), foram utilizados 77 testes de Covid-19 que foram doados para o propósito desse trabalho. Esse teste foi assim elaborado levando em consideração as limitações existentes quanto ao número de testes e por apenas haver a disponibilidade de testes de Covid-19 que passaram por uso real em laboratório e terem sido previamente classificados e marcados por profissionais da área da saúde. Não havia a disponibilidade desse tipo de material (com uso real e marcações de resultado) para os testes de HIV e hepatite.

Os testes foram separados em diferentes categorias sendo elas testes reagentes com alta carga viral, reagentes com baixa carga viral e não reagentes.

Na tabela 7, podemos visualizar como foram distribuídos os cassetes em cada uma das categorias citadas.

Tabela 7 - Cassetes divididos conforme banda de teste

Categoria	Total
Reagente - alta carga viral	46
Reagente - baixa carga viral	19
Não reagente	12

Fonte: Autoria própria (2023)

Essas categorias foram definidas de acordo com o grau de visibilidade da listra de teste no cassete. Sendo que um teste com alta carga viral apresenta listras bastante visíveis, um de baixa carga viral apresenta a listra de teste de forma menos visível e um não reagente apenas possui a listra de controle.

Essas categorias foram criadas para evidenciar a capacidade da leitora de classificar corretamente cassetes de “maior dificuldade” de classificação. Onde um cassete de alta carga viral e listras de fácil visualização dificilmente será erroneamente classificado, porém na categoria de baixa carga viral, é onde a maior dificuldade se encontra e é onde a maior parte dos erros humanos de classificação ocorrem.

Na figura 57 estão exemplos de testes referentes a cada uma das categorias, onde podem ser observadas as diferenças visuais entre as categorias.

Figura 57 - Alta e baixa carga viral e não reagente, respectivamente

Fonte: Autoria própria (2023)

O teste foi realizado com todos os 77 cassetes e a classificação dada pela leitora foi anotada e comparada com a marcação feita pelos profissionais da saúde no corpo do cassete. Os casos de acerto são os cassetes do tipo “Não reagente” sendo classificados como “Não reagente” e os cassetes das categorias “Reagente - alta carga viral” e “Reagente - baixa carga viral” sendo classificados como “Reagente”. Qualquer outro resultado é considerado um erro.

Na tabela 8 estão exibidos os resultados obtidos, estando os cassetes referentes a cada categoria e a classificação dada pela leitora.

Tabela 8 - Resultados obtidos pela leitora

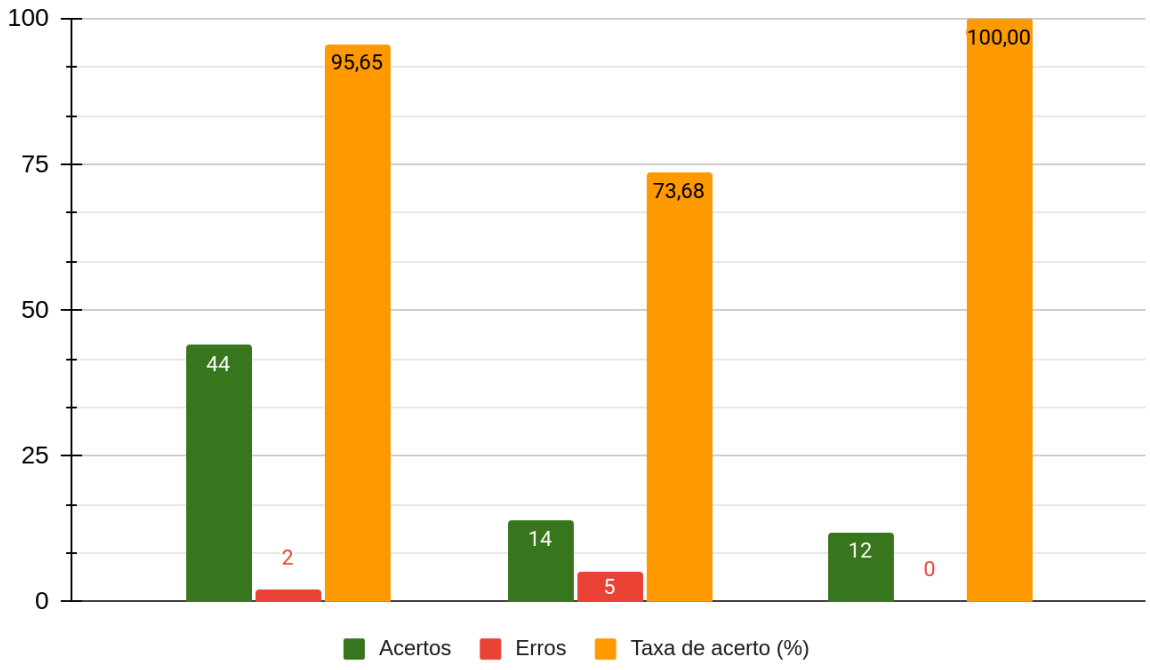
	CLASSIFICAÇÃO DADA PELA LEITORA		
CATEGORIA	Reagente	Não reagente	**Total**
Reagente - alta carga viral	44	2	46
Reagente - baixa carga viral	14	5	19
Não reagente	0	12	12

Fonte: Autoria própria (2023)

Podemos observar que a leitora classificou corretamente todos os cassetes do tipo “Não reagente” e teve o pior desempenho entre os cassetes do tipo “Reagente - baixa carga viral”, onde 5 dos 19 cassetes foram classificados como “Não reagente”.

No gráfico 8 podemos ver a taxa de acerto referente a cada categoria.

Gráfico 8 - Resultados obtidos



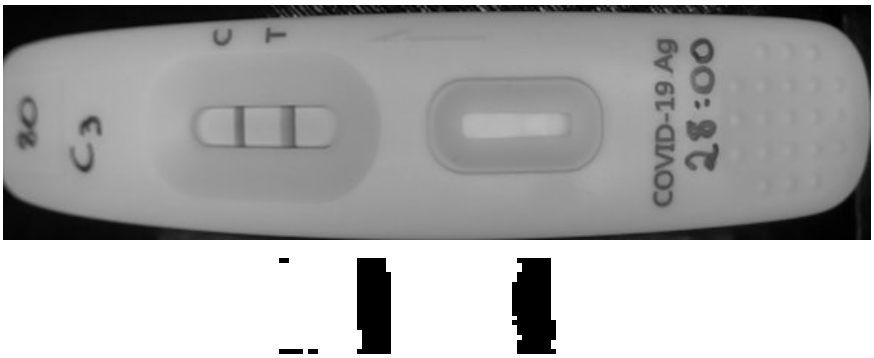
Fonte: Autoria própria (2023)

Podemos também observar que dos 77 cassetes, 70 deles foram corretamente classificados, obtendo assim uma taxa de acerto global de aproximadamente 91%.

Nas imagens a seguir, podemos visualizar as imagens geradas pela leitora durante esse teste, onde temos a foto tirada pela leitora do cassete e a imagem das listras de teste geradas ao final do processamento da imagem conforme demostrado anteriormente na seção 3.2.

Na figura 58, temos um cassete de Covid-19 com resultado positivo e alta carga viral, como podemos observar na nitidez da banda de teste.

Figura 58 - Exemplo de cassete de resultado reagente com alta carga viral



Fonte: Autoria própria (2023)

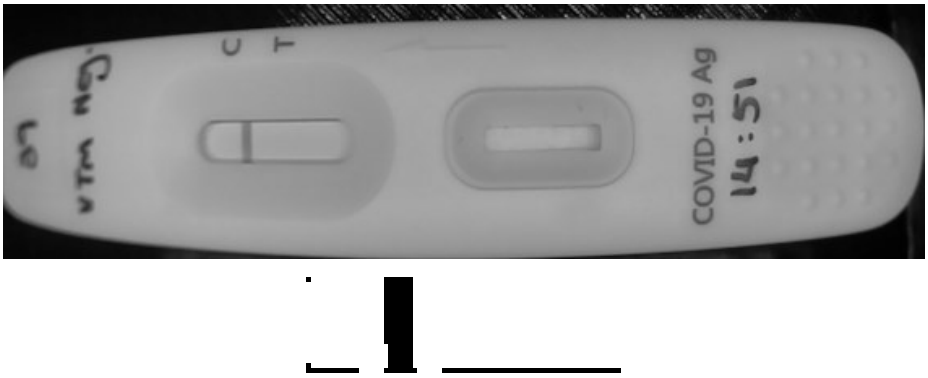
Na figura 59, temos um cassete de Covid-19 com resultado positivo e baixa carga viral, como podemos observar na banda de teste sendo pouco visível.

Figura 59 - Exemplo de cassete de resultado reagente com baixa carga viral

Fonte: Autoria própria (2023)

Na figura 60, temos um cassete de Covid-19 com resultado negativo, conforme podemos observar na ausência da banda de teste.

Figura 60 - Exemplo de cassete de resultado não reagente



Fonte: Autoria própria (2023)

5 CONSIDERAÇÕES FINAIS

Este trabalho propôs o desenvolvimento de uma leitora testes rápidos de fluxo lateral com baixo custo de fabricação e que possa ser utilizada em ambientes diversos como as grandes campanhas de testagem de doenças realizadas pelo SUS nos mais contrastantes cantos do país. Dessa forma espera-se disponibilizar uma ferramenta que possa servir de base para um contínuo refino do projeto, podendo se tornar um produto viável nos aspectos técnicos e econômicos para sua utilização por profissionais da saúde em território nacional.

5.1 Dificuldades encontradas

Durante o desenvolvimento, algumas limitações foram encontradas, uma delas sendo a escolha da lente a ser utilizada na câmera. Das lentes disponíveis para uso, foi necessário encontrar a que melhor se adequasse ao tamanho dos cassetes e à altura do gabinete (limita a distância entre a câmera e o cassete), gerando imagens que capturassem todo o corpo do teste rápido sem gerar grandes distorções nas bordas da imagem. Devido aos diferentes tamanhos e formatos dos cassetes e a limitações da câmera, não foi possível ter imagens ideais para todos os modelos de cassete, tendo se optado por otimizar o projeto para os testes de Covid-19.

Uma dificuldade contornada neste trabalho foi a presença de marcações no corpo dos cassetes do banco de imagens usado para treinamento das CNNs. Essas marcações foram feitas pelos profissionais de saúde que utilizaram esses exames em laboratório. Essas marcações eram interpretadas pela rede neural e acabavam gerando efeitos indesejáveis que comprometiam os resultados obtidos pela leitora. Este problema foi contornado adicionando uma etapa ao pré-processamento para limitar a área de interesse da imagem, contendo-se apenas às regiões das listras de teste e controle.

A iluminação do interior da estrutura da leitora também encontrou dificuldades, sendo necessário iluminar o cassete de forma que não houvesse sombras e que a claridade fosse alta o suficiente para que a câmera gerasse imagens de qualidade. A solução, como demonstrada neste trabalho, foi construir uma estrutura de LEDs de alta luminosidade em forma de retângulo, contornando a câmera e perpendicular ao corpo do cassete.

5.2 Desenvolvimento futuro

Tendo em vista que o trabalho foi desenvolvido com a intenção de atender demandas reais encontradas pelo SUS, é importante considerar potenciais melhorias e novas funcionalidades que podem ser futuramente implementadas, adicionando robustez e praticidade ao projeto.

Uma possível melhoria, seria a criação de uma nova versão do sistema utilizando-se de baterias recarregáveis para seu funcionamento. Isso poderia permitir o uso da ferramenta em regiões isoladas do país onde não há infraestrutura elétrica.

A adição de novos modelos de testes rápidos para o leque de exames que a leitora é capaz de atender deve ser uma melhoria que ocorra de forma orgânica, sendo realizada conforme a demanda. Essa evolução não deve encontrar grandes dificuldades, com o projeto já tendo sido feito com isso em mente. Surgindo a necessidade, um novo banco de imagens deverá ser usado para o treinamento e com pequenas alterações no código fonte a rede neural pode ser novamente treinada com esse banco de imagens expandido.

As distorções geradas nas imagens devido a limitação de lentes e espaço dentro da estrutura da leitora pode ser corrigida futuramente de maneira digital utilizando o OpenCV. A biblioteca possui ferramentas para detectar e compensar essas distorções causadas por projetas uma cena 3D em uma imagem 2D.

REFERÊNCIAS

AJIT, A.; ACHARYA, K; SAMANTA, **A. A Review of Convolutional Neural Networks.** 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1-5, doi: 10.1109/ic-ETITE47903.2020.049. Disponível em: https://ieeexplore.ieee.org/abstract/document/9077735.

ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. **Understanding of a convolutional neural network.** International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186. Disponível em: https://ieeexplore.ieee.org/abstract/document/8308186.

ASIMOV, I. **The Robots of Dawn**. 1. ed. New York, NY: Doubleday, 1983.

AXXIN BIOTECH. **AX-2X-S LATERAL FLOW READER.** Disponível em: https://www.axxin.com/LateralFlow-AX-2XS.php. Acesso em: 21 mar. 2023.

BURGER,W.;BURGE,M.J. **Principles of Digital Image Processing:** Fundamental Techniques. 1. ed. Berlin, Alemanha: Springer, 2009.

CARVALHO, A.P.L.F. **Rede Neurais Artificiais.** São Paulo: Universidade de São Paulo. Disponível em: https://sites.icmc.usp.br/andre/research/neural/. Acesso em: 18 dez. 2022.

DI NARDO, F.; CHIARELLO, M.; CAVALERA, S.; BAGGIANI, C.; ANFOSSI, L. **Ten Years of Lateral Flow Immunoassay Technique Applications:** Trends, Challenges and Future Perspectives. Sensors 2021, 21, 5185. Disponível em: https://doi.org/10.3390/s21155185.

FRANCESCHI, J; FAWZI, A; FAWZI, O. **Robustness of classifiers to uniform ℓp and Gaussian noise**. ArXiv abs/1802.07971, 2018. Disponível em: https://www.semanticscholar.org/paper/Robustness-of-classifiers-to-uniform-%F0%9D%93%81p-and-noise-Franceschi-Fawzi/262f8776fdc3c32de103d624ac36943282f310c6.

GÉRON, A. **Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow:** Concepts, Tools, and Techniques to Build Intelligent System. 2. ed. Sebastopol, USA: O'REILLY, 2019.

GONZALEZ, R. C.; WOODS, R. E.; EDDINS, S. L. **Processamento de Imagens Digitais Usando o MATLAB.** 2. ed. São Paulo: Pearson Education do Brasil, 2009.

HSIAO, W. W.-W.; LE, T.-N.; PHAM, D. M.; KO, H.-H.; CHANG, H.-C.; LEE, C.-C.; SHARMA, N.; LEE, C.-K.; CHIANG, W.-H. **Recent Advances in Novel Lateral Flow Technologies for Detection of COVID-19.** Biosensors 11, no. 9: 295, 2021. Disponível em: https://doi.org/10.3390/bios11090295.

IBM CORPORATION. **O que é Computer Vision.** Disponível em: https://www.ibm.com/br-pt/topics/computer-vision. Acesso em: 21 nov. 2022.

LCD WIKI. **5inch HDMI Display.** Disponível em: http://www.lcdwiki.com/5inch_HDMI_Display. Acesso em: 23 nov. 2022.

LOVELL, D.R.; MILLER, D.; CAPRA, J.; BRADLEY, A. **Never mind the metrics - what about the uncertainty? Visualising confusion matrix metric distributions**. (2022). ArXiv, abs/2206.02157. Disponível em: https://www.semanticscholar.org/paper/Never-mind-the-metrics-what-about-the-uncertainty-Lovell-Miller/c2b86d79eeb5ee9d062c639047fd88c99e5223df.

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, doi: 10.1109/TNNLS.2021.3084827. 2021. Disponível em: https://ieeexplore.ieee.org/abstract/document/9451544.

LOVE, D. **Tkinter GUI Programming by Example.** 1. ed. Birmingham, UK: Packt Publishing, 2018.

PARK, J. **Lateral Flow Immunoassay Reader Technologies for Quantitative Point-of-Care Testing.** Sensors (Basel). 2022 Sep 28;22(19):7398. doi: 10.3390/s22197398. PMID: 36236497; PMCID: PMC9571991. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9571991/.

PARKER, J.R. **Algorithms for Image Processing and Computer Vision.** 2. ed. Hoboken, USA: Wiley, 2011.

PAZ, A. R. de. **Tkinter GUI Application Development Cookbook**. 1. ed. Birmingham, UK: Packt Publishing, 2018.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning:** Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow. 2. ed. Birmingham, UK: Packt Publishing, 2017.

RASPBERRY PI FOUNDATION. **Raspberry pi model B plus product brief.** Disponível em: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#raspberry-pi-3-model-b. Acesso em: 27 dez. 2022.

SAS Inc. **Visão Computacional. O que é e qual sua importância?** Disponível em: https://www.sas.com/pt_br/insights/analytics/computer-vision.html. Acesso em: 21 nov. 2022.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. 1. ed. Berlin, Alemanha: Springer, 2011.

TensorFlow. **Guia do TensorFlow Lite.** Disponível em: https://www.tensorflow.org/lite/guide?hl=pt-br. Acesso em: 21 jan. 2023.

TWOMEY, J.M.; SMITH, A.E. **Performance measures, consistency, and power for artificial neural network models.** Mathematical and Computer Modelling, v. 21, n. 1-2, p. 243-258, 1995. ISSN 0895-7177. Disponível em: https://doi.org/10.1016/0895-7177(94)00207-5.

VILLÁN, A. F. **Mastering OpenCV 4 with Python:** A Practical Guide Covering Topics from Image Processing, Augmented Reality to Deep Learning with OpenCV 4 and Python 3.7 . 1. ed. Birmingham, UK: Packt Publishing, 2019.