CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

ETEC DA ZONA LESTE

Mtec Desenvolvimento de Sistemas AMS

Enzo Costa Paz

Felipe Vieira de Oliveira

Gustavo de Souza Morais

Sakiri Moon Payao Cestari

AGRO-G.E.S.F: Sistema Inteligente de Monitoramento e Prevenção de Pragas em Cultivos em Linha

São Paulo

2025

Enzo Costa Paz

Felipe Vieira de Oliveira

Gustavo de Souza Morais

Sakhir Moon Payao Cestari

AGRO-G.E.S.F: Sistema Inteligente de Monitoramento e Prevenção de Pragas em Cultivos em Linha

Trabalho de conclusão de curso apresentado ao Curso Técnico em Desenvolvimento de Sistemas da ETEC Zona Leste, orientado pelo Prof. Esp. Jeferson Roberto de Lima, como requisito parcial para obtenção do título de técnico em Desenvolvimento de Sistemas.

AGRO-G.E.S.F

Sistema Inteligente de Monitoramento e Prevenção de Pragas em Cultivos em Linha

Enzo Costa Paz

Felipe Vieira de Oliveira

Gustavo de Souza Morais

Sakhir Moon Payao Cestari

Aprovada em __/__/____.

BANCA EXAMINADORA

Prof. Esp. Jeferson Roberto de Lima

Universidade do Jeferson

Prof. (Professor avaliador) Universidade do Avaliador Prof. (Professor avaliador) Universidade do Avaliador

São Paulo

2025

“Frase”

Autor

RESUMO

O atual trabalho propõe se ao desenvolvimento de um sistema embarcado para o monitoramento de possíveis sinais de pragas em plantações no formato em linha. O sistema contará com uma câmera, um comunicador LoRa e seu receptor, para a transmissão e recepção dos dados. Bem como, a utilização de uma SBC (Single Board Computer) para o processamento local da imagem, e um sensor básico para captura de informações do solo. O projeto busca viabilizar um meio acessível para pequenos produtores, conseguirem monitorar suas lavouras de forma mais simplificada e eficiente

Palavras-chave: Python; Visão computacional; Machine Learning; LoRa; Praga.

ABSTRACT

The current work proposes the development of an embedded system for monitoring possible signs of pests in plantations in an online format. The system will have a camera, a LoRa communicator and its receiver, for transmitting and receiving data. As well as the use of an SBC (Single Board Computer) for local image processing, and a basic sensor for capturing ground information. The project seeks to provide an accessible means for small producers to monitor their crops in a more simplified and efficient way.

Keywords: Python; Computer vision; Machine Learning; LoRa; Prague.

LISTA DE ILUSTRAÇÕES

Figura 1 - Logitech Webcam Brio 100 18
Figura 2 - Exemplo de código Python para soma 19
Figura 3 - Resultado do código em Python 20
Figura 4 - Exemplo de utilização do PIP 21
Figura 5- Exemplo de interface com Tkinter 22
Figura 6 - Resultado do código TKinter 23
Figura 7 - Exemplo de código com CustomTkinter 24
Figura 8 - Resultado do código com Custom Tkinter 25
Figura 9 - Exemplo de código usando Matplotlib 26
Figura 10 - Resultado do código utilizando Matplotlib 27
Figura 11 - Código de Data Augmentation 29
Figura 12 – Exemplo de Data Augmentation 30
Figura 13 - Exemplo de código de CNN 32
Figura 14 - Exemplo de código utilizando OpenCV 35
Figura 15 - Imagem antes do processamento usando OpenCV 36
Figura 16 - Aplicação do filtro binário na imagem 37
Figura 17 - Resultado do processamento da imagem 38
Figura 18 - Exemplo de código utilizando PyTorch 39
Figura 19 - Resultado da análise de imagem 40
Figura 20 - Exemplo de código utilizando TorchVision 41
Figura 21 - Exibição da imagem classificada com TorchVision 43
Figura 22 - Resultado do código utilizando TorchVision 43
Figura 23 - Exemplo de Wireframe de baixa fidelidade 45
Figura 24 - Exemplo de Wireframe de alta fidelidade 46
Figura 25 - Exemplo de um Caso de Uso 47
Figura 26 - Exemplo de um Diagrama de Classe 49

LISTA DE ABREVIATURAS E SIGLAS

CNN - Convolutional Neural Network

UI – User Interface

UX – User Experience

UML - Unified Modeling Language

Lista de tabelas

LISTA DE SÍMBOLOS

Sumário

1 INTRODUÇÃO 15
2 REFERENCIAL TEORICO 17
2.1 Pragas nas plantações 17
2.2 Internet das coisas (IoT) 17
2.4 Raspberry Pi 4 17
2.3 LoRa 18
2.4 Câmera 18
2.5 Python 19
2.5.2 Tkinter 21
2.5.3 Customtkinter 23
2.5.4 Matplot lib 25
2.5.5 NumPy 27
2.5.6 Pillow 27
2.6 Machine learning 28
2.6.1 Data Augmentation 28
2.6.2 Deep Learning 31
2.6.3 Convolutional Neural Network 31
2.7 Visão computacional 33
2.7.1 Open-Source Computer Vision Library (OpenCV) 34
2.8 PyTorch 38
2.8.1 TorchVision 41
2.9 Docker 43
2.10 Banco de dados 43
2.11 Wireframes 44
2.11.1 Wireframe de baixa fidelidade 44
2.11.3 UX (User Experience) 46
2.11.4 UI (User Interface) 46
2.12 UML 46
3 DESENVOLVIMENTO 50
4 CONSIDERAÇÕES FINAIS 50

1 INTRODUÇÃO

O Sistema Inteligente de Monitoramento e Prevenção de Pragas em Cultivo de Linha é um dispositivo capaz de percorrer uma lavoura de plantas rasteiras em todos os cenários possíveis, identificando possíveis sinais de pragas e doenças nas folhas. Algumas das pragas e doenças identificáveis são: Vaquinha-Verde-Amarela, Requeima e Pinta-Preta. Ao identificar os sinais, passa as informações para o nosso dashboard em uma aplicação desktop via cabo USB tipo C.O objetivo é identificar de forma precoce, em cultivos de formação em linha, possíveis sinais de doenças e pragas, e permitir que o pequeno agricultor tenha uma forma de fazer o monitoramento de sua lavoura, como também, tem uma noção do histórico das pragas e doenças em suas plantações.

O principal problema é que o agricultor acaba por muitas vezes, fazendo a identificação de forma tardia da doença ou da praga, o que acaba resultando em uma perca elevada da cultura. Em complemente, isto acaba por exigir do agricultor, a aplicação de agrotóxicos de forma mais densa e agressiva, o que põe em risco os trabalhadores que o aplicam, a própria plantação e o consumidor final, a partir do momento que, ao aplicar-se com determinada ocorrência e força tais agrotóxicos, aumenta a possibilidade de erros na ajustar o equipamento de dispersão, a praga e doença presente podem consequentemente gerar uma resistência maior ao agrotóxico, dificultando futuras passagens do químico e aumentando a presença do mesmo no produto final para o consumidor.

Inicialmente, acredita-se que a principal causa para tais atos, é a falta de educação sobre o uso dos agrotóxicos nas lavouras, mais, a cultura presente em nosso país da aplicação tardia no campo e próxima a colheita, para extinguir qualquer praga e doença próxima a colheita e tentar extrair o máximo possível do produto plantado.

A sua importância se faz presente partir do momento em que se propõe a diminuir o uso de agrotóxico no meio agrícola por meio da identificação precoce das possíveis pragas e doenças, trazendo assim para o meio agrícola de menor porte, um meio para proteger suas lavouras e diminuir seus gastos. Em complemento, tal projeto, permite que o pequeno agricultor possa fazer parte do crescimento da agricultura de precisão no Brasil.

A abordagem desta pesquisa é de caráter qualitativo, pois baseia-se na análise de informações oriundas de artigos e autores que abordam a temática (Lakatos, Marconi, 2017) da presença de pragas e doenças dentro de lavouras, em destaque, as consequências oriundas de tal fato e a eficácia da utilização da linguagem Python neste meio.

O sistema se delimita inicialmente, a produtores de pequeno porte, principalmente, os quais possuam plantações de porte rasteiro e não possuam a capacidade de se equiparar a grandes produtores com alto poder aquisitivo.

A presente pesquisa tem como principais autores MARTINS; FONTES; FORNAZIER, 2013; MARTINS; FORNAZIER; FANTON, 2013, para falar sobre manejo de pragas e a ocorrência que tais possuem no meio agrícola, BRITO;GOMIDE e CÂMARA para falar sobre intoxicação dos trabalhadores, os quais discorrem sobre a importância do manejo e cuidado na área onde foi dispersado o químico, para abordar o assunto da ausência de formação dos agricultores sobre o uso indiscriminado de agrotóxicos e uso inadequado, utilizaremos os autores .....; para desenvolver tal meio para o monitoramento da lavoura, utilizaremos a linguagem de programação Python, que é destacada por MENENZES, para poder o aprendizado de máquina para possuir uma forma de identificar as pragas e doenças, se utilizará o PyTorch como aponta Jeronimo.

O trabalho está dividido em 4 capítulos: o 1º fala sobre a importância do uso correto dos agrotóxicos, bem como sobre o manejo adequado; o 2º, é o Referencial teórico; o 3º capítulo consta do “Desenvolvimento”, no qual será realizada a análise e discussão dos resultados; e o 4º, em que faremos as considerações apontando as limitações e possíveis melhorias no futuro.

2 REFERENCIAL TEORICO

Na atual seção são apresentadas os principais conceitos e tecnologias para a construção da base teórica para o desenvolvimento do Agro-G.E.S.F.

2.1 Pragas nas plantações

Pragas, conforme estabelece a Embrapa (EMBRAPA,2024) são qualquer espécie, raça ou biotipo de um animal ou agente patogênico que venha a causar algum tipo de dano a plantas ou culturas. Em auxílio, Senar (2017) complementa que pragas são usualmente caracterizadas como ácaros, cochonilhas e nematoides, enquanto fungos, vírus e bactérias são usualmente coligados a doenças. Complementarmente, Martins, Fontes e Fornazier (2013 apud FERRÃO et al.,2017) informam que as pragas podem ocorrer de forma esporádica ou sistemática. Aquelas que ocorrem de forma esporádica e causam danos significativos, são conhecidas como pragas secundárias, já aquelas que ocorrem de forma sistemática toda vez que se implementa a cultura na lavoura e causam danos quantitativos/qualitativos, são conhecidas como pragas-chaves.

2.2 Internet das coisas (IoT)

Oliveira (2017) afirma que a internet das coisas (IoT) é um conceito que está se tornando cada vez mais presente na vida cotidiana. Nas últimas duas décadas, a IoT tem auxiliado a vivência humana em diversas áreas por meio da automação de tarefas, ou em objetos como crachás eletrônicos e veículos. A IoT consiste em qualquer objeto capaz de manipular dados em uma rede, visando auxiliar e simplificar ações complexas (MAGRANI, 2018).

2.4 Raspberry Pi 4

Conforme Dobbin (2022) indica, o Raspberry pi é um computador de placa única (SBC), montado em uma placa de circuito impresso (PCB). Se caracteriza por poder realizar as mesmas funções de um computador doméstico comum, como acessar a internet, pesquisar sites e assistir vídeos, tendo somente o adendo de levar tempo levemente maior para realizar tais funções devido a seu tamanho.

Como apontam Santos, Lopes e Dias (2024), o Raspberry pi 4 apresenta um nível de processamento mediano devido a seu processador de maior capacidade em atividades multitarefas.

Em complemento Lauxen(2023) afirma que o Raspberry pi 4 pode ser alocado dentro de um sistema como o cérebro do projeto, de forma que permita fazer a captura de imagens e controle de elementos ligados a ele.

2.3 LoRa

Tal como descrito por Pastório et al. (2021), o LoRa é uma tecnologia que transmite dados por frequências de rádio a longas distâncias com baixo consumo de energia. Essa tecnologia é amplamente utilizada em IoT, principalmente em projetos que necessitam de um baixo custo energético.

2.4 Câmera

A Webcam Brio 100 consegue capturar imagens em Full HD com a resolução 1920x1080 pixels. Ela possui equilíbrio automático de luz, aumentando o brilho em até 50% e reduzindo sombras quando necessário (LOGITECH, 2023).

Figura 1 - Logitech Webcam Brio 100

Fonte: (Logitech,2025)

2.5 Python

Segundo Luiz Eduardo Borges (2014), a linguagem Python foi criada em 1990 por Guido van Rossum, no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda (CWI), tendo como foco, o auxílio para profissionais como físicos e engenheiros.

Em conformidade a Paiva *et al.* (2020), Python é uma linguagem com características interessantes e de simples aprendizado. Inicialmente, possuía a meta inicial de ser uma linguagem de código mais enxuto e menos verboso. Em respaldo à está programação simples, Python permite o uso de diversos módulos de extensão, que fornecem ao Python, uma forte presença e poder.

Consoante a Menezes (2014), Python vem se tornando uma forte linguagem de programação em diversas áreas da computação, como inteligência artificial, banco de dados, biotecnologia e até mesmo em aplicações web. Em suporte, Python é uma linguagem conhecida por ter “batteries included”, ou seja, possui baterias inclusas, uma linguagem de programação pronta para ser usada. Em auxílio a tudo isto, Python é uma linguagem de programação para se obter resultados em pouco tempo, devido a sua simples complexidade e alta densidade de módulos para importação.

Figura 2 - Exemplo de código Python para soma

```python
"""
Retorno de valores das funções (return)
"""

def soma(x,y):

    resultadosoma = x+y
    return resultadosoma


resultadochamada = soma(1,2)
print(resultadochamada)
```

Fonte: Autoria própria,2025

O exemplo de código acima, é um código em Python que serve para fazer a soma de dois números, estes quais são inseridos pelo próprio usuário previamente dentro do próprio código. Abaixo, segue uma breve explicação sobre o código e seu funcionamento:

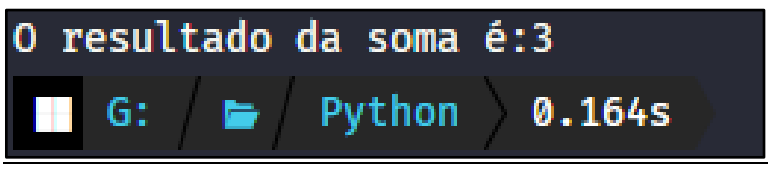
Linha 1 a 3: Cria uma “Docstring” (tipo de string), no qual no contexto atual, serve como uma forma de criar comentários, neste caso, o Python irá passar por estas linhas e irá ignorá-las;

Linha 5: Cria a função “soma”, está qual serve para modularizar o código e o deixar mais prático, permitindo que o programador não precise ficar repetindo as mesmas linhas de código, e, chamando a função somente quando necessária. Nesta aplicação, a função soma recebe dois parâmetros, dois itens, e os salva em variáveis (variáveis são formas de salvar valores específicos dentro da memória do computador) denominadas como “x” e “y”;

Linha 7 a 8: Na linha 7, é criada uma variável para receber o valor da soma das duas variáveis declaradas inicialmente, logo em seguida na linha 8, o valor dessa variável é devolvido ao meio que lhe chamou;

Linha 11 a 12: É criada uma variável chamada “resultadochamada”, à qual chama a função soma, enviando como parâmetros os números 1 e 2, ao mesmo tempo que recebe o valor do retorno dado pela função. Na linha 12, por meio da função “print()”, é exibido ao usuário por meio do prompt de comando integrado da IDE, o valor da variável “resultadochamada”.

Figura 3 - Resultado do código em Python



Fonte: Autoria Própria (2025)

Como consequência da execução do código antes ilustrado, na imagem acima há o retorno do código Python, no caso, foram passados o número um e dois, e o retorno da soma de tais números foi o número três.

2.5.1 PIP

Em conforme com a documentação oficial do Python (2025), o PIP é o administrador de pacotes, sendo utilizado majoritariamente e objetivamente para a instalação e desinstalação de pacotes.

No exemplo abaixo, há o funcionamento do comando PIP, sendo utilizado para instalar a biblioteca pytorch.

Figura 4 - Exemplo de utilização do PIP

Fonte: Autoria Própria,2025

2.5.2 Tkinter

Segundo Labaki (2011) o Tkinter, é uma biblioteca padrão que vem junto da linguagem Python, essa biblioteca oferece um kit de ferramentas GUI, ou *Graphical User Interface,* que se traduz para Interface Gráfica de Usuário, as interfaces gráficas de usuários são as áreas de interação dentro de um *software*, dentro dessas interfaces temos os *widget* que é usado para referenciar componentes dentro de GUIs, sendo elas, um botão, um menu ou um caixa de texto.

Continuando com Menezes (2024), o Tkinter é baseado em uma biblioteca gráfica chamada TK, que foi desenvolvida para uma linguagem chamada TCL. A junção de TCL/TK foi muito utilizada em sistemas operacionais e muito popular quando Python foi criado, com o passar dos anos, essa biblioteca foi modernizada e teve os mesmos controles com aparência nativa.

Para Reitz e Schlusser (2016) o Tkinter é uma das bibliotecas mais convenientes e compatíveis dentre todas que existem por já estarem junto ao Python quando instalado e está disponível na maioria dos sistemas operacionais, como Windows, Unix e macOS X. Segue abaixo um exemplo de uma interface gráfica usando Tkinter.

Figura 5- Exemplo de interface com Tkinter

Fonte: Autoria Própria, 2025

Este, é um exemplo de uma interface simples utilizando Tkinter, que exibe uma tela que pede para digitar o seu nome em uma caixa de texto, ao digitar, abaixo haverá um botão dizendo “Dizer Olá”, ao clicar, aparecerá um texto abaixo do botão que exibirá um cumprimento ao usuário e dizendo o nome que foi entregue na caixa de texto. Explicamos mais sobre o código abaixo:

Linha 1: é feita a importação da biblioteca Tkinter e estamos definindo um nome a essa importação para um melhor uso no futuro.

Linha 3 a 5: Definimos a função junto a um nome a ela que será executada quando o botão ser pressionado e exibirá o nome do usuário junto a um rótulo.

Linha 8 a 10: Criamos a tela principal que será exibida e damos um nome junto de um tamanho para essa tela que será de 300 pixels por 150 pixels de altura.

Linha 12 a 13: É criado um rótulo com uma mensagem pedindo para digitar o nome do usuário e exibimos na tela junto de um espaçamento vertical.

Linha 15 a 16: É definido um campo de texto para o usuário digitar seu nome e é exibido o campo na janela com um espaçamento.

Linha 18 a 19: É adicionado o botão com um texto, ao clicar no botão a função que definimos no início que exibe o nome do usuário junto com uma saudação

Linha 21 a 22: É necessário criarmos um rótulo que não terá texto para podermos mostrar a saudação

Linha 24: É iniciado o loop da aplicação, essa linha mantém a janela em funcionamento esperando pelas ações do usuário.

Figura 6 - Resultado do código Tkinter

Fonte: Autoria Própria, 2025

A imagem acima é o resultado do código explicado sendo exibida em uma tela de computador utilizando Tkinter.

2.5.3 Customtkinter

Segundo Frota,Ruver e Figueiredo (2024), o CustomTkinter é uma biblioteca de criação de interfaces gráficas inspirada no Tkinter, a biblioteca apresenta um visual mais moderno e customizável quanto o Tkinter, mas mantendo a simplicidade da biblioteca inspirada, o CustomTkinter é ótimo para aprendizes e especialistas que queiram criar interfaces elegantes sem uma dificuldade exagerada.

O código abaixo se utiliza do exemplo que foi apresentado no capítulo do Tkinter, neste exemplo vamos implementar a estilização do CustomTkinter.

Figura 7 - Exemplo de código com CustomTkinter

```python
import customtkinter as ctk

ctk.set_appearance_mode("System")
ctk.set_default_color_theme("blue")

def dizer_ola():
    nome = entrada.get()
    mensagem = f"Olá, {nome}!"
    rotulo_resultado.configure(text=mensagem)

# Janela principal
janela = ctk.CTk()
janela.title("Exemplo CustomTkinter")
janela.geometry("300x180")

# Widgets
rotulo = ctk.CTkLabel(janela, text="Digite seu nome:")
rotulo.pack(pady=5)

entrada = ctk.CTkEntry(janela)
entrada.pack(pady=5)

botao = ctk.CTkButton(janela, text="Dizer Olá", command=dizer_ola)
botao.pack(pady=5)

rotulo_resultado = ctk.CTkLabel(janela, text="")
rotulo_resultado.pack(pady=5)

# Loop da interface
janela.mainloop()
```

Fonte: Autoria Própria, 2025

Este é o mesmo exemplo que utilizamos no Tkinter, porém agora escrito utilizando o CustomTkinter, o que será exibido é o mesmo, uma interface que irá exibir uma mensagem pedindo para inserir seu nome em uma caixa de texto, ao inserir seu nome e clicar no botão abaixo da entrada da área de texto, será exibido um cumprimento ao usuário e dizendo o nome que foi apresentado no campo, abaixo segue o que está acontecendo no código:

Linha 1: é feito a importação da biblioteca CustomTkinter e definindo um nome para essa importação e ter um melhor uso.

Linha 2 a 3: Definimos qual será o modo visual dependendo do sistema operacional do usuário.

Linha 6 a 9: Criamos a função que irá guardar o texto que digitado no campo e criamos um rótulo que irá apresentar o cumprimento e o nome digitado.

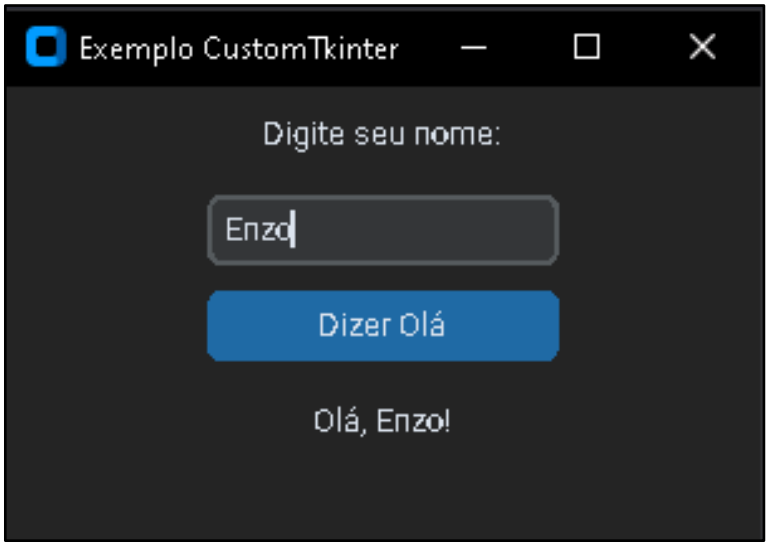
Linha 11 a 14: adicionamos a janela principal que conterá um nome e um tamanho específico.

Linha 17 a 21: Criamos um rótulo que está pedindo ao usuário para digitar o seu nome e criamos o campo de texto para o usuário poder digitar, é definido um espaçamento vertical para ambos os elementos.

Linha 23 a 27: Criamos o botão junto com um texto dentro que ao ser clicado irá chamar a função que exibirá a saudação e criamos um rótulo sem texto para uma melhor apresentação final, e novamente definimos um espaçamento vertical para os elementos.

Linha 30: é feito o loop da estrutura, essa linha permite que a tela seja exibida enquanto o código estiver ativo.

Figura 8 - Resultado do código com Custom Tkinter



Fonte: Autoria Própria, 2025

A imagem que está acima exibe o código em funcionamento e vemos que ele está personalizado com as funcionalidades do CustomTkinter.

2.5.4 Matplot lib

Desenvolvido em 2002 por John Hunter, o Matplotlib é uma biblioteca para Python para esboçar gráficos de todos os tipos. O Matplotlib consegue suportar diversos banco de dados de interfaces gráficas, pode funcionar em vários sistemas operacionais, além de poder exportar suas figuras em vários formatos de arquivos possíveis. (McKinney, 2022).

Consoante a Grus (2019), o Matplotlib oferece uma grande variedade de gráficos, sendo eles de níveis simples até níveis mais complexos de figuras, esses mapas podem ser interativos e com modelos otimizados podendo ser personalizado da forma que seja desejado.

Para Vasiliev (2022), o Matplotlib é uma das principais bibliotecas python quando o assunto é sobre criar gráficos e figuras, o Matplotlib é responsável por criar diversas outras bibliotecas também responsáveis por criar gráficos que se baseiam na estrutura do Matplotlib Vamos mostrar abaixo um exemplo de código que exibe um gráfico utilizando Matplotlib:

Figura 9 - Exemplo de código usando Matplotlib

Fonte: Autoria Própria, 2025

Neste código, será exibido uma imagem de um gráfico com valores específicos junto de um título para o gráfico, vamos explicar melhor sobre o código abaixo:

Linha 1: Faz a importação da biblioteca matplotlib e damos um nome a essa biblioteca.

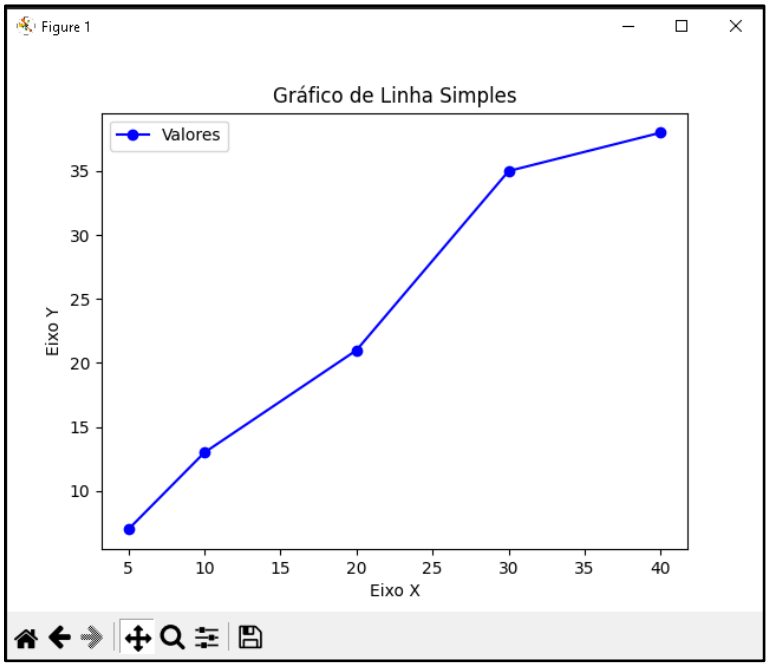
Linha 4 a 5: Definimos os valores do eixo X e os valores do eixo Y, cada valor que está sendo exibido é coordenada dentro do gráfico.

Linha 8: Cria o gráfico junto com os eixos que definimos nas linhas 4 e 5, adicionamos algumas estilizações como o tipo de marcador, qual o formato da linha e a sua cor e por fim criamos um rótulo que irá nomear a linha em específico.

Linha 11 a 14: adicionamos um título para o gráfico, um título para o eixo X e o eixo Y e colocamos um comando para exibir a legenda ao passar sobre a linha.

Linha 17: Exibe o gráfico em uma janela quando o código ser executado

Figura 10 - Resultado do código utilizando Matplotlib



Fonte: Autoria Própria, 2025

Como podemos ver na imagem acima, está sendo exibido o gráfico conforme definimos no código e utilizando o Matplotlib.

2.5.5 NumPy

Conforme aponta Lins e Souza (2023) o pacote NumPy (Numerical Python) é essencial para a computação cientifica. Em consonância com Grus (2021) o NumPy se torna viável e interessante devido a presença de seu tipo de arrays, que possuem um desempenho superior ao se comparar as listas, vetores e matrizes comuns do Python.

2.5.6 Pillow

Pillow, assim como afirma De Paula (2024) é uma biblioteca popular do Python que tem como objetivo a manipulação de imagens. Em apoio ao que foi escrito, Moreira, Marmontel, Chamorro (2023) explicam que a biblioteca Pillow é uma evolução da biblioteca PIL do próprio python, ademais, uma das principais vantagens do Pillow é sua simplicidade, o que permite o uso de tal tanto por iniciantes quanto por programadores avançados, em complemento, ela também se destaca por ser

otimizada para o trabalho em imagens de alta resolução, o que a torna útil em aplicações com a necessidade de processamento em lote.

2.6 Machine learning

O aprendizado de máquina, ou mais comumente conhecido como “Machine Learning”, é, segundo Paixão et al. (2022), uma subárea da ciência da computação que trabalha com a união de técnica matemáticas e estatísticas aplicadas à algoritmos computacionais. Parafraseando Escovedo e Koshiyama (2020), o conceito de Machine Learning se foca na descoberta de padrões por meio dessa união de técnicas, independentemente do grau de utilidade de tais descobertas, ademais, a utilização de tais padrões para a automatização de determinadas tarefas, que antes, se poderiam se mostrar como difíceis para serem realizadas por humanos.

Quando se trata do aprendizado de máquina, se prova por meio de Ludemir (2021) que é necessário um grande volume de dados para que o algoritmo possa aprender de forma automática. Em auxílio, a acurácia de um modelo treinado pode variar conforme a quantidade de dados fornecidos ao modelo, juntamente a qualidade de tais dados, como diz Tsunoda *et al.* (2020). Em acréscimo, destaca-se o fato de que os modelos treinados têm se destacado dentro do cenário da agricultura de precisão, estes quais são sustentados por grandes volumes de dados e poder de processamento, como também, são amplamente utilizados para a detecção de doenças e ervas daninhas em culturas, previsão de produtividade e gestão da saúde do solo, conforme demonstra Santos (2022).

2.6.1 Data Augmentation

Perante a Brownlee et al (), a técnica de Data Augmentation (Aumento de Dados) tem como intuito a criação de dados artificiais, para a alimentação de dados para algoritmos de aprendizado de máquina a partir de dados já existentes, como, a aplicação de mudança de ângulos da imagem, como a diminuição ou aumento de brilho e de saturação. Tal técnica tem uma devida importância como informa Shorten et al (2021, apud Oliveira et al.[2024]), por poder auxiliar a minimizar o impacto da ausência de dados para determinados treinamentos de máquina.

O código a seguir, mostra que, por meio de determinadas configurações, se é possível aumentar de forma artificial a quantidade de imagens:

Figura 11 - Código de Data Augmentation

```python
import os
import matplotlib.pyplot as plt
from PIL import Image
import torch
from torchvision import transforms

transform_aug = transforms.Compose([
    transforms.RandomResizedCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406],
                         [0.229, 0.224, 0.225]),
])

transform_original = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])
```

Fonte: Autoria própria,2025

Linha 1 a 5: Ocorre a importação das bibliotecas necessárias para o funcionamento do código;

Linha 7: É definida uma variável que irá guardar uma sequência de transformações para aplicar a técnica de Data Augmentation nas imagens;

Linha 8: Corta aleatoriamente uma região da imagem e redimensiona para 224x224 pixels. Desta forma, o modelo não fica preso à posição ou escala fixa da imagem original;

Linha 9: Aplica um espelhamento horizontal na imagem com probabilidade de 50%, o que ajuda o modelo a generalizar imagens invertidas;

Linha 10: Modifica de forma aleatória o brilho, contraste, saturação e matiz(cor) da imagem dentro dos parâmetros dados, tornando o modelo menos sensível a variações de iluminação e cor;

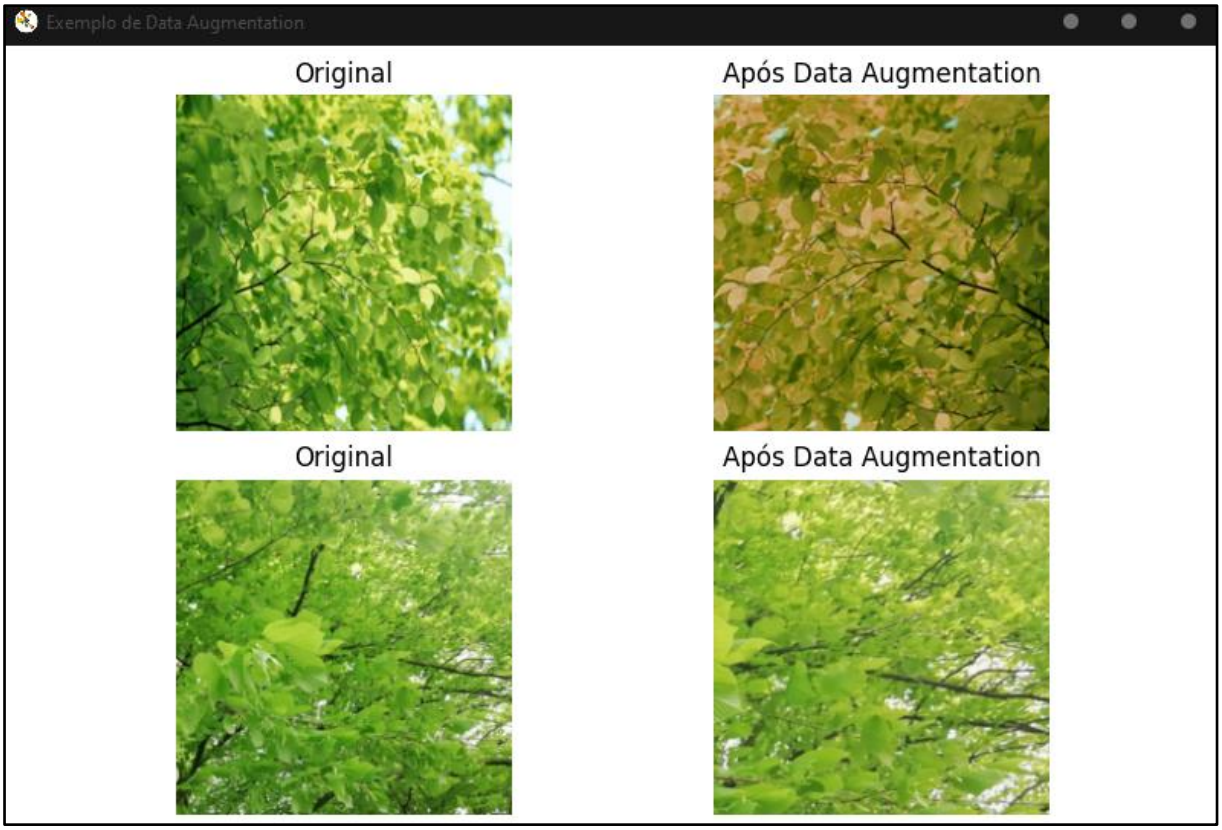
Linha 11: Transforma a imagem, para um tensor PyTorch, normalizando os valores dentro da faixa de 0 e 1;

Linha 12 e 13: Normaliza o tensor usando uma média e desvio padrão, centralizando os dados para facilitar o aprendizado de máquina;

Linha 14: É fechada a chave e os parênteses da variável, assim passando para a variável inicial, todos parâmetros que ela deve guardar;

Como resultado, a imagem abaixo demonstra o resultado após aplicar tais filtros a duas imagens de folhas de plantas.

Figura 12 – Exemplo de Data Augmentation



Fonte: Autoria própria, 2025

2.6.2 Deep Learning

Parafraseando Hosaki (2021), o Deep Learning ou aprendizado profundo, é um conjunto de técnicas de Machine Learning que se utilizam de redes neurais profundas, estas quais tiveram sua utilização inicial no ano de 2006, e atualmente estas redes neurais são compostas por diversos níveis, onde cada nível possui seu nível de abstração. O Deep Learning possui em sua essência uma similaridade com a rede neural do ser humano, o que permite uma análise de informações sem a necessidade de um processamento prévio ou a necessidade de um supervisionamento.

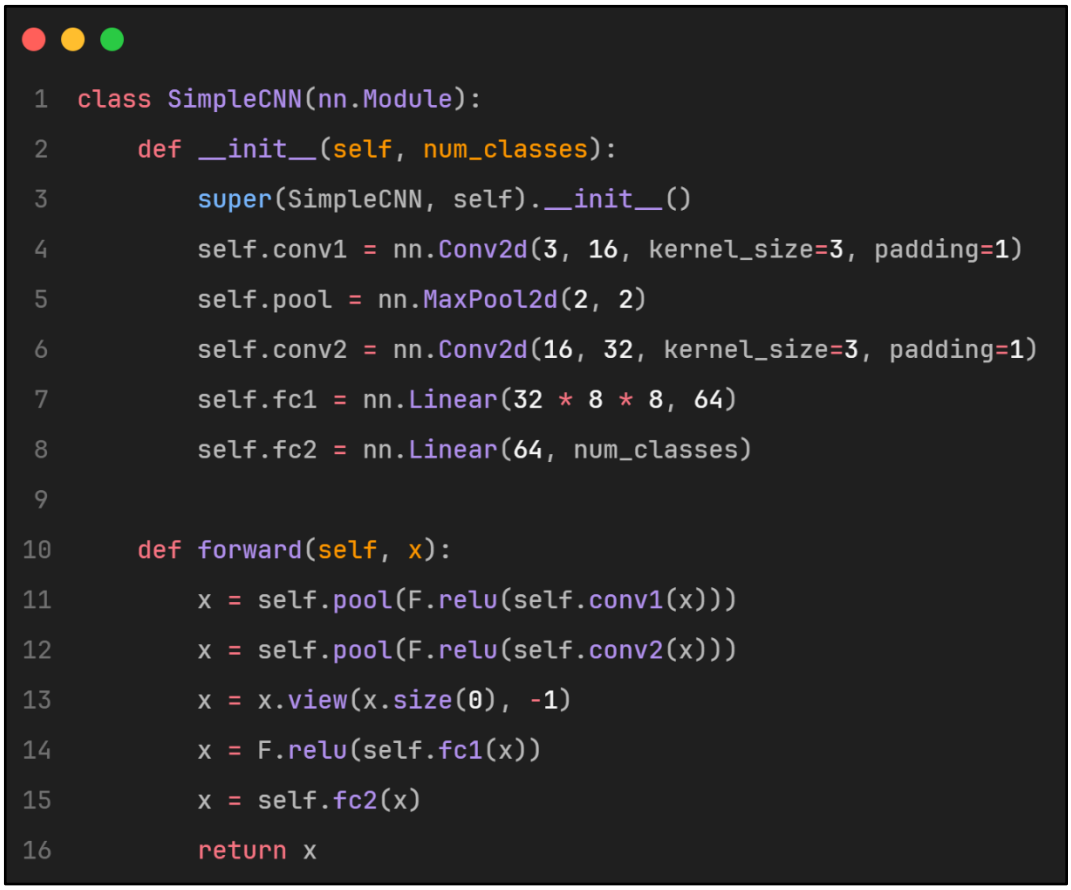
2.6.3 Convolutional Neural Network

Uma Convolutional Neural Network (CNN), em português Rede Neural Convolucional, é perante a Carneiro et al (2025) um tipo de rede neural artificial que é usualmente empregada para a classificação de padrões bidimensionais, e é composta por três camada, onde cada uma fica responsável por funções matemáticas especificas, para que no final, haja uma precisão maior.

Em complemento, Vargas, Paes e Vasconcelos (2016), este tipo de rede neural vem sendo amplamente utilizado em áreas como detecção, classificação de reconhecimento de imagens e vídeos.

A imagem abaixo demonstra o código de criação de uma CNN básica, possuindo apenas duas camadas.

Figura 13 - Exemplo de código de CNN



```python
class SimpleCNN(nn.Module):
    def __init__(self, num_classes):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(32 * 8 * 8, 64)
        self.fc2 = nn.Linear(64, num_classes)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Fonte: Autoria própria, 2025

Logo abaixo, há a explicação prévia do que cada linha corresponde dentro da CNN:

Linha 1: Se declara o nome da classe da rede neural, que é “SimpleCNN”. Similar a uma receita, é ela que vai descrever o que é necessário e quais os passos para fazer a rede. O “nn.Module” é uma ferramenta do PyTorch que ajuda a organizar as camadas da rede;

Linha 2: É onde se inicia a configuração da rede, e o num_classes, é o número que corresponde a quantos tipos de itens a rede pode identificar;

Linha 3: Mais um detalhe técnico, mas que basicamente “conecta” a rede ao PyTorch, dizendo que ela segue as regras de um nn.Module;

Linha 4: É onde se cria a primeira camada convolucional, se assimila a seguinte ideia, seria como um filtro que observa a imagem e encontra padrões simples, como bordas ou cores;

Linha 5: A camada de pooling é responsável por reduzir o tamanho da imagem para poder simplificar o trabalho, onda ela divide a imagem em 2x2 pixels e escolhe o valor mais importante para cada quadrado;

Linha 6: Similar a primeira camada convolucional, esta segunda camada se diferencia por estar em um nível mais profundo, trabalhando com mais mapas que a primeira camada e produzindo mais mapas. É por meio destes mapas que se encontram padrões mais complexos, como formas e texturas;

Linha 7: É aqui onde se inicia a parte de classificação. A camada fc1 pega todos os padrões encontrados pelas camadas anteriores e organiza-os em números. Depois das camadas anteriores, os dados das imagens foram diminuídos para 32 mapas, onde cada um possui 8x8 pixels, o que resulta em 2048 números. Destes 2048 números, eles são transformados em uma lista menor de 64 números, que resumem as informações das imagens;

Linha 8: Nesta camada, se toma a lista de 64 número e é feita a resposta final, trazendo em que categoria a imagem se encontra;

Linha 9: Correspondente a dizer como a rede deve funcionar, o forward é como o passo a passo, descrevendo a ordem em que as camadas devem ser usadas. O x é a imagem de entrada;

Linha 10: A imagem passa pela primeira camada convolucional, que encontra padrões. Em seguida, se aplica a função F.relu(), a qual funciona como um filtro, mantendo somente os valores positivos e ajudando a rede a aprender melhor. E no final, a camada de pooling, a qual reduz o tamanho da imagem;

Linha 11: Logo mais, a imagem entra novamente em outra camada, mas agora na segunda camada, e novamente ocorre o mesmo processo que ocorreu na camada anterior, contudo, já identificando mais padrões;

Linha 12: Aqui, os dados que ainda estão na forma de mapa, são “achatados” em uma lista de números. O x.size(0) faz que com que se mantenha o número de imagens no lote;

Linha 13: Os 2048 números passam pela primeira camada, que transforma eles em 64 números, e em seguida, a função F.relu age novamente para manter apenas os valores positivos;

Linha 14: Os 64 números passam pela segunda camada, a qual dá uma pontuação para cada categoria possível para aquela imagem;

Linha 15: A rede devolve o resultado, que é a previsão da qual categoria a imagem pertence;

2.7 Visão computacional

A visão computacional, segundo Barelli (2018) em 1982, Ballard e Brown definiram a visão computacional em sua obra chamada Computer Vision, como a ciência que que estuda e desenvolve meios para permitir que computadores possam por meio de sensores, enxergar e extrair informações do que enxergam do mundo. Conforme estabelece Silva (2020), a visão computacional é comumente associada a coleta, análise e processamento dos dados visuais por meio de computadores para diversos objetivos, indo desde a identificação de rostos até a identificação de objetos.

2.7.1 Open-Source Computer Vision Library (OpenCV)

Segundo Marengoni e Stringhini (2010), OpenCV (Open-Source Computer Vision) é uma biblioteca open source criada pela Intel Corporation com mais de 500 funções. O intuito inicial da empresa, era permitir que usuários e programadores tivessem uma forma mais acessível de se utilizar a visão computacional. A biblioteca é dividida em cinco partições: Processamento de imagens; Análise Estrutural; Análise de movimentos; Reconhecimento de padrões e calibração de câmera e reconstrução 3D.

Em consonância com Fernades Neto (2020), OpenCV é utilizada pelas principais big techs do mercado, como exemplo, a Microsoft,IBM e Google. A OpenCV possui diversas funções internas para o processamento da imagem digital, mas vale-se ressaltar a identificação de bordas e a transformação de espaço de cores.

Como indicado por Antonello () o OpenCV possui uma maleabilidade maior para a aplicação com visão computacional a partir do momento que permite separar e visualizar os três canais RGB (Red,Green e Blue).

Figura 14 - Exemplo de código utilizando OpenCV

```python
import cv2
import numpy as np

imagem = cv2.imread("exemplo.jpg")
imagem = cv2.resize(imagem,(1000,1000))

imagem_hsv = cv2.cvtColor(imagem,cv2.COLOR_BGR2HSV)

verde_baixo = np.array([35,40,40])
verde_alto = np.array([85,255,255])

mask = cv2.inRange(imagem_hsv,verde_baixo,verde_alto)

resultado = cv2.bitwise_and(imagem,imagem, mask = mask)

cv2.imshow("Original",imagem)
cv2.imshow("Máscara",mask)
cv2.imshow("Resultado",resultado)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fonte: Autoria própria,2025

Linha 1 a 2: Ocorre a importação das bibliotecas para o funcionamento do código

Linha 4: Lê uma imagem chamada de “exemplo.jpg” e é armazenada na variável imagem.

Linha 5: Redimensiona a imagem para um tamanho fixo de 1000x1000 pixels

Linha 7: Converte a imagem do espaço de cores BGR (padrão do OpenCV) para HSV, o qual é mais adequado para uma segmentação de cores. A imagem convertida, é armazenada dentro da variável *imagem_hsv.*

Linha 9 e10: Definem os limites da cor verde dentro do espaço HSV, a variável *verde_baixo* define um limite inferior (tons de verde mais escuros) e a variável *verde_alto* define um limite superior (tons de verde mais claros). Estes valores são usados para isolar apenas os pixels da imagem que se encaixam nessa faixa.

Linha 12: Cria uma máscara binária (preto e branco) onde os pixels dentro da faixa de verde são brancos (255) e os demais são pretos (0). Essa máscara é utilizada para identificar as áreas com verde na imagem.

Linha 14: Aplica a máscara sobre a imagem original usando a função *bitwise_and().* Isso faz com que apenas os pixels dentro da faixa de cor verde apareçam, e o resto seja escurecido. O resultado é salvo na variável *resultado*.

Linha 16 a 18: Exibem as janelas com as imagens, o primeiro comando "*cv2.imshow()"* mostra a imagem original, logo mais, a segunda vez que chama a função, mostra a máscara binária aplicada na imagem, e na terceira e última chamada da função, é exibida a imagem filtrada com apenas a cor verde visível.

Linha 20: Espera o usuário apertar alguma tecla.

Linha 21: Fecha todas as janelas abertas do OpenCV após uma tecla ser pressionada

Figura 15 - Imagem antes do processamento usando OpenCV

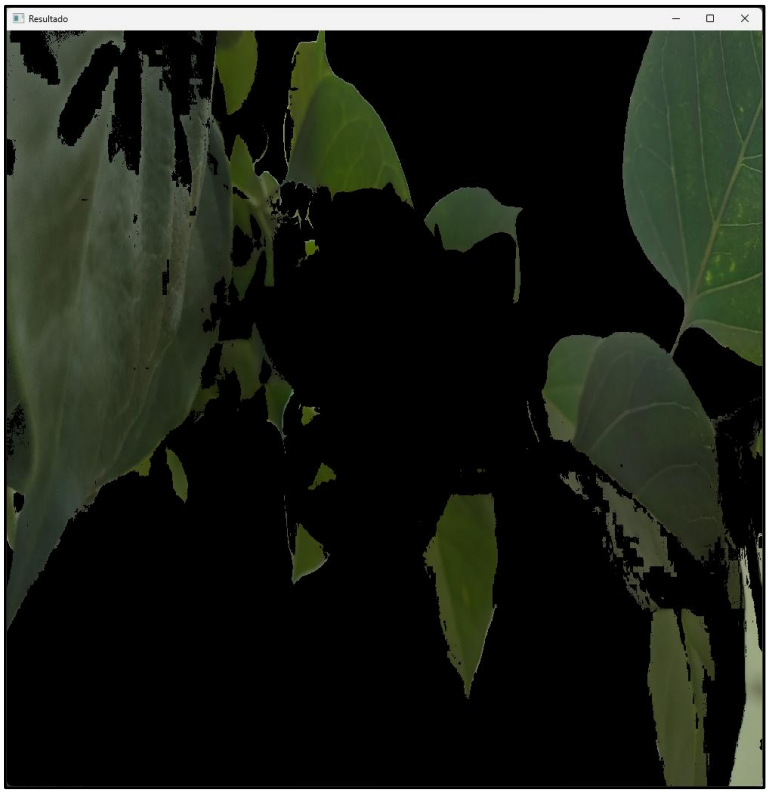
Fonte: Autoria própria,2025

Figura 16 - Aplicação do filtro binário na imagem

Fonte: Autoria própria,2025

A imagem acima mostra a aplicação da máscara binária na imagem original, onde se encontra da cor preta, é a parte da foto a qual o programa não encontrou a cor verde, e branca a parte dentro da imagem a qual o programa identificou como verde.

Figura 17 - Resultado do processamento da imagem



Fonte: Autoria própria,2025

Como resultado, a imagem acima mostra como o programa recorta toda parte que não esteja no espectro verde anteriormente definido e mantém somente aquelas que se caracterizam dentro deste espectro.

2.8 PyTorch

Em conformidade com Jeronymo (2019), PyTorch é uma biblioteca open-source para python, que busca oferecer uma flexibilidade e eficiência no treinamento de redes neurais profundas. Isto é possível, pois um dos métodos que permite isto, é uma programação prévia de funções comuns para tal proposito.

Outrossim, como informa Saavedra et al. (2022) o PyTorch se mostra como uma boa biblioteca para o uso no aprendizado de máquina, devido ao uso de tensores, estes quais, são extensão lógica de matrizes multidimensionais capazes de organizar e gerenciar os dados a serem utilizados para treinamento.

O PyTorch demonstrou uma vantagem de desempenho ao utilizar GPU’s embarcadas em relaçao a CPU’s, superior a 80 vezes, como indica Crestani *et al.* (2018)

Figura 18 - Exemplo de código utilizando PyTorch

```python
import torch
from torchvision import models, transforms
from PIL import Image

imagem = Image.open("flor.jpg")

preprocessador = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])

img_tens = preprocessador(imagem).unsqueeze(0)

modelo = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
modelo.eval()

with torch.no_grad():
    saida = modelo(img_tens)
    probabilidade = torch.nn.functional.softmax(saida[0], dim=0)
    classe = probabilidade.argmax().item()

import json
import urllib.request
with urllib.request.urlopen("https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt") as f:
    labels = [line.strip() for line in f.readlines()]

print(f"Classe prevista: {labels[classe]} (probabilidade: {probabilidade[classe]*100:.2f}%)")
```

Fonte: Autoria própria

O exemplo de código ilustrado é um código em Python que se utiliza da biblioteca PyTorch juntamento do modelo pré-treinado ResNet18 para a realizar a classificação de uma imagem. A imagem é pré-processada, classificada, e o resultado é exibido junto da janela em uma imagem.

Abaixo, segue uma breve explicação sobre o código e seu funcionamento:

Linha 1 a 4: Nestas linhas são realizadas as importações de cada biblioteca necessária para rodar o programa

Linha 5: A imagem chamada “exemplo.jpg” é carregada usando a biblioteca PIL e armazenada na variável imagem.

Linha 7 a 15: Antes de usar o modelo para fazer a classificação, o programa a prepara:

- Redimensiona a imagem para um tamanho padrão;
- Recorta o centro da imagem;
- Converte a imagem para um formato que o programa consiga compreender;

- Ajusta as cores, de forma que fique similar com os tons parecidos aos usados no Machine Learning.

Linha 17: Aplica toda essa configuração na imagem e a transforma em algo que possa ser uado pelo modelo pré-treinado.

Linhas 19 a 20: O programa carrega o modelo chamado ResNet18. Este modelo já foi treinado anteriormente para fazer o reconhecimento de milhares de tipos diferentes de imagens. Em seguida, o modelo é colocado em um modo de previsão, que é um modo de uso e não para treinamento.

Linhas 22 a 25: Nesta parte, é onde o processamento bruto é feito:

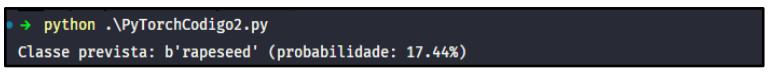
- A imagem é enviada para o modelo por meio da variável “imagem”;
- O modelo análise a imagem e retorna várias possibilidades possíveis do que pode ser aquela imagem e associa uma “nota de confiança” a cada possiblidade;
- O programa escolher a opção com a maior nota de confiança, ou seja, à opção que o modelo acha que é mais provável de ser.

Linhas 27 a 30: O programa acessa um site que possui uma lista de nomes das categorias que o modelo tem conhecimento. Por meio desta lista, é traduzida o número (por exemplo, “183”) da classe em um nome (como “cachorro vira-lata”).

Linha 32: Ao final do programa, o mesmo mostra por via de linha de comando dentro do terminal integrado a IDE, qual foi a resposta do modelo e com que confiança ele deu para tal resposta

Como resultado, o modelo identifica a imagem como “rapeseed”, uma espécie de flor, mas com uma baixa nota de confiança, o que mostra a necessidade do treinamento

Figura 19 - Resultado da análise de imagem



Fonte: Autoria própria, 2025

2.8.1 TorchVision

Consoante a Zuanazzi (2024) TorchVision é uma extensão da biblioteca PyTorch que permite usufruir-se de um conjunto de dados mais populares, arquiteturas de modelos outrossim também a transformações comumente utilizadas em visão computacional. Por meio de modelos pré-treinados, o TorchVision permite que o programador se utilize de modelos com um prévio treinamento com grandes quantidades de dados para o treinamento mais breve para a execução de tarefas específicas.

Figura 20 - Exemplo de código utilizando TorchVision

```
1  import torch
2  from torchvision import models, transforms
3  from PIL import Image
4  import cv2
5  import numpy as np
6
7  imagem = Image.open("image.jpg")
8
9  preprocessador = transforms.Compose([
10     transforms.Resize(256),
11     transforms.CenterCrop(224),
12     transforms.ToTensor(),
13     transforms.Normalize(
14         mean=[0.485, 0.456, 0.406],
15         std=[0.229, 0.224, 0.225]
16     )
17 ])
18
19 img_tens = preprocessador(imagem).unsqueeze(0)
20
21 modelo = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
22 modelo.eval()
23
24 with torch.no_grad():
25     saida = modelo(img_tens)
26     probabilidade = torch.nn.functional.softmax(saida[0], dim=0)
27     classe = probabilidade.argmax().item()
28
29 import json
30 import urllib.request
31 with urllib.request.urlopen("https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt") as f:
32     labels = [line.strip() for line in f.readlines()]
33
34 imagem_cv = cv2.cvtColor(np.array(imagem), cv2.COLOR_RGB2BGR)
35 cv2.imshow("Imagem Classificada", imagem_cv)
36 cv2.waitKey(0)
37 cv2.destroyAllWindows()
38
39 print(f"Classe prevista: {labels[classe]} (probabilidade: {probabilidade[classe]*100:.2f}%)")
40
```

Fonte: Autoria própria,2025

Linhas 1 a 5: São importadas as bibliotecas necessárias para o funcionamento do código e utilização de modelos pré-treinados;

Linha 7: Carrega uma imagem chamada *image.jpg* usando a biblioteca *Pillow* e à armazena dentro de uma variável chamada imagem;

Linha 9 a 17: Define o pré-processamento da imagem, este qual inclui o redimensionamento da imagem para 256 pixels, cortar de forma central para 224x224pixels, converter a imagem para tensor e normalizar com a média e desvio padrão das imagens do dataset (conjunto de alto volume de imagens).

Linha 19: Aplica as transformações definidadas anteriormente à imagem, e em seguida é adcionada uma dimensão extra (batch size = 1), esta qual é necessária para o modelo;

Linha 21 e 22: Carrega o modelo pré-treinado *ResNet18* e o coloca em modo de avaliação, o qual é o modo usado para poder fazer previsões sem a necessidade de um treinamento inicial;

Linha 24 a 27: Inicia um bloco onde o PyTorch não calcula os gradientes (por economia de memória e desempenho), em seguida a imagem é passada para o modelo, gerando uma saída bruta (logits). Essa saída é convertida em probabilidade com *softmax*, e a classe mais provável é extraída com *argmax*;

Linha 29 a 32: Faz o download do arquivo *imagenet_classes.txt* diretamente do GitHub, que contém os nomes de 1000 classes do ImageNet. Cada linha do arquivo representa uma classe, e é lida e armazenada na lista *labels*;

Linha 34: Converte a imagem de *PIL* para *NumPy* e altera o formato de cor de RGB para BGR (formato esperado pelo OpenCV)

Linha 35 a 37: Exibe a imaegm original com o rótulo da classe prevista usando o OpenCV, espera que o usuário pressione uma tecla e depois fecha todas as janelas abertas ligadas ao programa;

Linha 39: Imprime no terminal a classe prevista com a probabilidade associada. O valor da probabilidade é multiplicado por 100 para exibir como porcentagem.

Figura 21 - Exibição da imagem classificada com TorchVision

Fonte: Autoria própria,2025

Como a imagem abaixo ilustra, o modelo caracterizou a imagem passada como “banana” e com uma porcentagem de confiança de 56.23%.

Figura 22 - Resultado do código utilizando TorchVision

Fonte: Autoria própria,2025

2.9 Docker

Segundo Vitalino e Castro (2018), tudo se iniciou em 2008 com a fundação da dotCloud pelo CEO Solomon Hykes, contudo, o Docker nasceu somente a partir do momento em que Hykes transformou o core de sua plataforma em código aberto.

Além disso, parafraseado Gomes (2019), o Docker oferece ferramentas para poder usufruir de ambientes isolados para rodar aplicações de formas separadas, e tendo um maior poder de gerenciamento da infraestrutura da aplicação.

2.10 Banco de dados

Para Elmasri e Navathe (2011), um banco de dados de modo simples, é como uma coleção organizada de dados, designada para atender às necessidades de diversas aplicações. Toda essa estrutura é gerenciada por um Sistema de Gerenciamento de Banco de Dados (SGBD), um software que permite a definição, criação, manutenção

e controle do acesso aos dados armazenados, de acordo com Date (2004). O SGBD é como um intermediário entre os usuários e o banco de dados mantendo a integridade, segurança e disponibilidade das informações, elementos essenciais para sistemas modernos, conforme ressaltam Ramakrishnan e Gehrke (2011).

2.10.1 SQLite

De acordo com Comachio (2011), o SQLite é um banco de dados open-source que permite comandos SQL e que, na prática, cria um arquivo em disco, que mantém diversas tabelas e entidades num arquivo com extensão ".db".

Como diz Silva et al (2017), uma base de dados em servidor, tal como um Sistema Gerenciador de Banco de Dados (SGBD), pode acabar sendo uma opção robusta demais para uma aplicação simples, sendo normalmente utilizados em sistemas com alta manipulação de dados e muito armazenamento de entidades.

Assim como diz Beaulieu (2019), o SQL foi criado com a intenção de manejar o modelo relacional de banco de dados, que utiliza tabelas relacionadas entre si. Isso faz com que a administração de um banco de dados seja simples e eficaz, com apenas alguns comandos SQL.

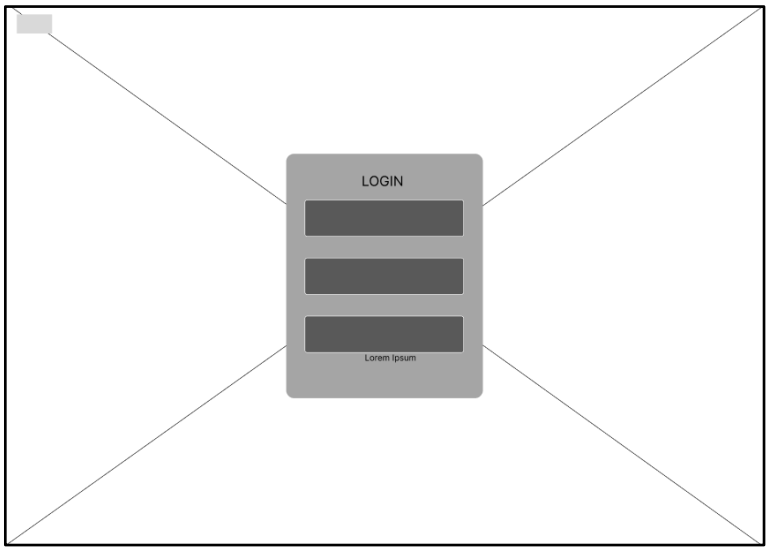
2.11 Wireframes

Wireframe, como afirma Teixeira (2022) é um rascunho, um desenho básico do produto final, para se possuir uma direção de como o desenvolver do projeto deve ocorrer.

2.11.1 Wireframe de baixa fidelidade

Os *Wireframes* de baixa fidelidade, não são totalmente fiéis a ideia final do projeto, contudo são meios que permitem a apresentação simples e rápida de uma determinada ideia para os interessados sem a necessidade de uma codificação e permitir diversas alterações devido a seu baixo custo, isto como afirma Rodrigues (2017).

Figura 23 - Exemplo de Wireframe de baixa fidelidade

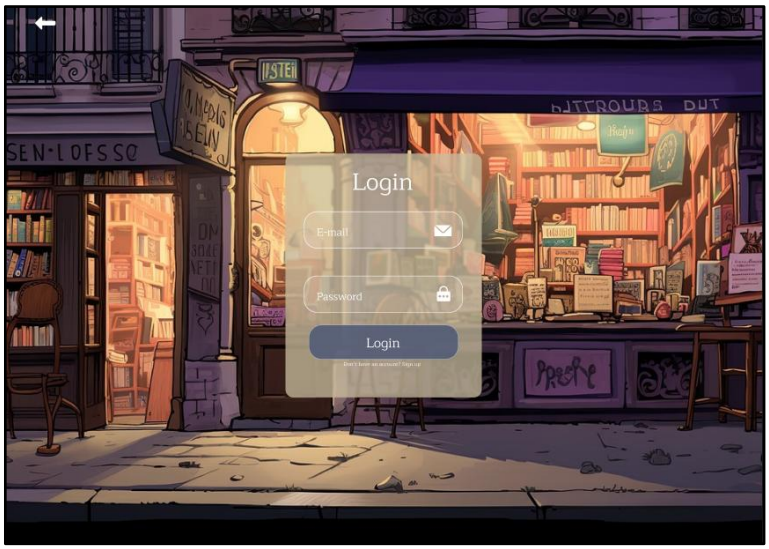


Fonte: Autoria própria,2025

2.11.2 Wireframes de alta fidelidade

Em consonância com Rodrigues (2017), a prototipação de alta fidelidade, mais conhecida como Wireframe de Alta fidelidade, utiliza-se de forma preferencial, elementos que irão estar presentes no produto final, trazendo assim um maior nível de fidelidade, além de trazer um maior desempenho para a venda do produto para a parte interessada. Contudo, este tipo de prototipação, tem como empecilho inicial o maior tempo para construção, custo mais elevado e maior dificuldade para caso haja a necessidade da alteração.

Figura 24 - Exemplo de Wireframe de alta fidelidade



Fonte: Autoria própria, 2025

2.11.3 UX (User Experience)

Segundo o Texeira (2019), *User Exeprience*, ou UX, é um modelo de design, com o objetivo de ajustar quais ações os usuários podem realizar em um sistema, quais atividades podem ser feitas e quais serão as ordens de apresentação de cada dela e como será feito essa exposição.

2.11.4 UI (User Interface)

De acordo com Krunk (2015), o *User Interface*, traduzido para Interface Visual, ou UI, se compõe dos utensílios que o usufruidor pode interagir dentro do produto para conseguir realizar suas metas e garantir que a sua alegria seja realiza ao utilizar um produto.

Em congruência Neves (2018), durante a criação da interface visuais de sistemas, é definido como será tratado e implementado a composição dos objetos visuais que busca comunicar as interações e dados da tela, durante a manipulação dos aspectos visuais é sempre importante lembrar que o ser humano possui a capacidade de diferenciar objetos apenas olhando para a sua aparência.

2.12 UML

Criado em 1995 com a unificação de três linguagem de modelagem, a Unified Modeling Language, traduzindo para o português, Linguagem de Modelagem

Unificada ou UML é uma linguagem teórica para a modelagem de softwares que seguem o modelo de orientado objeto. (Guedes,2009).

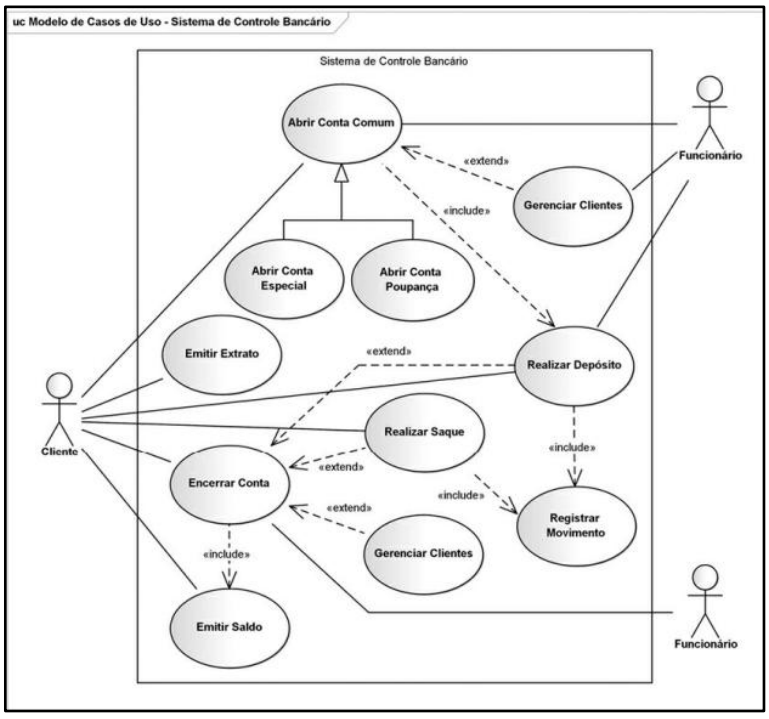
Segundo os autores Booch, Rumbaugh e Jacobson (2012), a UML é utilizada em softwares de estrutura complexa para determinar os requisitos, processos e funções de um sistema de nível simples ou avançados, além disso a UML permite definir quais serão as classes que criadas dentro do projeto e esquemas de banco de dados, a UML é o esqueleto dos projetos de software.

Consoante a Craig Larman (2005), a UML se tornou extremamente importante dentro de documentações de softwares por conta da requisição de padrões, essas normas se tornam importante no desenvolvimento criam melhores práticas para os programadores.

2.12.1 Caso de Uso:

De acordo com Fowler (2005), para descrever os requisitos funcionais de um software, utilizamos a técnica do caso de uso, o caso de uso serve para apresentar quais são as interações típicas que haverá entre os usuários e o sistema, explicando em um formato de narrativa sobre as interações. Abaixo temos um exemplo de caso de uso:

Figura 25 - Exemplo de um Caso de Uso



Fonte: Guedes, 2009, p.31

O exemplo acima exemplifica o caso de uso de um sistema de controle de banco, no esquema, podemos ver que existem 3 personagens principais que no UML chamamos de atores, sendo eles, o funcionário, cliente e outro funcionário. Para um melhor entendimento vamos explicar alguns termos:

Extend: esse termo se refere a ações que ocorrem de forma opcional dentro do sistema, por exemplo, para um funcionário, é opcional que ele abra uma conta no banco para poder gerenciar os clientes

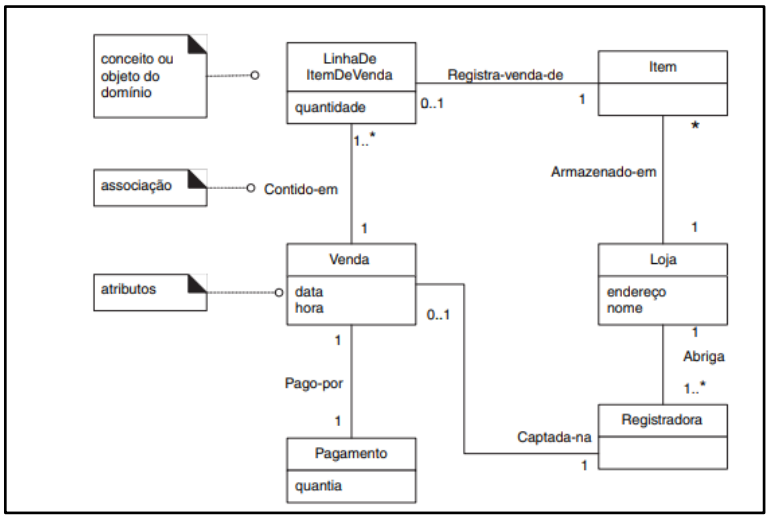
Include: Por outro lado, esse termo se refere a ações que devem acontecer caso uma outra atividade for executada, por exemplo, para um cliente poder encerrar sua conta, é obrigatório que seja emitido o saldo de sua conta.

Generalização: o termo é utilizado quando existe ações muito semelhante uma das outras, porém ainda assim são levemente diferentes e precisam ser diferenciadas, para isso, criamos um caso de uso geral e os elementos específicos serão herdados do elemento geral, por exemplo, quando o cliente escolher abrir uma conta, ele deve também escolher se ele deseja abrir uma conta especial ou uma conta poupança.

2.12.2 Diagrama de Classe:

Segundo Barbosa e Sarro (2013), diagramas de classes retratam quais serão os itens, funções e como funcionará os relacionamentos entre eles e respectivamente suas restrições, fora isso, esses diagramas conseguem exemplificar as operações de uma classe no sistema. Vamos exemplificar melhor abaixo:

Figura 26 - Exemplo de um Diagrama de Classe



Fonte: Larman, (2005), p. 53

Vemos acima um exemplo de um diagrama de classe de um sistema de Pagamento e Vendas, para poder entender melhor essa estrutura, vamos explicar certos termos:

Classe: representa um modelo dentro do sistema que irá definir quais sãos os dados e métodos, por exemplo, na imagem temos a classe “Loja”.

Atributo: também chamado de variáveis, representam as propriedades de uma classe, são nos atributos que vamos guardar os dados, por exemplo, na classe “Loja”, temos os atributos “endereço” e “nome”.

Método: são as funções e ações que a classe pode fazer no sistema, na imagem acima, não temos um método, mas poderíamos ter o método “FazerPagamento ()” (os métodos são representados por parênteses) dentro da classe Pagamento.

Associação: Exibe qual o relacionamento entre as classes, junto do relacionamento deve ter também a sua multiplicidade, que indica quantos elementos podem transmitidos de uma classe a outra, no exemplo acima a classe “Venda” se relaciona com a classe “Pagamento” com uma multiplicidade de 1 para 1, ou seja, pelo menos um elemento está sendo transmitido.

3 DESENVOLVIMENTO

4 CONSIDERAÇÕES FINAIS

REFERÊNCIAS

EMBRAPA. *Pragas quarentenárias: perguntas e respostas*. 2024. Disponível em: https://www.embrapa.br/tema-pragas-quarentenarias/perguntas-e-respostas. Acesso em: 26 maio 2025.

SENAR – SERVIÇO NACIONAL DE APRENDIZAGEM RURAL. *Café: controle de pragas, doenças e plantas daninhas*. Brasília: SENAR, 2017. 71 p. (Coleção SENAR, 190). ISBN 978-85-7664-151-3.

FERRÃO, Romário Gava; FONSECA, Aymbiré Francisco Almeida da; FERRÃO, Maria Amélia Gava; MUNER, Lúcio Herzog De (ed.). *Café Conilon*. 2. ed. atual. e ampl., 2. reimp. Vitória, ES: Incaper, 2017.

SANTOS, P. R. S.; LOPES, A. F. G.; DIAS, W. R. A. Usando RaspBerry Pi para redução energética no IFRO. **Revista de Gestão e Secretariado**, *[S. l.]*, v. 15, n. 4, p. e3596, 2024. DOI: 10.7769/gesec.v15i4.3596. Disponível em: https://ojs.revistagesec.org.br/secretariado/article/view/3596. Acesso em: 12 ago. 2025.

DOBBIN, Ivan Diniz. **Desenvolvimento de software com interface gráfica em Raspberry PI 4 para controle de uma máquina de cisalhamento**. 2022. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) – Universidade de Brasília, Faculdade UnB Gama, Brasília, 2022.

SOUZA, Andréa Lins e Lins. *Python para matemáticos*. Rio de Janeiro: Sociedade Brasileira de Matemática, 2023. eBook. ISBN 978-85-8337-216-5.

GRUS, Joel. *Data Science do Zero: Noções Fundamentais com Python*. 2. ed. Rio de Janeiro: Alta Books, 2021. 416 p. ISBN 978-85-5081-176-5.

DE PAULA, Leonardo Scarmato Jorge. Mineração de repositórios para avaliar a influência das mudanças de código ao longo do tempo. 2024. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru, 2024.

PAIXÃO, Gabriela et al. **Machine learning na medicina: revisão e aplicabilidade.** *Arquivos Brasileiros de Cardiologia*, [S. l.], [s. n.], [s. v.], [s. n.], [20--]. Disponível em:

https://www.scielo.br/j/abc/a/WMgVngCLbYfJrkmC65VFCkp/?format=pdf&lang=pt. Acesso em: 11 ago. 2025.

TSUNODA, D.F.; MOREIRA, P.S.C.; GUIMARÃES, A.J.R. **Machine learning e revisão sistemática de literatura automatizada: uma revisão sistemática.** Rev. Tecnol. Soc., Curitiba, v. 16, n. 45, p. 337-354, out./dez., 2020. Disponível em: https://periodicos.utfpr.edu.br/rts/article/view/12119. Acesso em: XXX.

SANTOS, Rogério P. dos; BEKO, Marko; LEITHARDT, Valderi R. Q.. **Modelo de Machine Learning em Tempo Real para Agricultura de Precisão**. *In*: ESCOLA REGIONAL DE ALTO DESEMPENHO DA REGIÃO SUL (ERAD-RS), 22. , 2022, Curitiba. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2022 . p. 69-70. ISSN 2595-4164. DOI:

HOSAKI, Gabriel Yuri; RIBEIRO, Douglas Francisco. Deep learning: ensinando a aprender. Revista Científica da FATEC de Assis, Assis, v. 3, n. 1, 2021. ISSN 2674-6743.

VISÃO COMPUTACIONAL E RACISMO ALGORÍTMICO: BRANQUITUDE E OPACIDADE NO APRENDIZADO DE MÁQUINA**.** Revista da Associação Brasileira de Pesquisadores/as Negros/as (ABPN), *[S. l.]*, v. 12, n. 31, 2020. Disponível em: https://abpnrevista.org.br/site/article/view/744. Acesso em: 27 maio. 2025.

MARENGONI, M.; STRINGHINI, S. **Tutorial: introdução à visão computacional usando OpenCV**. *Revista de Informática Teórica e Aplicada*, Porto Alegre, v. 16, n. 1, p. 125–160, 2010. DOI: 10.22456/2175-2745.11477.

FERNANDES NETO, Euripedes Purcinio**. Visão computacional para identificação de cores em tempo real com OpenCV e Python**. 2020. Monografia (Graduação em Engenharia da Computação) – Faculdade de Tecnologia e Ciências Sociais Aplicadas, Centro Universitário de Brasília, Brasília, 2020.

SAAVEDRA, Marcos Rodrigo Mendes; SOUZA, Flávio Ramon Almeida de; ARAÚJO, Josivaldo de Souza. **Avaliação do uso do TensorFlow e do PyTorch em uma rede neural artificial utilizada para o reconhecimento facial**. In: CONTECSI – INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGY MANAGEMENT, 19., 2022, São Paulo. *19th CONTECSI 2022 - Proceedings and Abstracts*. São Paulo: TECSI – FEA USP, 2022. p. 01-11.

CRESTANI, Angelo Nery Vieira; SOUZA, Paulo Silas Severo de; MARQUES, Wagner dos Santos; KONZEN, Marcos Paulo; ROSSI, Fábio Diniz**. Avaliando o**

desempenho do PyTorch sobre GPUs embarcadas. *In*: ESCOLA REGIONAL DE ALTO DESEMPENHO DA REGIÃO SUL (ERAD-RS) , 2018, Porto Alegre. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2018 . ISSN 2595-4164.

ZUANAZZI, T. P**. *Aprendizado profundo para a segmentação de regiões teciduais em cirurgias minimamente invasivas***. 2024. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrônica e de Telecomunicações) – Faculdade de Engenharia, Universidade Estadual Paulista “Júlio de Mesquita Filho”, São João da Boa Vista, 2024.

PEINADO, Hugo Sefrian; OLIVEIRA, Carolina Andrade de; OTTONI, André Luiz Carvalho; MELO, Roseneia Rodrigues Santos de; COSTA, Dayana Bastos; NOVO, Marcela Silva. **Análise de técnicas de data augmentation para aperfeiçoamento da detecção de sistemas de guarda-corpo e rodapés em canteiros de obras com inteligência artificial**. In: SIMPÓSIO BRASILEIRO DE GESTÃO E ECONOMIA DA CONSTRUÇÃO, 13., 2023. Anais [...]. Porto Alegre: ANTAC, 2023. p. 1–10. DOI: 10.46421/sibragec.v13i00.2585. Disponível em: https://eventos.antac.org.br/index.php/sibragec/article/view/2585. Acesso em: 11 ago. 2025.

OLIVEIRA, Walisson Santos; SILVA, Alisson Souza; MELO, Roseneia Rodrigues Santos de; BRAGA, Pedro Afonso Vieira Fernandes; COSTA, Dayana Bastos. **Uso de Data Augmentation para reconhecimento automatizado de anomalias em fachada**. In: ENCONTRO NACIONAL DE TECNOLOGIA DO AMBIENTE CONSTRUÍDO, 20., 2024. Anais [...]. Porto Alegre: ANTAC, 2024. p. 1–13. DOI: 10.46421/entac.v20i1.5843. Disponível em: https://eventos.antac.org.br/index.php/entac/article/view/5843. Acesso em: 11 ago. 2025.

CARNEIRO, Maria Sheila; GATTO, Dacyr Dante de Oliveira; SASSI, Renato José; GASPAR, Marcos Antonio. R**ede neural convolucional aplicada na análise de sentimentos em comentários de clientes de empresa do ramo varejista**. *Navus – Revista de Gestão e Tecnologia*, v. 16, e2028, 2025. DOI: https://doi.org/10.22279/navus.v16.2028. Disponível em: https://navus.sc.senac.br/index.php/navus/article/view/2028. Acesso em: 12 ago. 2025.

VARGAS, Ana Caroline Gomes; PAES, Aline; VASCONCELOS, Cristina Nader. **Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres**. In: CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES (SIBGRAPI), 29., 2016, São José dos Campos, SP, Brazil. *Proceedings*. Porto Alegre: Sociedade Brasileira de Computação, 2016. Disponível em: http://urlib.net/ibi/8JMKD3MGPAW/3ME3L2P. Acesso em: 12 ago. 2025.

BEAULIEU, Alan. **Aprendendo SQL: dominando os fundamentos de SQL**. São Paulo: Novatec, 2010. E-book. Disponível em: https://www.amazon.com.br/Aprendendo-SQL-Alan-Beaulieu-ebook/dp/B07B4J9L1M. Acesso em: 12 ago. 2025.

SILVA, Paulo; SCHANTZ, Douglas; ANTUNES, Rodrigo; SCHUCH, Regis Rodolfo. **SQLite para dispositivos móveis**. In: SEMINÁRIO INTERINSTITUCIONAL DE ENSINO, PESQUISA E EXTENSÃO, 22., 2017, Cruz Alta. Anais [...]. Cruz Alta: Universidade de Cruz Alta, 2017. p. 1-5. Disponível em: [URL do Academia.edu]. Acesso em: 12 ago. 2025.

REITZ, Kenneth; SCHLUSSER, Tanya. **O Guia do Mochileiro Python: melhores práticas para desenvolvimento.** São Paulo: Novatec, 2017.

FROTA, Hélder S.; RUVER, Fábio S.; FIGUEIREDO, Josiel. **Implementação de software desktop em Python.** In: ESCOLA REGIONAL DE INFORMÁTICA DE MATO GROSSO, 13., 2024, Alto Araguaia. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024. p. [página inicial-final]. Disponível em: https://doi.org/10.5753/eri-mt.2024.245813. Acesso em: 12 ago. 2025.

MCKINNEY, Wes. **Python para Análise de Dados: tratamento de dados com pandas, NumPy & Jupyter**. 3. ed. São Paulo: Novatec, 2023.

VASILIEV, Yuli. **Python para Ciência de Dados: uma introdução prática**. São Paulo: Novatec, 2023.

BARBOSA, Rafael de Pieri. **Modelagem de sistemas utilizando a UML: aplicando a engenharia reversa**. 2013. 84 f. Monografia (Bacharelado em Análise de Sistemas e Tecnologia da Informação) – Americana: Faculdade de Tecnologia de Americana, Centro Estadual de Educação Tecnológica Paula Souza, 2013.

LAUXEN, Rafael; LOVATTO, Andressa; SERPA DA ROSA, Ronaldo**. MicroscopePi: o desenvolvimento de um microscópio digital com Raspberry Pi 4**. In: SALÃO DE PESQUISA, EXTENSÃO E ENSINO DO IFRS, 8., 2023, Bento Gonçalves. Anais [...]. Bento Gonçalves: Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul, 2023. p. [página inicial-final]. Disponível em: https://eventos.ifrs.edu.br/index.php/salao/8salao/paper/view/2456. Acesso em: 12 ago. 2025.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. (2012). **UML Guia do Usuário.** Rio de Janeiro: Elsevier.

LARMAN, Craig. **Utilizando UML e Padrões.** Porto Alegre: Bookman, 2000

FOWLER, Martin.**UML Essencial.** Porto Alegre: Bookman, 2005

MARCONI,Marina;LAKATOS,Eva. **Fundamentos de metodologia científica.** 8. ed. São Paulo: Atlas, 2017.

TEXEIRA, Fabricio**. Introdução e boas práticas em UX Design**. São Paulo: Casa do Código, 2014

KRUK, Steve. **Não me Faça Pensar**. 2.ed. Rio de Janeiro: Alta Books, 2006

FERRÃO, Romário *et al.* **Café Conilon.** 2.ed. Vitória: Incaper, 2017.

YARA, Neves**. User Experience & User Interface Desenvolvimento e Concepção de Projetos Digitais**. Caldas da Rainha: Escola Superior de Artes e Desing, 2018