Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

GABRIEL MOSSINI DOS SANTOS

UTILIZAÇÃO DE REDES NEURAIS CONVOLUCIONAIS PARA DETECÇÃO DE OBJETOS EPI

Assis/SP
2024

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

GABRIEL MOSSINI DOS SANTOS

UTILIZAÇÃO DE REDES NEURAIS: CONVOLUCIONAIS PARA DETECÇÃO DE OBJETOS EPI

Projeto de pesquisa apresentado ao curso de Ciência da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Gabriel Mossini dos Santos
Orientador(a): Dr. Luiz Carlos Begosso

Assis/SP
2024

Santos, Gabriel Mossini dos

S237u Utilização de redes neurais convolucionais para detecção de objetos epi / Gabriel Mossini dos Santos. -- Assis, 2024.

58p.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) -- Fundação Educacional do Município de Assis (FEMA), Instituto Municipal de Ensino Superior de Assis (IMESA), 2024.

Orientador: Prof. Dr. Luiz Carlos Begosso.

1. Visão computacional. 2. Inteligência artificial. 3. Segurança do trabalho. I Begosso, Luiz Carlos. II Título.

CDD 004

Elaborada por Anna Carolina Antunes de Moraes – Bibliotecária – CRB-8/10982

UTILIZAÇÃO DE REDES NEURAIS CONVOLUCIONAIS PARA DETECÇÃO DE OBJETOS EPI

GABRIEL MOSSINI DOS SANTOS

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: _____
Dr. Luiz Carlos Begosso

Examinador: _____
Me. Guilherme de Cleva Farto

Assis/SP
2024

DEDICATÓRIA

Dedico este trabalho a minha mãe Valéria Ap. Mossini, meu pai Airton V. dos Santos e meu irmão Miguel Mossini dos Santos. E a todos que convivi ao longo desses anos de curso e que certamente tiveram impacto no meu conhecimento e formação.

RESUMO

Com o avanço das tecnologias e a crescente aplicação de Inteligências Artificiais no cotidiano, abre-se espaço para explorar o uso das Redes Neurais Convolucionais (CNNs) em diversas áreas, incluindo a segurança do trabalho. Essas redes têm se mostrado ferramentas poderosas na análise e classificação de imagens, contribuindo significativamente para áreas como a assistência médica. Este trabalho foca na utilização de CNNs, especificamente com o modelo Yolov8, para a detecção de Equipamentos de Proteção Individual (EPIs), visando a prevenção de acidentes e a promoção de um ambiente de trabalho mais seguro. A pesquisa busca demonstrar como essa tecnologia pode ser aplicada para garantir o uso dos EPIs, oferecendo suporte tanto para trabalhadores quanto para gestores em suas práticas de segurança no trabalho.

Palavras-chave: Visão Computacional, Inteligência Artificial, Segurança do Trabalho.

ABSTRACT

With the advancement of technology and the increasing application of Artificial Intelligence in everyday life, new opportunities emerge to explore the use of Convolutional Neural Networks (CNNs) in various fields, including workplace safety. These networks have proven to be powerful tools for image analysis and classification, contributing significantly to areas such as medical assistance. This study focuses on the application of CNNs, specifically the Yolov8 model, for detecting Personal Protective Equipment (PPE), aiming at accident prevention and promoting a safer work environment. The research seeks to demonstrate how this technology can be applied to ensure proper PPE usage, offering support to both workers and managers in their safety practices.

Keywords: Computer Vision, Artificial Intelligence, Occupational Safety.

Lista de ilustrações

Lista de tabelas

LISTA DE ABREVIATURAS E SIGLAS

EPIs: Equipamentos de Proteção Individual.
CNNs: Redes Neurais Convolucionais em Inglês.
Kernel: Uma Matriz.
NR: Norma Regulamentadora
NBR: Norma Brasileira
AP: Average Precision
IoU: Interseção sobre União
Bounding Boxes: Caixas delimitadoras

LISTA DE SÍMBOLOS

$\phi(\cdot)$: Função de ativação.

θ_k: Limiar aplicado ao neurônio

α: Parâmetro de inclinação da função logística.

y_k: Sinal de saída

SUMÁRIO

1. INTRODUÇÃO

Este trabalho de conclusão visa aprimorar a segurança no local de trabalho, otimizar o cumprimento das normas de segurança e reduzir riscos ocupacionais por meio da automação na detecção de EPIs.

Na área da detecção EPIs em ambientes de trabalho por meio de CNNs, algumas lacunas e desafios se destacam. Primeiramente, existe a necessidade de analisar a precisão e a confiabilidade dos modelos de detecção, especialmente em condições variáveis de iluminação e ambiente, sendo que estes modelos como YOLO (You Only Look Once), R-Cnn (e suas variantes), SSD (Single Shot MultiBox) entre outros. Estão em estado da arte como destacado por Jocher (2023) sobre YOLO e LI et al., (2022) sobre Mask Dino. Além disso, a adaptação de algoritmos de CNNs para detectar uma ampla gama de tipos e modelos de EPIs, considerando variações de design e aparência, representa um desafio significativo, como destacado por Gallo (2022), a dificuldade como falta de etiquetas para alguns objetos, a presença de trabalhadores distante das câmeras e a predominância de capacetes em relação a outros EPIs, são um dos desafios na detecção de objetos.

Outro ponto a ser considerado é a eficácia da detecção em tempo real, uma vez que muitos ambientes industriais demandam respostas instantâneas para garantir a segurança dos trabalhadores, também destacado por Gallo (2022). A compreensão dos limites éticos, privacidade e segurança dos dados também é uma lacuna crucial, onde Gallo (2022) destaca sua preocupação com a privacidade e o tratamento de dados na nuvem, mencionando que ela não é preservada devido à análise das imagens ser em uma infraestrutura externa geralmente ser propriedade de uma terceira empresa.

Segundo de Souza e de Melo (2020):

> Os Equipamentos de Proteção Individual (EPI), entendidos como medidas de controle para a redução de acidentes de trabalho, são definidos como todos os dispositivos usados pelo trabalhador, de forma individual, para se proteger de riscos que possam ameaçar sua saúde e segurança no trabalho. A utilização de CNNs pode ser uma abordagem promissora para aprimorar a detecção e o uso eficaz desses equipamentos no ambiente de trabalho, reduzindo assim o risco a saúde do trabalhador.

Como destaca o Tribunal Superior do Trabalho (2021), Dados do Observatório de Segurança e Saúde no Trabalho da Plataforma SmartLab, uma iniciativa conjunta do

Ministério Público do Trabalho (MPT) e da Organização Internacional do Trabalho (OIT), em 2020, ocorreram 46,9 mil acidentes de trabalho no Brasil, com subnotificações prováveis. Entre 2012 e 2020, as lesões mais comuns foram cortes, fraturas e contusões. Membros mais afetados: dedos, pés, mãos e joelhos. Principais causas: máquinas, agentes químicos e quedas. Ocupações mais citadas: alimentador de linha de produção, técnico de enfermagem e faxineiro. No INSS, as fraturas representam 40% dos afastamentos, seguidas por problemas osteomusculares.

Na detecção de objetos, é crucial estabelecer de forma clara o modelo de CNN a ser utilizado com o intuito de maximizar a detecção e identificação de objetos. Isso constitui um problema clássico na visão computacional, que envolve o reconhecimento do quê e onde estão localizados os objetos em uma determinada imagem. Contudo, não fornece a localização precisa dos objetos na imagem. Além disso, outro desafio enfrentado é a classificação de múltiplos objetos em uma única imagem, o que se torna uma significativa barreira para as CNNs, como destacado por Lima e Cunha (2023).

O objetivo deste trabalho é analisar e avaliar a eficácia da utilização de CNNs na detecção de EPIs no local de trabalho. O estudo visa contribuir para a melhoria da segurança no local de trabalho, explorando a aplicação de tecnologias avançadas de visão computacional, como as CNNs, na identificação e monitoramento de EPIs, com o propósito de reduzir os riscos à saúde e segurança dos trabalhadores e promover a conformidade com as regulamentações de segurança no trabalho.

O trabalho está organizado em seis capítulos principais. O Capítulo 1, introdução, oferece uma visão geral do estudo, apresentando os objetivos gerais (Seção 1.1) e específicos (Seção 1.2), a justificativa (Seção 1.3), a motivação (Seção 1.4) e as perspectivas de contribuição (Seção 1.5). Capítulo 2 trata da revisão da literatura, abordando os conceitos fundamentais sobre redes neurais (Seção 2.1), perceptron (Seção 2.2) e redes neurais convolucionais (Seção 2.3), com subseções detalhadas sobre as camadas de convolução (Seção 2.3.1), camadas de pooling (Seção 2.3.2), camadas totalmente conectadas (Seção 2.3.3) e perceptron multicamadas (Seção 2.3.4).

O Capítulo 3 explora a metodologia de pesquisa, seguida pelo Capítulo 4, que discute o desenvolvimento do estudo. Este capítulo inclui as tecnologias utilizadas (Seção 4.1), classes de objetos (Seção 4.2), a abordagem com o modelo YOLO (Seção 4.3), desenvolvimento do dataset (Seção 4.4), treinamento do dataset (Seção 4.5) e a

implementação do framework Django, com subseções específicas sobre VENV (Seção 4.6.1), MySQL (Seção 4.6.2) e a integração entre Django e YOLO (Seção 4.6.3).

O Capítulo 5 apresenta os resultados da aplicação, incluindo autenticação (Seção 5.1), detecção (Seção 5.2), usuários (Seção 5.3) e resultados (Seção 5.4). Por fim, o Capítulo 6 traz a conclusão do estudo, apontando possíveis trabalhos futuros (Seção 6.1).

1.1. OBJETIVOS GERAIS

O objetivo geral deste estudo consiste em investigar a aplicação de CNNs para aprimorar a detecção de objetos de EPIs. A finalidade primordial é aprimorar a segurança no ambiente de trabalho, desenvolvendo um sistema robusto e eficiente que seja capaz de identificar com precisão a presença e o posicionamento de EPIs em cenários laborais, visando reduzir potenciais riscos à saúde dos trabalhadores.

1.2. OBJETIVOS ESPECÍFICOS

Inicialmente, foi realizada uma revisão detalhada da literatura, com foco nas aplicações de Redes Neurais Convolucionais (CNNs) para detecção de objetos em contextos industriais, especialmente na identificação de Equipamentos de Proteção Individual (EPIs). Essa revisão teve como objetivo aprofundar a compreensão das abordagens existentes, identificar lacunas na pesquisa e apontar oportunidades de aprimoramento.

Em seguida, o estudo coletou e curou um conjunto de dados representativo, contendo imagens relevantes de ambientes de trabalho onde a presença e o uso de EPIs eram cruciais. Esse conjunto de dados serviu como base para o desenvolvimento e otimização de uma arquitetura de CNN específica, com foco na detecção precisa de EPIs. O aprimoramento dos critérios de precisão, eficiência computacional e escalabilidade foi considerado fundamental. Os objetivos estabelecidos neste estudo visaram fornecer uma base para a implementação de sistemas de detecção de EPIs em ambientes industriais.

1.3. JUSTIFICATIVA

A condução deste estudo se baseia em justificativas sólidas, enfatizando a importância da pesquisa no contexto acadêmico e prático. Em primeiro lugar, a atualidade do tema é inegável, uma vez que a segurança do trabalho assumem uma relevância crescente devido aos desafios emergentes na manutenção de ambientes de trabalho seguros. A utilização de Redes Neurais Convolucionais (CNNs) para a detecção de Equipamentos de Proteção Individual (EPI) oferece uma solução contemporânea e eficaz para abordar estes desafios.

A inovação desempenhou um papel fundamental nesta pesquisa, ao explorar o potencial das CNNs na detecção de EPIs. A inovação foi evidenciada pela capacidade dessas redes neurais em proporcionar eficiência e precisão, algumas abordagens convencionais enfrentariam dificuldades para alcançar. Esse enfoque teve o potencial de apoiar como a segurança do trabalho é gerenciada, trazendo, assim, uma contribuição para o tema.

Por fim, esta pesquisa enquadrou-se de um interesse profundo na exploração das capacidades das CNNs e sua aplicação na segurança ocupacional. O aprimoramento da segurança no local de trabalho pode proteger os trabalhadores, mas também reduziu custos associados a acidentes, que podem variar de graves a fatais, além de impulsionar a produtividade das indústrias (Cunha, 2022). Como resultado, espera-se que este trabalho tenha contribuído substancialmente, estabelecendo um novo paradigma na segurança do trabalho e proporcionando benefícios tangíveis tanto para a comunidade acadêmica quanto para a sociedade em geral.

1.4. MOTIVAÇÃO

Este trabalho é motivado pela necessidade de explorar e aplicar as capacidades das Redes Neurais Convolucionais (CNNs) como uma solução para a detecção de objetos EPI. A motivação é impulsionada pela aspiração de melhorar a proteção dos trabalhadores, reduzir custos associados a acidentes e aumentar a eficiência na gestão da segurança nos ambientes de trabalho. Portanto, este estudo se enquadra de uma motivação intrínseca em busca de soluções inovadoras e eficazes para aprimorar a segurança ocupacional por meio da tecnologia.

1.5. PERSPECTIVAS DE CONTRIBUIÇÃO

As possíveis contribuições deste estudo para a área de visão computacional e aprendizado de máquina, com foco na aplicação de Redes Neurais Convolucionais (CNNs), podem ser divididas em três aspectos distintos. Primeiramente, este trabalho tem o potencial de aprimorar significativamente a precisão na detecção de objetos em imagens, uma vez que as CNNs demonstraram excelência na extração de padrões complexos.

Além disso, as contribuições deste estudo podem estender-se a diversos campos, como a eficiência computacional, onde a otimização de recursos pode acelerar a implementação prática dessas técnicas em várias aplicações. Essa eficiência é de particular relevância no contexto da visão computacional, onde o processamento de imagens de alta resolução é comum.

As contribuições potenciais deste estudo para a área de visão computacional e aprendizado de máquina, com foco na aplicação de Redes Neurais Convolucionais (CNNs), podem ser divididas em três aspectos principais. Primeiramente, este trabalho tem o potencial de aprimorar significativamente a precisão na detecção de objetos em imagens, dado que as CNNs têm se mostrado altamente eficazes na extração de características e padrões complexos, superando as limitações de abordagens convencionais.

Além disso, as contribuições deste estudo podem impactar diretamente a eficiência computacional, uma vez que a otimização dos recursos pode acelerar a aplicação prática dessas técnicas em diversas áreas. Esse ganho de eficiência é particularmente relevante no campo da visão computacional, onde o processamento de imagens de alta resolução é frequente e exige alta capacidade de processamento.

Por fim, ao propor novas abordagens para detecção de objetos, o trabalho pode impulsionar futuras pesquisas em visão computacional e aprendizado de máquina, ajudando a abrir novas possibilidades e aplicações em vários campos. A exploração do potencial das CNNs pode gerar avanços importantes, contribuindo para o crescimento do conhecimento e trazer benefícios para a sociedade.

2. REVISÃO DA LITERATURA

Este capítulo aborda os principais conceitos fundamentais e desenvolvimentos na área de redes neurais. A revisão cobre desde o perceptron básico até o tema do projeto com as redes neurais convolucionais, incluindo camadas e estruturas associadas. O objetivo é proporcionar uma compreensão abrangente das estruturas técnicas que formam a base das redes de convolução.

2.1. REDE NEURAL

As redes neurais artificiais constituem de modelos computacionais concebidos com inspiração nas funcionalidades do cérebro humano. Sua finalidade primordial não consiste em uma reprodução direta do sistema neural humano, mas sim em fornecer uma base para aprendizagem e abordagem de problemas de natureza complexa. Segundo Haykin (2001) a rede neural é uma máquina projetada para modelar a maneira que o cérebro realiza uma tarefa em particular ou função de interesse; a rede é normalmente implementada utilizando-se componentes eletrônicos ou é simulada por programação em computador.

De acordo com Furtado (2019), no neurônio artificial, as entradas funcionam como os dendritos do neurônio biológico, e as conexões com o corpo celular artificial são feitas através de canais de comunicação com pesos específicos. Os estímulos recebidos são processados por uma função chamada soma, que simula a integração dos sinais no neurônio real. Já o disparo do neurônio biológico é substituído, no modelo artificial, por uma função de transferência. A Figura 1 mostra o modelo de neurônio artificial criado por McCulloch e Pitts.

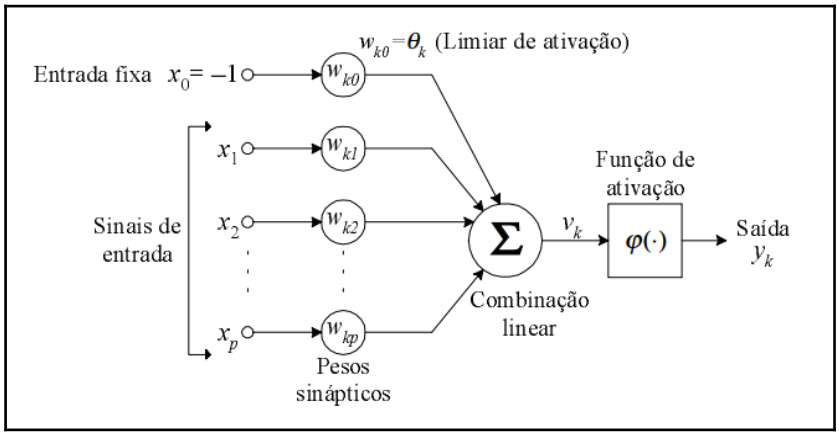

Figura 1: Modelo de neurônio artificial de McCulloch e Pitts
Fonte: Fernandes, 1999

O modelo de neurônio de McCulloch e Pitts persiste em sua relevância mesmo em meio aos avanços tecnológicos. Neste modelo, criado para simular uma célula do sistema nervoso é descrito matematicante por:

$$v_k = \sum_{j=0}^{p} w_{kj} x_j$$

E seu modelo simplificado é:

$$y_k = \varphi(v_k)$$

Onde x_1, x_2, ..., x_p são os sinais de entrada; w_{k1}, w_{k2}, ..., w_{kp} são os pesos dos neurônios; v_k é a saída da combinação linear; θk é o limiar de ativação; ϕ (·) é a função de ativação; e y_k é o sinal de saída. Assim, os sinais de entrada são combinados linearmente com os pesos correspondentes para produzir v_k, que é então transformado pela função de ativação ϕ (·) para obter o sinal de saída do neurônio (Fernandes, 1999).

A função de ativação tem o papel crucial de processar o sinal resultante da combinação linear entre entradas e pesos das sinapses, gerando o sinal de saída correspondente. Alguns exemplos de neurônios são:

Função Linear: Também conhecida como função identidade, produz resultados de saída que são idênticos aos valores do potencial de ativação {u}. Matematicamente, esta abordagem é representada pela expressão:

$$g(u) = u$$

Função de Limiar: Normalmente restringe a saída da RNA em valores binários [0, 1]. A saída do neurônio assume valor 0 quando seu resultado for negativo, e 1 caso contrário (FLECK, 2016).

$$f(u) = \begin{cases} 1 & \text{se } u \geq 0 \\ 0 & \text{se } u < 0 \end{cases}$$

Função de Linear por partes: De acordo com FLECK (2016), a função linear por partes assume que o fator de amplificação dentro de sua região linear é unitário. Ela generaliza outras funções de ativação, comportando-se como uma combinação linear quando a região linear não está em saturação. Quando o fator de amplificação é muito alto, a função se aproxima da função de limiar.

$$f(u) = \begin{cases} u & \text{se } -\frac{1}{2} < u < \frac{1}{2} \\ u^2 + \frac{1}{2} & \text{se } u \geq \frac{1}{2} \\ 0 & \text{se } u \leq -\frac{1}{2} \end{cases}$$

Função sigmoide: é uma das funções de ativação mais comuns em redes neurais. Ela é crescente e gera saídas entre 0 e 1, equilibrando comportamento linear e não linear. Um exemplo popular de função sigmoide é a função logística, na qual "α" representa o parâmetro de inclinação - quanto maior o valor de "α", mais inclinada se torna a curva (Fleck, 2016).

$$f(\mu) = \frac{1}{1+\exp(\alpha\,\mu)}$$

Função Tanh: A Tangente Hiperbólico (Tanh) é semelhante ao da função sigmoide, no entanto, é simétrico em relação à origem e varia de -1 a 1. Essa função resolve o problema dos valores estarem todos do mesmo sinal. Todas as outras propriedades do Tanh são semelhantes às da função sigmoide. (DATA SCIENCE ACADEMY, 2022).

$$tahn(x) = \frac{2}{\left(1+e^{(-2x)}\right)}$$

Função ReLU: A função ReLU (Rectified Linear Unit) é amplamente utilizada na concepção de redes neurais modernas, sendo não linear, permitindo a propagação eficiente de erros através de várias camadas de neurônios ativados por ela. Sua principal vantagem sobre outras funções de ativação é a capacidade de ativar apenas os neurônios relevantes, tornando a rede leve e eficiente em termos computacionais. No entanto, pode enfrentar problemas com gradientes nulos (DATA SCIENCE ACADEMY, 2022).

$$\left(f(x) = max(0\,x)\right)$$

Abaixo na figura 2 o autor faz uma comparação entre as funções linear, linear por partes, limiar, sigmóide, tahn e ReLu, sendo este gráfico desenvolvido utilizando as bibliotecas NumPy e Matplotlib em Python.

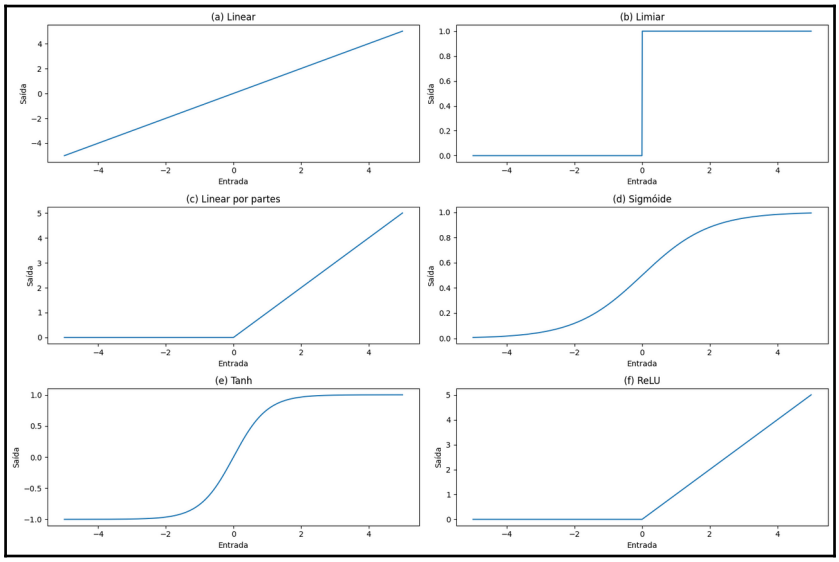

Figura 2: Comparação de todos os neurônios
Fonte: Autoria própria, 2024

2.2. PERCEPTRON

O conceito de Perceptron, que foi implementado por Frank Rosenblatt como parte de uma pesquisa realizada no Cornell Aeronautical Laboratory, teve como objetivo criar um análogo do cérebro capaz de realizar tarefas de reconhecimento de padrões e estabelecer a viabilidade técnica e econômica do perceptron. O perceptron é uma evolução do modelo de neurônio de McCulloch-Pitts, proposto em 1943, com a diferença fundamental de que o perceptron pode aprender ajustando seus pesos, enquanto os pesos do neurônio McCulloch-Pitts são fixos.

De acordo com Rosenblatt (1958) a teoria que o perceptron é baseada pode ser resumida da seguinte forma:

1. As conexões nos sistemas nervosos relacionados ao aprendizado e reconhecimento não são iguais. Ao nascer, a formação das redes neurais é em grande parte aleatória, com poucas limitações genéticas;
2. O sistema de células conectadas tem certa flexibilidade. Após a atividade neural, a chance de um estímulo provocar uma resposta em outras células pode mudar, devido a alterações duradouras nos neurônios;

3. Quando expostos a muitos estímulos, os que são mais semelhantes tendem a ativar os mesmos grupos de células. Estímulos muito diferentes, por outro lado, costumam se conectar a grupos distintos;
4. O uso de reforço positivo ou negativo pode ajudar ou dificultar a formação de conexões que estão em desenvolvimento;
5. A similaridade, nesse sistema, significa que estímulos parecidos ativam os mesmos grupos de células. Essa similaridade não está relacionada a formas ou categorias específicas de estímulos, mas depende da organização do sistema nervoso, que muda com a interação com o ambiente. Tanto a estrutura do sistema quanto a relação estímulo-ambiente influenciam como percebemos e classificamos o mundo ao nosso redor.

Sendo assim o perceptron é um algoritmo que nos permite aprender e processar elementos do conjunto de treinamento.

Segundo Banoula (2023), o perceptron pode ser dividido em dois modelos. O Modelo de Perceptron de Camada Única é um dos mais simples de Redes Neurais Artificiais, consistindo em uma rede de alimentação direta que inclui uma função de transferência com limiar. O principal objetivo desse modelo é analisar objetos que são linearmente separáveis, produzindo resultados binários. Já o Modelo de Perceptron de Múltiplas Camadas é semelhante ao modelo de camada única, mas possui camadas adicionais que permitem o aprendizado de padrões mais complexos.

2.3. REDES NEURAIS CONVOLUCIONAIS

Redes Neurais Convolucionais são algoritmos de Deep Learning que analisa uma imagem de entrada, atribui importância a diferente aspectos ou objetos da imagem por meios de pesos e viesses aprendidos, sendo assim capaz de distinguí-los. Segundo Karpathy (2014), devido às restrições computacionais, as CNN’s até recentemente foram apenas aplicadas para imagens em pequena escala para reconhecimento de problemas (como mnsit, cifar-10/100, norb e caltech-101/256), mas com os avanços de hardwares de GPU (Graphic Processing Unit) foi possibilitado aumentar as redes para milhões de parâmetros, no qual trouxe melhorias para a classificação de imagens (Krizhevsky, 2012).

Um exemplo no quesito de classificação de imagens é a utilização para a detecção de dígitos de casas baseando-se em diversos formatos de números nas ruas utilizando as CNNs, onde foi obtida uma precisão de estado da arte de 94.58% de acurácia (Sermanet; Chintala; Lecun, 2012).

Outro exemplo é a integração do reconhecimento, localização e detecção de objetos, onde foi introduzido o aprendizado para prever as bordas de um objeto, em que as bordas são acumuladas em vez de suprimidas, com o objetivo de aumentar a confiança na detecção (Sermanet, 2013).

A arquitetura de uma CNN é estruturada como uma série de estágios. Segundo LeCun, Bengio e Hinton (2015), os primeiros estágios incluem dois tipos de camadas: camadas convolucionais e camadas de pooling. As unidades em uma camada convolucional são organizadas em mapas de características, onde cada unidade está conectada a regiões locais dos mapas de características da camada anterior por meio de um conjunto de pesos chamado filtros. O resultado dessa soma ponderada local é, então, passado por uma função de ativação não linear, como a ReLU. Em uma arquitetura típica de CNN, todas as unidades em um mapa de características compartilham o mesmo conjunto de filtros, enquanto diferentes mapas de características utilizam conjuntos de filtros distintos.

Essa abordagem tem duas razões principais. Primeiro, em dados matriciais, como imagens, grupos locais de valores frequentemente são altamente correlacionados, formando padrões locais distintos que podem ser facilmente detectados. Em segundo lugar, as estatísticas locais de imagens e outros sinais são invariantes à localização. Ou seja, se um padrão pode aparecer em uma parte da imagem, ele pode aparecer em qualquer lugar. Assim, unidades em diferentes localizações compartilham os mesmos pesos e conseguem detectar o mesmo padrão em diferentes partes da matriz.

A figura 3 ilustra as saídas (não os filtros) de cada camada (horizontalmente) de uma arquitetura típica de rede convolucional aplicada à imagem de um cachorro (canto inferior esquerdo; e entradas RGB (vermelho, verde, azul), canto inferior direito). Cada imagem retangular é um mapa de características, uma matriz 2D de neurônios (Lecun; Bengio; Hinton, 2015).

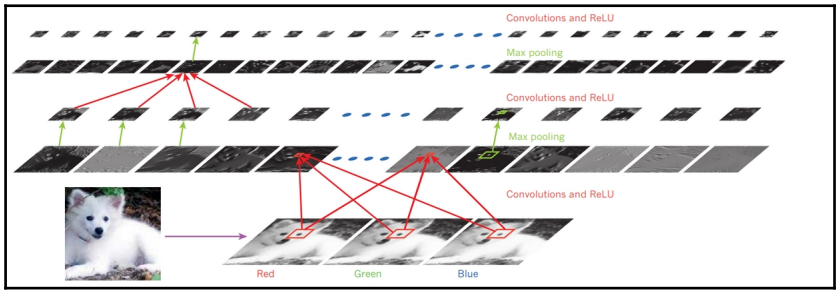

Figura 3: Exemplo de uma rede convoluional
Fonte: Lecun; Bengio; Hinton, 2015

Vale a pena destacar também uma terceira camada, chamada camada totalmente conectada e quando estão empilhadas, uma rede CNN é formada como demostrado na figura 4.

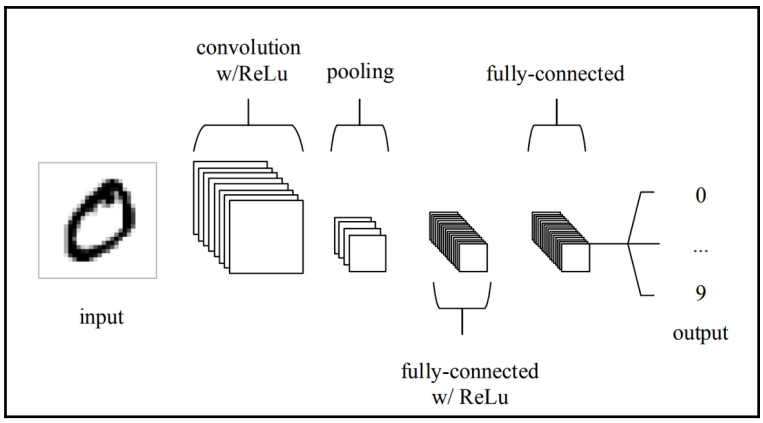

Figura 4: Arquitetura simples composta por cinco camadas
Fonte: Keiron O'shea; Nash, 2015

De acordo com Keiron O'Shea e Nash (2015) as funcionalidades básicas de uma arquitetura de rede neural totalmente conectada pode ser divido em quatro áreas principais:

1. A camada de input que irá definir os valores do pixel da imagem;
2. A camada de convolução que irá determinar a saída dos neurônios no qual são conectados em uma região local da entrada atráves de cálculos do produto escalar entre os pesos e as regiões conectadas na entrada. A ReLU tem como objetivo aplicar elemento por elemento da função assim como uma sigmoide na saida da ativação da camada anterior;
3. A camada de pooling irá então simplesmente performar uma redução de resolução atráves da dimensidade espacial da entrada, em seguida reduzindo o número de paramentos dentro da ativação;
4. A camada totalmente conectada irá performar as mesmas obrigações encontradas em outras Redes Neurais Artificiais e tentará produzir pontuações de classe

através de ativações, que são usadas para classificação. Que também sugere que ReLu talvez possa ser utilizado entre as camadas, para melhora de performance.

Através de um simples método de transformação, uma CNN pode conseguir transformar uma entrada camada por camada usando técnicas de convoluções e redução de resolução para produzir suas classificações. No entanto, é crucial compreender as camadas de convolução, camadas de pooling e camadas totalmente conectadas para uma implementação eficaz e compreensão completa do funcionamento da rede neural convolucional.

2.3.1. CAMADAS DE CONVOLUÇÃO

A camada de convolução é quem faz o papel principal de um CNN. Segundo Zazo e Protopapas (2018), as camadas convolucionais são compostas por kernels, mapas de características e funções de ativação. Os filtros realizam essencialmente a convolução na camada, e seu tamanho determina o número de parâmetros a serem treinados na rede. Se a camada de entrada for uma imagem, o filtro irá convoluir os pixels da imagem.

O mapa de características é a saída de uma camada anterior. Dependendo do passo do mapa de características (o número de pixels que o filtro se move de uma amostragem da saída para a próxima) e de preenchimento da camada de entrada (o número de zeros adicionados na camada de entrada para controlar o tamanho da convolução), o tamanho da saída é determinado (Zazo; Protopapas, 2018).

À medida que percorremos a entrada, o produto escalar é calculado para cada valor nesse kernel. Na figura 5, a rede aprenderá kernels que "disparam" quando detectam uma característica específica em uma determinada posição espacial da entrada. Esses são comumente conhecidos como ativações. O elemento central do kernel é posicionado sobre o vetor de entrada, o qual é então calculado e substituído por uma soma ponderada de si e de quaisquer pixels próximos (Keiron O'shea; Nash, 2015).

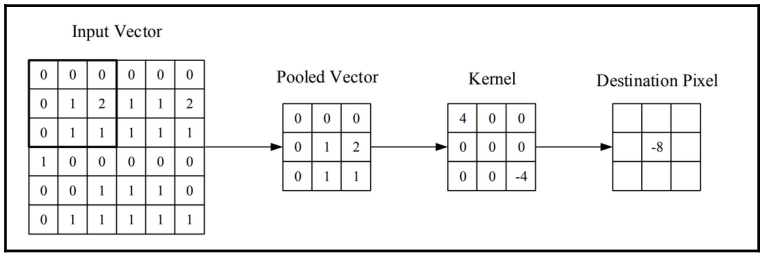

Figura 5: Mapa de ativação de camadas
Fonte: Keiron O'shea; Nash, 2015

Cada filtro terá um mapa de ativação correspondente, que será empilhado ao longo da dimensão de profundidade para formar o volume de saída completo da camada de convolução (Keiron O'shea; Nash, 2015).

Considere uma imagem de entrada em escala de cinza de 63 × 63 pixels. Se usarmos 8 filtros de tamanho f = 3, s = 1 e p = 2, acabaremos com uma saída de 63 × 63 × 8. Isso é chamado de convolução 'same', porque o preenchimento ajuda a saída a ter o mesmo tamanho que a entrada. Se usarmos 8 filtros com f = 3, p = 0 e s = 2, teremos uma saída de 30 × 30 × 8. Isso é chamado de convolução 'valid', e reduz pela metade o tamanho das camadas de entrada. O número de filtros (ou canais) nC = 8 é decidido entre as camadas e é uma característica de design. Por fim, uma camada convolucional geralmente é concluída com uma função de ativação não linear elemento a elemento, como as unidades ReLU (Zazo; Protopapas, 2018).

2.3.2. CAMADA DE POOLING

De acordo com James, et al (2023), a camada de pooling fornece uma maneira de condensar uma imagem grande em uma imagem de resumo menor. Embora existam várias maneiras possíveis de realizar o pooling, a operação de max pooling resume cada bloco não sobreposto de 2 × 2 pixels em uma imagem usando o valor máximo no bloco. Isso reduz o tamanho da imagem pela metade em cada direção, e também fornece alguma invariância de localização: ou seja, se houver um valor grande em um dos quatro pixels no bloco, todo o bloco é registrado como um valor grande na imagem reduzida.

$$\text{Max Pool} \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}$$

2.3.3. CAMADAS TOTALMENTE CONECTADAS

De acordo com Zazo e Protopapas (2018), as camadas totalmente conectadas são camadas normais do tipo feed-forward plano. Essas camadas geralmente estão no final de uma rede convolucional. Para conectar essas camadas com uma camada convolucional ou de pooling típica, sua saída é normalmente vetorizada e, em seguida,

são criadas conexões com uma camada FC subsequente ou uma camada de saída. Esses neurônios também incorporam funções de ativação não lineares típicas de redes feed-forward, como ReLU, sigmoide, tangente hiperbólica ou entropia cruzada para validação de saída.

2.3.4. PERCEPTRON MULTICAMADAS

Embora compartilhe semelhanças com a camada totalmente conectada, o perceptron multicamadas apresenta distinções fundamentais. Essas diferenças incluem a presença de uma ou mais camadas totalmente conectadas, a incorporação de funções de ativação não lineares entre as camadas e a existência de múltiplas camadas de neurônios.

3. METODOLOGIA DE PESQUISA

A condução do trabalho para explorar a utilização de redes neurais convolucionais é fundamentada na necessidade de aprimorar os métodos de identificação de Equipamentos de Proteção Individual (EPIs) e consequentemente a segurança do trabalhador. Este estudo visa explorar o conhecimento teórico e prático no campo das Redes Neurais Convolucionais (CNNs), concentrando-se em sua aplicação específica para detectar objetos relacionados à segurança individual em ambientes diversos. A metodologia proposta abrange uma série de estágios cuidadosamente estruturados para explorar, compreender e implementar essas técnicas com o objetivo de contribuir significativamente para a segurança e eficácia no reconhecimento de EPIs.

Inicialmente, foi conduzido um amplo Levantamento de Referências Bibliográficas. Este estágio tinha como objetivo reunir informações pertinentes às Redes Neurais Convolucionais (CNNs) e à detecção de objetos, com foco específico na aplicação em Equipamentos de Proteção Individual (EPIs). Pretendeu-se estabelecer uma base teórica sólida, identificando abordagens e estudos relevantes para embasar a pesquisa.

Paralelamente, foi realizado o Levantamento de Requisitos específicos para a detecção de objetos EPI por meio de CNNs. Esse processo contribuiu para definir critérios essenciais para o desenvolvimento e implementação dos algoritmos necessários.

Seguindo, um período foi dedicado ao estudo sobre Python, uma linguagem fundamental para a implementação e experimentação dos modelos de redes neurais convolucionais.

Posteriormente, o estudo sobre Redes Convolucionais permitiu uma exploração aprofundada dos princípios teóricos dessas redes. O foco esteve nas suas arquiteturas, funcionamento e aplicações na detecção de objetos, especialmente relacionados aos EPIs.

Uma etapa subsequente foi o estudo sobre PyTorch, TensorFlow, YOLO e visão computacional, destinado à análise e comparação das estruturas e funcionalidades dessas bibliotecas. O objetivo é selecionar a mais adequada para a implementação dos modelos de CNNs.

O documento de qualificação foi redigido para sintetizar os estágios iniciais da pesquisa, incluindo revisão bibliográfica, requisitos identificados e metodologias escolhidas para a implementação.

O exame de qualificação foi conduzido para avaliar a consistência e relevância dos elementos apresentados até então, validando a direção e os métodos adotados.

A implementação de conjuntos de imagens envolveu a implementação e treinamento dos modelos de redes neurais convolucionais utilizando conjuntos de imagens específicos de EPIs, visando uma detecção precisa em variados contextos.

Após a fase de implementação, houve uma análise e avaliação dos resultados dos conjuntos de imagens, compreendendo uma avaliação minuciosa dos resultados obtidos, comparando o desempenho dos modelos implementados com métricas relevantes para a detecção de objetos epi.

A investigação sobre a aplicação das redes neurais convolucionais na detecção de objetos epi ressalta a relevância desse campo de estudo para a segurança e identificação eficiente de equipamentos de proteção individual. A contínua exploração e implementação dessas técnicas visam aprimorar a eficácia dos sistemas de detecção, contribuindo para a promoção de ambientes mais seguros e protegidos.

4. DESENVOLVIMENTO

Neste capítulo, serão abordados as tecnologias empregadas, com um foco específico no modelo YOLO, suas métricas de avaliação e a escolha do modelo mais adequado. Também será detalhada a seleção das classes de objetos, com a justificativa para cada escolha, além do desenvolvimento e treinamento do dataset, fundamental para a implementação do projeto. Por fim, explicarei a decisão de optar por Django em vez de Flask para o desenvolvimento da aplicação.

4.1. TECNOLOGIAS UTILIZADAS

Neste projeto, foi utilizado o modelo YOLOv8, treinado na versão gratuita do Google Colab, que oferece uma GPU Tesla T4 com restrições de tempo e alocação diária. Para testar os modelos, foram realizados quatro treinamentos: modelos pequenos foram treinados por 32 épocas, e modelos grandes foram treinados por 60 épocas, utilizando um conjunto de 2.000 imagens. Após a seleção do modelo ideal, o dataset foi desenvolvido e, em sua versão final, contém 6.346 imagens, treinadas em 32, 37, 42 e 47 épocas, respectivamente.

O projeto foi desenvolvido em um notebook com processador, Intel i5 (12ª geração), 16 GB de RAM DDR4, placa integrada Iris Xe e sistema operacional Linux. A ferramenta Roboflow foi essencial para o gerenciamento das imagens, permitindo controle de versões, rotulagem e aplicação de métodos de pré e pós-processamento e nuvem.

4.2. CLASSES DE OBJETOS

As classes de objetos estão sendo seguindo o conforme descrito na NR 06, que estabelece os requisitos para aprovação para utilização de EPIs (Ministério do Trabalho, 2022) e a NBR: 15.292 que embora o colete reflexivo não seja considerado um EPI, ele é útil sendo classificado como uma vestimenta de alta visibilidade (3M, 2013). Abaixo os equipamentos que serão utilizados e suas respectivas classes:

- *Capacete*: hard_hat
- *Óculos***:** safety_glasses

- *Protetor Auditivo*: ear_protection
- *Respirador Purificador de ar não motorizado:* mask
- *Luvas*: safety-gloves
- *Calçados*: safety-boots
- *Colete Refletivo*: vest

Para caso da falta de utilização destes equipamentos as seguintes classes foram rotuladas:

- *Sem colete*: no-vest
- Sem capacete: no-helmet
- *Sem Luva*: no-gloves
- *Sem Calçado*: no-safety-boots

Formando assim onze classes distintas para a detecção entre proteção e perigo. Sendo definidas como prioritárias as classes: sem colete, sem capacete, capacete e colete.

4.3. YOLO

Os primeiros estudos realizados sobre o YOLOv8 foram fundamentais para o desenvolvimento do projeto, focando na seleção do modelo adequado antes da criação do dataset "FEMA-Data". Para o treinamento dos modelos, foi utilizado um conjunto de dados público, o "PPE DETECTION Dataset", contendo 2000 imagens e 9 classes, disponível no Roboflow Universe para testes iniciais e análise. Foi elaborada uma tabela comparativa com os quatro principais modelos: Nano, Small, Medium e Large. Enquanto as versões mais leves, Nano e Small, foram treinadas por 60 épocas, as versões mais pesadas foram treinadas por 32 épocas. Essa abordagem diferenciada no número de épocas visou compensar a necessidade dos modelos mais leves de mais tempo de treinamento para alcançar um desempenho semelhante ao dos modelos mais pesados, facilitando assim uma avaliação de custo-benefício.

Algumas métricas de avaliação não são apenas importantes para a YOLO, mas são amplamente aplicadas por diferentes modelos de detecção. Conforme a

documentação da Ultralytics (2023) a utilização de Average Precison (AP) considera tanto a precisão (precision) quanto a recuperação (recall) do modelo, sendo calculada pela área sob a curva que relaciona ambas a cada classe e a Intersection over Union (IoU) sendo usada para avaliar quão bem uma predição corresponde ao objeto real.

Com base na documentação da Ultralytics (2023), se fornece informações importantes para o desempenho de um modelo em relação à sua capacidade de prever corretamente as bounding boxes ao redor dos objetos em uma imagem, sendo necessário a compreensão detalhada de cada componente a seguir:

- *P (Precision):* Refere-se à acurácia das detecções de objetos, indicando a proporção de detecções corretas em relação ao total de detecções realizadas ou de forma mais simples.
- *R (Recall):* Mede a capacidade do modelo de identificar todas as instâncias de um objeto presente nas imagens, ou seja, sua sensibilidade.
- *mAP50:* Calcula a média da AP para todas as classes no dataset, mas considera apenas detecções com um IoU ao menos 0,50. Ou seja, para cada classe, a AP é calculada considerando apenas as detecções que têm um IoU de pelo menos 0,50 com as caixas verdadeiras. A mAP50 é então a média dessas APs para todas as classes.
- *mAP50-95:* Representa uma versão mais rigorosa da mAP50. Ela calcula a média da AP em vários limiares de IoU, que variam de 0,50 a 0,95, em intervalos de 0,05. Dando uma visão mais completa do desempenho do modelo em diferentes níveis de sobreposição.
- *Interseção sobre União (IoU)*: Como pode ser visto matematicamente na figura 6 a Interseção em União é uma métrica que quantifica a sobreposição entre uma caixa delimitadora predita e uma caixa delimitadora de verdade terrestre. Ela desempenha um papel fundamental na avaliação da precisão da localização de objetos, ajudando a medir quão bem a previsão do modelo corresponde à realidade.

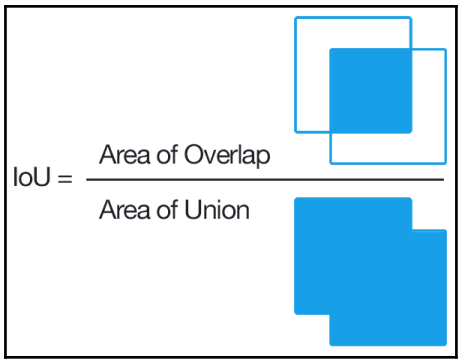

Figura 6: Fórmula do Cálculo do IoU
Fonte: Rosebrock, 2016

Na Tabela 1, os modelos Nano, Small, Medium e Large foram avaliados com base nas métricas Precision, Recall, mAP50 e mAP50-95. Embora o modelo Small tenha a menor precisão, ele se destaca em mAP50-95, quase alcançando o desempenho dos modelos Medium e Large, mesmo com menos épocas de treinamento.

Modelos	Épocas	Imagens	P (Precision)	R (Recal)	mAP50	mAP50-95
YoloV8n	60	2000	0.717	0.469	0.508	0.265
YoloV8s	60	2000	0.599	0.536	0.562	0.300
YoloV8m	32	2000	0.637	0.537	0.582	0.311
YoloV8l	32	2000	0.674	0.549	0.603	0.318

Tabela 1: Tabela de comparação entre os modelos
Fonte: Autoria própria, 2024

Assim, a métrica mAP50-95 foi escolhida porque valores elevados indicam melhor desempenho em vários limiares de IoU, refletindo maior precisão e robustez do modelo em condições de detecção variadas. A Figura 7 faz uma comparação entre os modelos demonstrando sua eficiência.

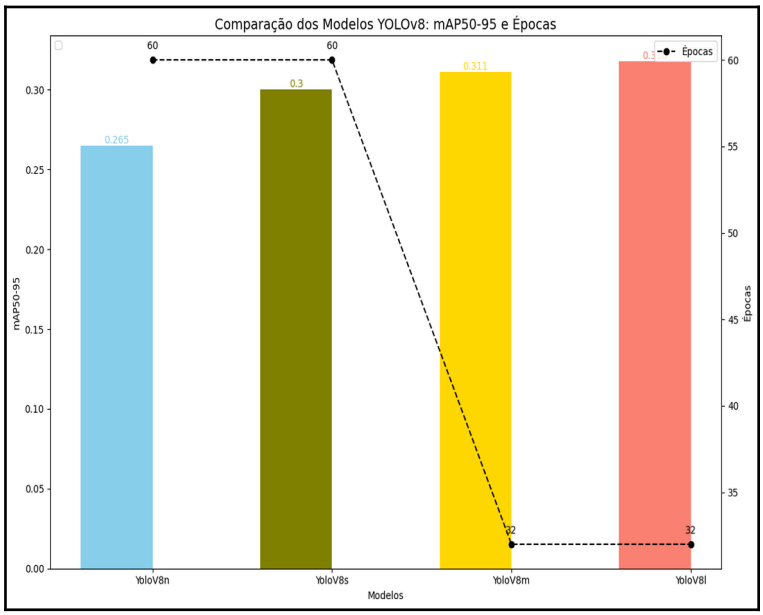

Figura 7: Comparação entre os modelos e suas respectivas épocas
Fonte: Autoria própria, 2024

Na Figura 8, é ilustrado como o modelo Small realizou o treinamento em cada imagem e a probabilidade de detecção dos objetos. Esses objetos foram classificados em uma escala de 0,0 a 1,0, onde uma classificação mais próxima de 1,0 indica uma maior probabilidade de acerto naquele treinamento, ou seja, quanto maior a pontuação, mais confiável é a detecção realizada pelo modelo.

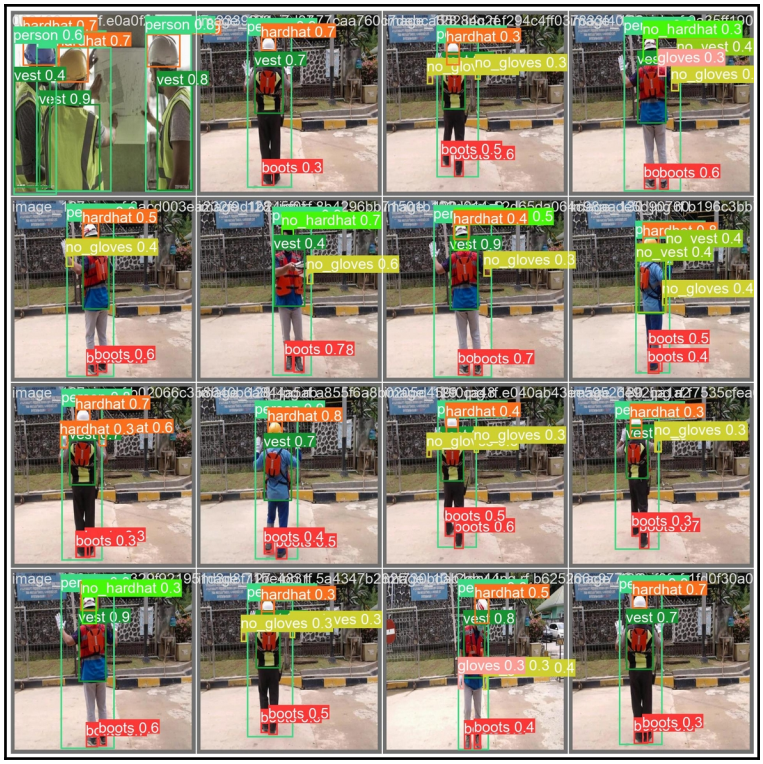

Figura 8: Exemplo de objetos detectados em treinamento
Fonte: Autoria própria, 2024

No dataset utilizado para os testes, já é demonstrada a dificuldade de detecção de objetos pequenos, como botinas e mãos com ou sem luvas. Isso demonstra a necessidade de maior atenção ao desenvolver o dataset final, garantindo que haja uma quantidade adequada de exemplos desses objetos para melhorar a precisão do modelo na detecção desses casos específicos.

4.4. DESENVOLVIMENTO DO DATASET

Um dos principais componentes do projeto foi a criação do conjunto de dados 'FEMA-Data' na plataforma Roboflow, utilizando imagens públicas para compor o dataset e evitar problemas de copyright. Essas imagens passaram por um processo de rotulação descrito na Seção 4.2, onde as classes são denominadas e anotadas, conforme demonstrado na Figura 9.

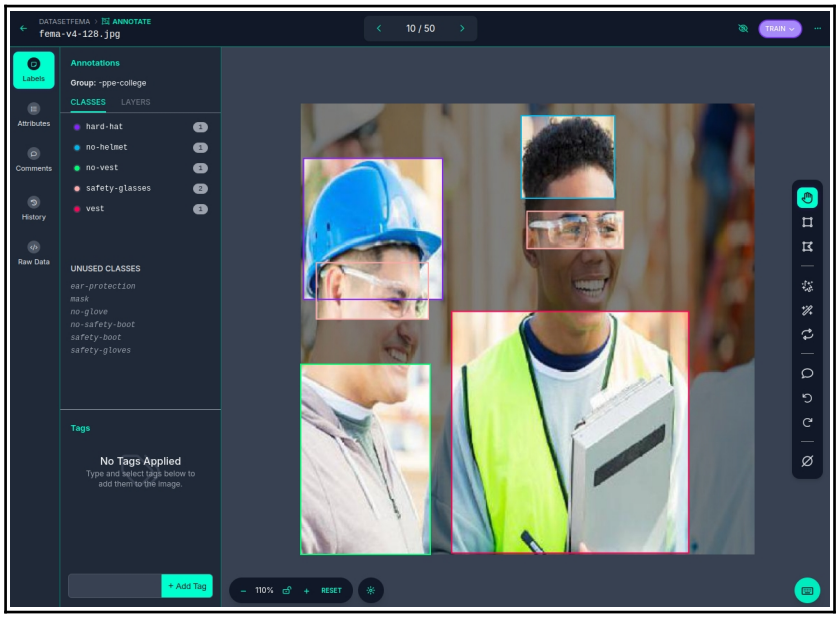

Figura 9: Exemplo de anotação na Roboflow
Fonte: Flôres, 2023

A primeira versão do dataset continha 1.210 imagens, que foram inicialmente rotuladas para descrever seu conteúdo conforme as classes de objetos definidas na seção 4.2. As imagens passaram por um pré-processamento que incluiu técnicas como redimensionamento, anotação e normalização para padronizá-las conforme o modelo a ser treinado.

Após o pré-processamento, as imagens foram divididas em conjuntos de treinamento, validação e teste, garantindo que a proporção de amostras de cada classe fosse mantida em todos os subconjuntos. Apenas no segundo treinamento, foram

aplicadas técnicas de augmentations às imagens do conjunto de treinamento, criadas por pesquisadores do Google para usar apenas o aumento da caixa delimitadora para criar dados ideais, criando melhorias sistêmicas (Zoph, 2019).

O dataset final, com as imagens pré-processadas e aumentadas, foi então compilado seguindo a pipeline demonstrada na figura 10.

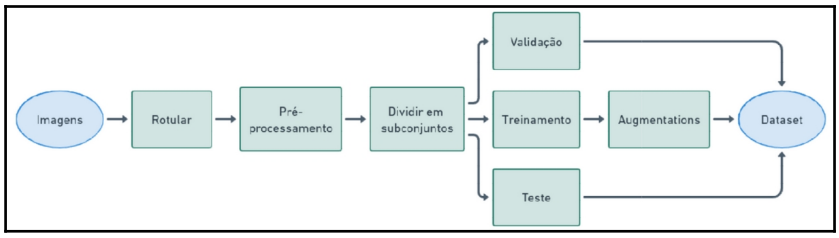

Figura 10: Pipeline de um dataset
Fonte: Flôres, 2023

No primeiro treinamento utilizando a plataforma Google Colab, a classe hard-hat teve um bom resultado de quase 0,80, enquanto a classe no-helmet apresentou um desempenho insatisfatória com apenas 0,39, com dificuldade em detectar e frequentemente confundindo cabelos com capacetes. A detecção da classe no-vest foi boa, mas necessitava de mais treinamento, com o YOLO reconhecendo apenas camisas vistas na vertical e confundindo camisas amarelas com a classe vest. As classes safety-gloves e safety-glasses tiveram quase nenhuma detecção, como esperado, mas a classe no-glove obteve alguns resultados quando o objeto estava sendo segurado. A classe mask foi confundida com hard-hat, sendo testada com uma máscara azul. As classes safety-boot, no-safety-boot e ear-protection não foram testadas.

No segundo treinamento, houve melhorias significativas na detecção das classes ear-protection, mask e safety-glasses. A detecção de hard-hat e no-vest mostrou uma redução no desempenho, causando preocupação por serem as classes principais como descrito na seção 4.2. A detecção de no-glove e safety-boots permaneceu consistente, enquanto a classe no-safety-boot teve um desempenho ruim em ambas as versões. O novo dataset incluiu augmentations como inversão da imagem, rotação (entre -15º e +15º) e variação de exposição (-10% e +10%). No geral, o segundo treinamento trouxe avanços para algumas classes e redução de desempenho para as principais, refletindo uma eficácia variada entre as categorias, conforme demonstrado na figura 11 utilizando métricas AP.

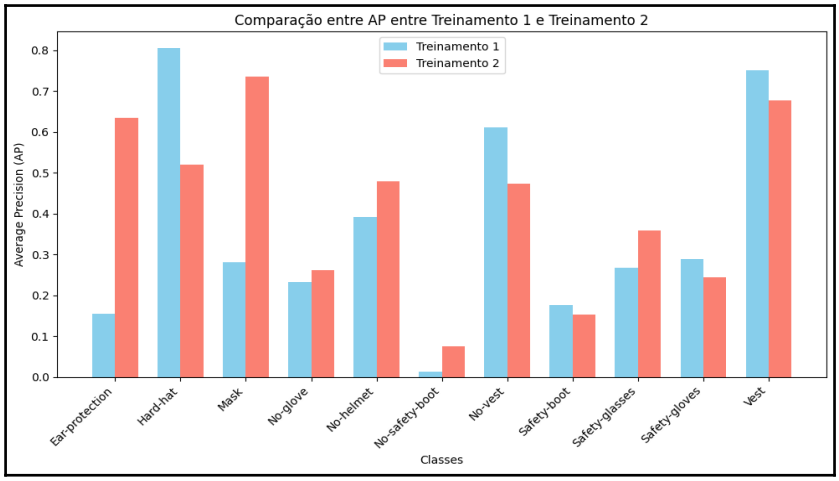

Figura 11: Comparação entre as duas primeiras versões
Fonte: Autoria própria, 2024

Inicialmente, considerou-se remover classes com baixo desempenho, como safety-gloves e safety-boot, mas optou-se por mantê-las para tentar melhorar sua precisão, mesmo que seja pelo aumento por número de épocas. O dataset final tem 2.727 imagens, expandidas para 6.346 imagens com 17.407 anotações após pré-processamento e augmentations, sendo apenas mantido a inversão horizontal da imagem, resultando em uma média de 6,4 anotações por imagem. As imagens foram distribuídas da seguinte forma: 5.515 imagens (87%) para treinamento, 411 imagens (6%) para validação e 420 imagens (7%) para teste. A ideia original era utilizar 80% das imagens para treinamento, 10% para validação e 10% para testes. No entanto, a plataforma Roboflow impôs algumas limitações, pois o pré-processamento e as técnicas de augmentação só podem ser aplicadas após a definição dessas distribuições. Como resultado, a distribuição dos dados foi ajustada, devido ao quase triplicamento das imagens originais e os valores podem ser alterados quando o modelo é compactado e preparado para uso.

4.5. TREINAMENTO DO DATASET

Após a conclusão do dataset e a escolha do modelo Small, foi realizado um treinamento com 32 épocas e comparado com os resultados dos dois primeiros treinos e a versão final. Conforme mostrado na figura 12, observou-se uma queda pequena queda nas classes ear-protection e mask. As classes no-glove, no-safety-boot apresentaram desempenhos semelhantes em todas as versões. Por outro lado, houve um pequeno aumento na detecção das classes hard-hat, no-helmet, safety-glasses e safety-gloves.

Além disso, as classes vest e no-vest mostraram resultados equivalentes em todas as versões.

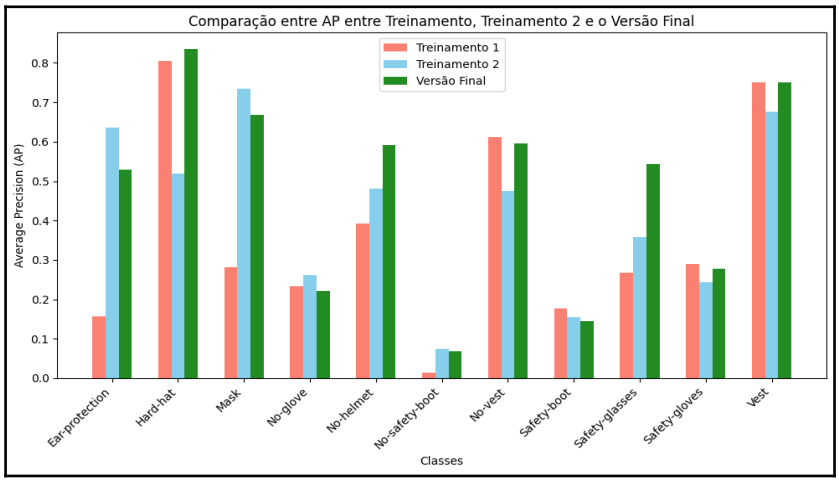

Figura 12: Comparação entre os dois primeiros teste e a versão final
Fonte: Autoria própria, 2024

Esse resultado já é o suficiente para a utilização na aplicação, pois as classes principais, como hard-hat, no-helmet, vest e no-vest, apresentaram resultados satisfatórios.

Conforme descrito na seção 4.1 a continuidade dos treinamentos foram realizados com as épocas restantes, 37, 42 e 47 épocas para medir a capacidade de detecção das classes no dataset. Conforme as épocas iam aumentando o tempo de treinamento também, chegando no final a quase 6 horas para conclusão, outro fator foi considerado para qual treinamento a aplicação utilizaria é o conceito de overfitting e underfitting, no caso de overfitting é quando o modelo aprende tão bem que apenas acaba fazendo predições e a sua precisão acaba sendo prejudicada e o underfitting é quando o modelo não treinou por tempo suficiente e seus inputs não são significantes o suficiente para determinar uma relação entre as variáveis de input e output (IBM, 2024).

A tabela 2 demonstra os resultados do treinamento usando as mesmas métricas para a escolha do modelo. Os resultados revelam que o modelo treinado com menos épocas apresenta o pior desempenho em todas as métricas. Por outro lado, o treinamento com 37 épocas apresenta a melhor precisão, mas a pior recall. O treinamento com 42 épocas exibe a segunda pior precisão, um recall igual à do modelo de 32 épocas, e uma mAP50-95 baixa. O modelo treinado por 47 épocas, apesar de ter uma precisão menor, se destaca em outras métricas e, portanto, foi escolhido como o modelo para a aplicação devido ao seu desempenho geral superior.

Épocas	Imagens	P (Precision)	R (Recal)	mAP50	mAP50-95
32	5.515	0.556	0.491	0.477	0.278
37	5.515	0.605	0.476	0.492	0.289
42	5.515	0.561	0.491	0.496	0.287
47	5.515	0.573	0.502	0.498	0.295

Tabela 2: Tabela de comparação entre os treinamentos
Fonte: Autoria própria, 2024

Para confirmar que não houve overfitting, a Figura 13 mostra os resultados do treinamento do modelo com 32, 37, 42 e 47 épocas. A análise revela que não haver de overfitting até a 47ª época, já que tanto as perdas de treinamento quanto as perdas de validação estão diminuindo. Além disso, as métricas de desempenho estão em ascensão, indicando uma melhoria contínua na capacidade do modelo em generalizar e detectar objetos.

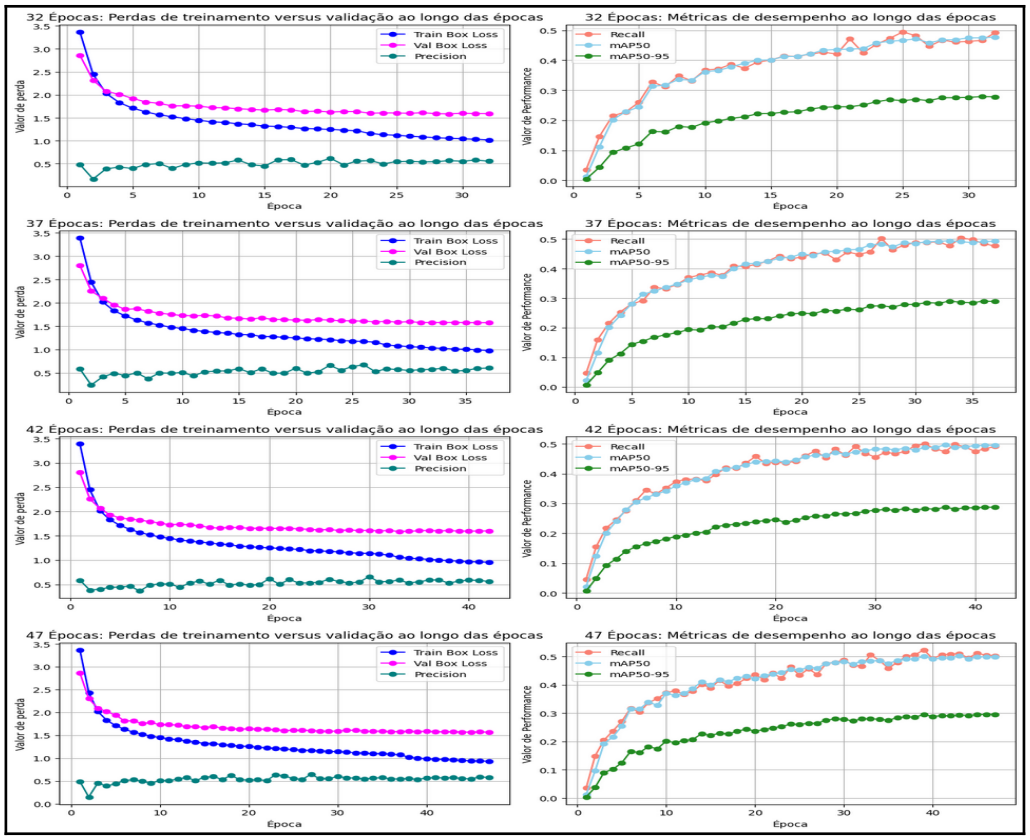

Figura 13: Treino e validação e perfomance ao longo das épocas
Fonte: Autoria própria, 2024

No geral, apesar de alguns treinamentos apresentarem valores baixos, foi demonstrado que o aprendizado está estável. Isso sugere que um número maior de épocas seria necessário para identificar qualquer indício de overfitting. As métricas de desempenho e as perdas observadas indicam que o modelo continua a melhorar até o ponto observado, sem sinais claros de overfitting.

4.6. DJANGO

Django e Flask são dois frameworks Open-Source para a utilização de Python, e, apesar de ambos terem arquiteturas relativamente simples, suas abordagens são diferentes. Django segue o padrão MVT (Model, Template, View), enquanto Flask é um microframework focado no desenvolvimento de aplicações web minimalista (SIMPLILEARN, 2021). As principais vantagens do Flask são sua leveza, simplicidade e flexibilidade, tornando-o ideal para projetos pequenos e médios, especialmente em aplicações single-page, onde a seleção de componentes necessários pode ser feita conforme a demanda, oferecendo ao desenvolvedor maior liberdade e controle. No entanto, apesar dessas vantagens, Django, com sua filosofia "batteries-included" (10. Brief Tour of the Standard Library, [s.d.]), oferece uma estrutura pronta para usar como jsons, xml e sql e com recursos como gerenciamento de pacotes e autenticação (contrib packages | Django documentation, 2024), tornando-o a escolha ideal para este projeto.

Como descrito acima, o Django oferece a arquitetura MVT, cuja compreensão é essencial para a construção da aplicação. Embora os frameworks sejam amplamente conhecidos por suas arquiteturas baseadas em MVC (Model, View, Controller), o Django adota um padrão diferente, substituindo o Controller pelo Template. Em Django, a “View” é responsável pela lógica que seleciona os dados que serão representados, enquanto o “Template” lida com a exibição desses dados. Em contraste, no MVC, é o “Controller” que cumpre essa função. De acordo com Buczyński (2023), podemos resumir a arquitetura MVT da seguinte forma:

- **Model:** Representa os dados e a estrutura, geralmente mapeados para um banco de dados;
- **View:** Trata as requisições do usuário e suas respostas, atuando como uma ponte entre o Model e o Template;

- **Template:** Define como a interface de usuário deve se parecer, utilizando arquivos HTML.

Por outro lado, em MVC:

- **Model:** Similar ao MVT, representa os dados da aplicação;
- **View:** Trata da parte gráfica da interface, renderizando o Model;
- **Controller:** Controla a lógica de negócios e gerência a interação entre o Model e a View.

Com este modelo de arquitetura podemos criar padrões de projetos e melhores práticas de forma simples e dinâmica, como demonstrado na figura 14.

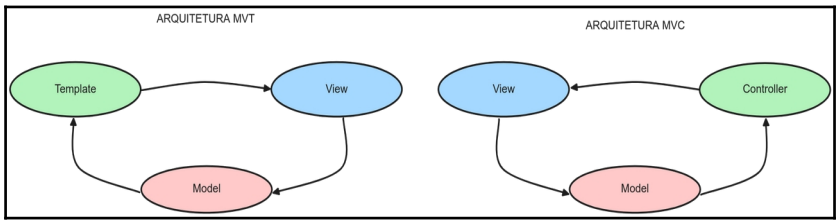

Figura 14: Arquitetura MVT
Fonte: Buczyński, 2023 (Adaptada pelo autor)

4.6.1. VENV

Utilizado o módulo venv de Python, que cria um ambiente de virtualização de forma leve, no qual pacotes podem ser instalados de forma independente para cada projeto, esse ambiente é criado no topo de uma aplicação python existe e isolada dos pacotes do ambiente base (venv — Creation of virtual environments — Python 3.8.1 documentation, 2019). Essa estrutura permite definir apenas os pacotes necessários sejam instalados como demonstrado na figura 15.

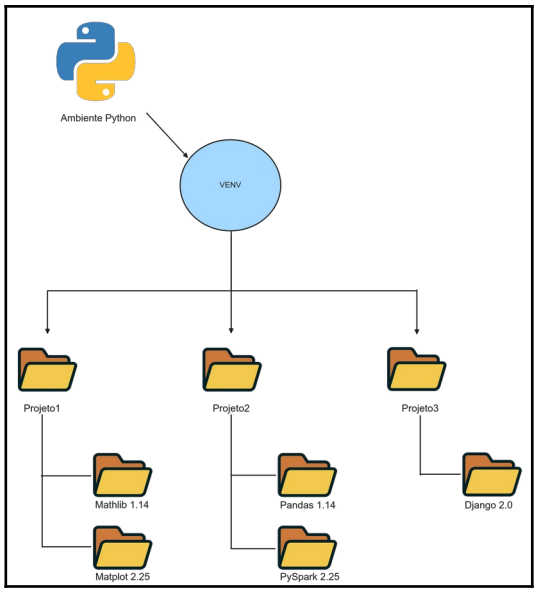

Figura 15: Ambiente de virtualização de Python
Fonte: Autoria própria, 2024

Para a instalação deste ambiente, utilizamos o comando no terminal Linux “python -m venv /caminho/para/novo/virtual/ambiente”. Após a execução desse comando, o ambiente pode ser iniciado com o comando “source venv/bin/activate”. No terminal, a estrutura será exibida como “(venv) usuario@computador: pasta/onde/esta/o/projeto”. No caso deste projeto além dos pacotes padrões que vem instalados no Django como channels para permitir a comunicação de websockets, forma instalados:

- *OpenCV-Python*: Biblioteca de visão computacional;
- *Numpy*: Biblioteca para computação numérica;
- *Nvidia*: Mesmo não sendo utilizado uma placa Nvidia, foi necessário a instalação destes pacotes para o funcionamento da YOLO sendo o CUDA e cuDNN para aceleração de GPU e suporte de operações deep learning;
- *PyMySQL*: Connector para MySQL;
- *Ultralytics*: Para executar modelos YOLO;
- *Torch e Torchvision*: Biblioteca de Deep Learning (PyTorch) e extensão para visão computacional.

4.6.2. MYSQL

A decisão de migrar do SQLite para o MySQL se baseia em razões semelhantes à escolha entre Django e Flask: escalabilidade. Embora o SQLite seja mais leve e de código

aberto, o MySQL oferece suporte para múltiplos usuários simultâneos, maior robustez em termos de segurança e é mais adequado para projetos que utilizam framework web, onde a escalabilidade e o gerenciamento de grandes volumes de dados são fundamentais. (What to Use – SQLite or MySQL? In-Depth Analysis For Your Convenience, 2019).

4.6.3. INTEGRAÇÃO DJANGO E YOLO

Para que a integração funcionar é necessário seguir alguns passos. O primeiro sendo o carregamento do modelo, conforme demonstrado na figura 16.

```
20    model_path = os.path.join(os.path.dirname(__file__), '..', 'models', 'best.pt')
21    model = YOLO(model_path)
```

Figura 16: Carregamento do modelo
Fonte: Autoria própria, 2024

Após é definido a inicialização das variáveis conforme sendo também necessário um dicionário que armazena a contagem de detecções para cada classe para ser utilizado em um gráfico para contagem de objetos detectados, sendo inicializado com zero para cada classe, demonstrado na figura 17.

```
23  latest_detections = []
24  classNames = ['Protecao de Ouvido', 'Capacete', 'Mascara', 'Sem Luva', 'Sem Capacete', 'Sem Botina', 'Sem Colete Refletivo', 'Botina',
25                'Oculos de Protecao', 'Luvas de Protecao', 'Colete Refletivo']
26
27  detection_count = {class_name: 0 for class_name in classNames}
```

Figura 17: Inicialização de váriveis
Fonte: Autoria própria, 2024

Para as funções de manipulações de requisições foram definidos duas, a primeira sendo a detect, para a renderização da página html e video_feed para uma resposta HTTP que envia frames dos vídeos gerados pela função video_feed_generator. O conteúdo é enviado no formato de múltiplos segmentos com a boundary frame. A imagem 18 demonstra essas duas funções.

```
44  def detect(request):
45      return render(request, 'detect.html')
46
47  def video_feed(request):
48      return StreamingHttpResponse(video_feed_generator(), content_type='multipart/x-mixed-replace; boundary=frame')
```

Figura 18: Funções manipulação de requisições
Fonte: Autoria própria, 2024

Para a geração de vídeo, foi definida a função video_feed_generator onde é responsável por capturar e processar o feed do vídeo, começando criando um objeto VideoCapture do OpenCV para capturar vídeo da câmera com a saída 0, ou seja, webcam, como demonstrado na figura 19.

```python
def video_feed_generator():
    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        print("Error: Não foi possível abrir a câmera.")
        return

```

Figura 19: Inicio da função
Fonte: Autoria própria, 2024

Então é definido as variáveis globais e locais, onde latest_detections lista as últimas atualizações, frame index o índice do quadro atual, batch_size números de quadros a serem processados em um lote e frame_buffer para armazenar os quadros de forma temporaria, conforme é demonstrado na figura 20.

```python
    global latest_detections
    latest_detections = []
    frame_index = 0  # Inicializa o index de Frame

    batch_size = 1  # Processa o frame um de cada vez
    frame_buffer = []
```

Figura 20: Definições de váriaveis globais
Fonte: Autoria própria, 2024

Inicia um loop para capturar e processar o vídeo de forma contínua registrando o tempo de início para calcular o tempo de inferência, para caso o frame falhe será exibido uma mensagem de erro e sai do loop. Em frame_buffer adiciona quadro capturados ao buffer de dados e verificá-se se o buffer contém pelo menos o tamanho do lote, se sim, remove o quadro mais antigo. E inicia a listas para armazenar informações sobre caixas delimitadoras e IDs conforme mostrado na figura 21.

```python
    while True:
        start_time = time.time()

        success, img = cap.read()
        if not success:
            print("Error: Falha ao capturar imagem.")
            break

        frame_buffer.append(img)

        if len(frame_buffer) >= batch_size:
            # Processa apenas um frame
            img = frame_buffer.pop(0)  # Usa e Remove os frames mais velhos do buffer
            results = model([img], stream=True)
            detection_count = {}  # Incializa a detecção em cada frame

            boxes = []
            confidences = []
            class_ids = []
```

Figura 21: Início da estrutura de loop
Fonte: Autoria própria, 2024

Na figura 22 é demonstrado como é iterado os resultados da detecção e extração das coordenadas e caixas delimitadoras, a confiança e a classe detectada. Se a confiança for maior que 0,5, adiciona a caixa e o ID da classe às listas correspondentes, sendo atualizado a lista latest_detections e o dicionário detection_count.

```
            for r in results:
                for box in r.boxes:
                    x1, y1, x2, y2 = box.xyxy[0]
                    conf = math.ceil((box.conf[0] * 100)) / 100
                    cls = int(box.cls[0])

                    if conf > 0.5:
                        # Converte as coordenadas em float
                        boxes.append([float(x1), float(y1), float(x2), float(y2)])
                        confidences.append(float(conf))
                        class_ids.append(cls)

                        currentClass = classNames[cls]
                        latest_detections.append(currentClass)

                        if currentClass in detection_count:
                            detection_count[currentClass] += 1
                        else:
                            detection_count[currentClass] = 1

```

Figura 22: Processo de extração de coordenadas e identificação de classes
Fonte: Autoria própria, 2024

Em seguida, conforme demonstrado na figura 23, é processado cada caixa e obtém suas coordenadas, confiaça e classes. Também é definido as cores das caixas detectada. Se a classe for listada como alerta, é definida como vermelho e um aviso sonoro é emitido. Caso contrário, define-se verde dependendo da classe.

```
        # Aplica Non-Maximum Suppression
        indices = cv2.dnn.NMSBoxes(boxes, confidences, score_threshold=0.5, nms_threshold=0.4)

        if len(indices) > 0:
            indices = indices.flatten()  # Achata os indices de Array
            for i in indices:
                x1, y1, x2, y2 = boxes[i]
                conf = confidences[i]
                cls = class_ids[i]
                currentClass = classNames[cls]

                if currentClass in ['Sem Capacete', 'Sem Luva', 'Sem Colete Refletivo', 'Sem Botina']:
                    myColor = (0, 0, 255)  # Vermelho para alertas
                    cv2.putText(img, 'ALERTA', (11, 100), 0, 1, [0, 0, 255], thickness=3, lineType=cv2.LINE_AA)
                    play_sound()  # Ativa o som
                elif currentClass in ['Protecao de Ouvido', 'Capacete', 'Mascara', 'Botina', 'Oculos de Protecao', 'Colete Refletivo', 'Luvas de Protecao']:
                    myColor = (0, 255, 0)  # Verdes para seguro
                else:
                    myColor = (0, 0, 255)  # Azul para outros

                cv2.rectangle(img, (int(x1), int(y1)), (int(x2), int(y2)), myColor, 2)
                cvzone.putTextRect(img, f'{currentClass} {conf}',
                                   (max(0, int(x1)), max(35, int(y1))), scale=1, thickness=1, colorB=myColor,
                                   colorT=(0, 0, 0), colorR=myColor, offset=6)
```

Figura 23: Adição de buffer e caixas delimitadoras
Fonte: Autoria própria, 2024

Porém, a caixa ainda não foi desenhada, com a cv2.rectangle e cv2.putTextRect a caixa e o texto é criado e exibido conforme as linhas, em seguida o tempo de inferência é calculado e se obtém resolução do quadro, também criando uma legenda com o índice do quadro, resolução, contagem de detecções e tempo de inferência. São gerados dois yield um para o quadro codificado JPEG para streaming de vídeo e os dados de detecção JSON e ao final executado a garbage collector para liberar memória, conforme demonstrado na figura 24.

```
122
123                     cv2.rectangle(img, (int(x1), int(y1)), (int(x2), int(y2)), myColor, 2)
124                     cvzone.putTextRect(img, f'{currentClass} {conf}',
125                                        (max(0, int(x1)), max(35, int(y1))), scale=1, thickness=1, colorB=myColor,
126                                        colorT=(0, 0, 0), colorR=myColor, offset=6)
127
128             # Termina o timer e calcula o tempo de inferencia
129             inference_time = (time.time() - start_time) * 1000  # Tempo em ms
130
131             # Resolução do Frame
132             height, width = img.shape[:2]
133             resolution = f'{width}x{height}'      ga.mossini, 6 days ago
134
135             # Gera um String para as legendas
136             caption = f"{frame_index}: {resolution} " + ", ".join([f'{v} {k}' for k, v in detection_count.items()]) + f", {inference_time:.1f} ms"
137             cv2.putText(img, caption, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)
138
139             frame_index += 1  # Incrementa o frame de index
140
141             # Codifica os quadros para streaming
142             ret, buffer = cv2.imencode('.jpg', img)
143             frame = buffer.tobytes()
144
145             # Produza o quadro e os dados de detecção como JSON
146             yield (b'--frame\r\n'
147                    b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n' +
148                    b'--detection-data\r\n'
149                    b'Content-Type: application/json\r\n\r\n' +
150                    json.dumps(detection_count).encode() + b'\r\n\r\n')
151
152             yield (b'--frame\r\n'
153                    b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
154
155         gc.collect()
```

Figura 24: Desenho das caixas, cálculo do tempo e codificação de quadros
Fonte: Autoria própria, 2024

5. RESULTADO DA APLICAÇÃO

Neste capítulo, são apresentados a aplicação final e os resultados obtidos, as principais funcionalidades implementadas e desempenho alcançado ao longo do desenvolvimento do projeto.

5.1. AUTENTICAÇÃO

Para criar um sistema que seja seguro, principalmente considerando que a câmera é a principal funcionalidade, surgiu a necessidade de autenticar os usuários. Isso foi implementado por um sistema de login conforme demonstrado na figura 25.

Figura 25: Página de login
Fonte: Autoria própria, 2024

Na figura 26, é demonstrado que, após a autenticação, o usuário é redirecionado a página inicial onde aparecerá e duas opções estarão disponíveis: Detecção, para o acesso a câmera e Usuários onde é feito o controle para cadastro, edição, exclusão e relatórios de usuários.

Figura 26: Página incial
Fonte: Autoria própria, 2024

Caso o usuário não seja autenticado, ou não ter nível de acesso necessário, ele irá redirecionar para uma página de erro, como demonstrado na figura 27.

Figura 27: Página de segurança
Fonte: Autoria própria, 2024

5.2. DETECÇÃO

Na página de detecção o usuário terá acesso ao botão para abrir câmera, e será. Conforme é mostrado na figura 28.

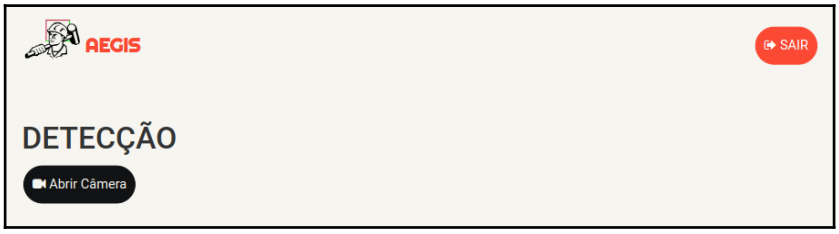

Figura 28: Página de detecção de objetos
Fonte: Autoria própria, 2024

Na figura 29 é demonstrado, quando a câmera é ativada pelo usuário, que visualizará a implementação de um botão para habilitar ou desabilitar os alertas sonoros. Na parte superior da interface será exibido informações sobre o frame atual, a resolução da imagem, o tempo de resposta em ms, um alerta sonoro e visual e a quantidade de objetos detectados.

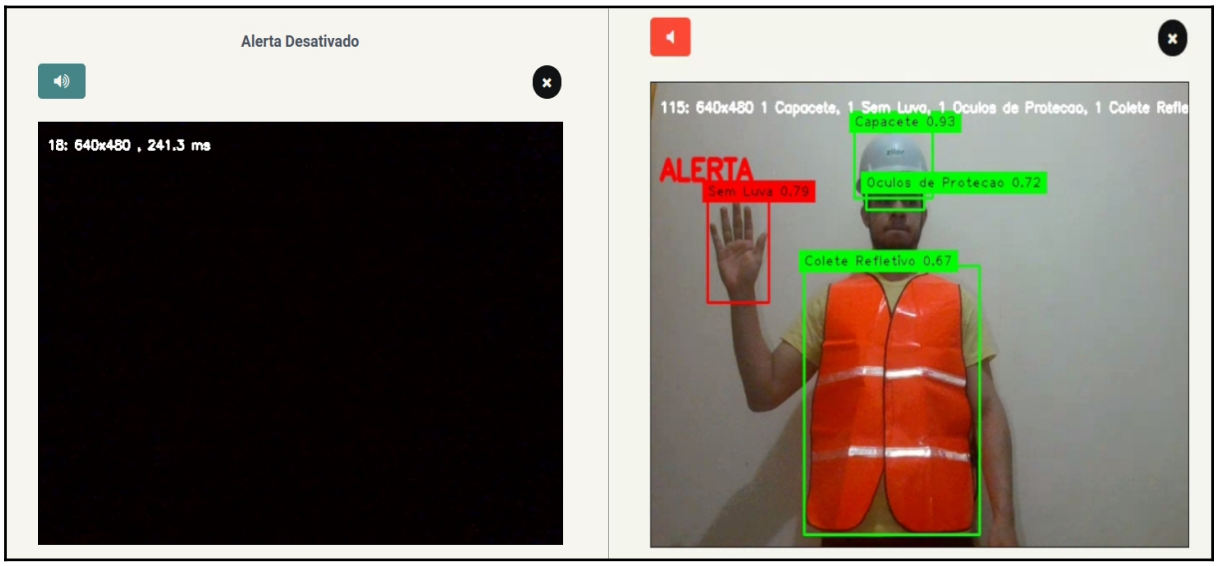

Figura 29: Exemplos de câmeras
Fonte: Autoria própria, 2024

Também será exibido abaixo um gráfico e tabela com o total de detecções per frame, sendo possível averiguar a quantidade de detecções foram feitas enquanto a câmera esteve aberta, como demonstrado na figura 30.

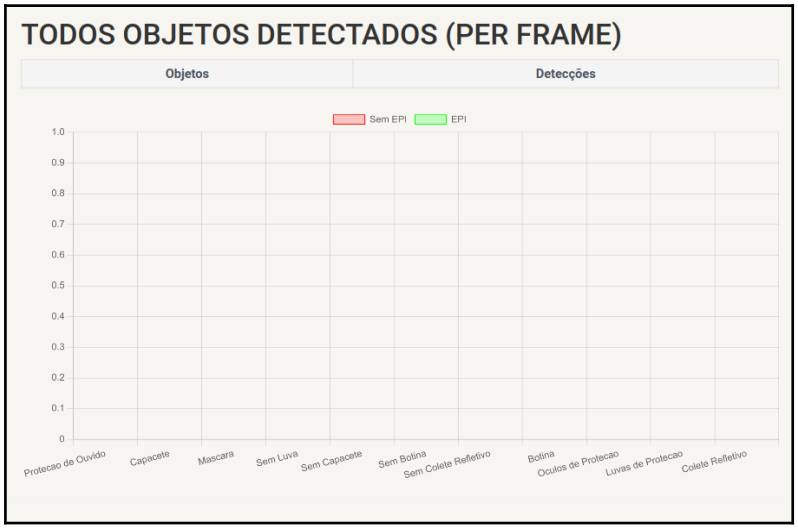

Figura 30: Gráfico de detecção de objetos per frame
Fonte: Autoria própria, 2024

5.3. USUÁRIOS

Na página de usuários, são visualizados resumidamente todos os usuários cadastrados no sistema, incluindo seu ID, nome completo, status (ativo ou inativo), data de criação e último login. Além disso, a interface oferece opções para visualização detalhada, download de relatórios de todos os usuários, alteração de informações e exclusão de registros, conforme demonstrado na Figura 31.

Figura 31: Página de usuários
Fonte: Autoria própria, 2024

Também é solicitado informações como CPF, Data de Nascimento, E-Mail, Funções (Administrador ou Usuário). Caso o usuário queria que estes dados sejam deletados, todas as informações e impossíveis de se recuperar.

5.4. RESULTADOS

No início da aplicação, um desafio crucial foi melhorar a eficiência no processamento de frames. Inicialmente, considerou-se o uso da API da Roboflow, com o modelo operando na nuvem. No entanto, devido à velocidade de internet, o tempo de resposta frequentemente atingia no mínimo 2000 ms. Optou-se então por utilizar o modelo treinado localmente no Google Colab, o que resultou em uma melhoria no tempo de resposta para 1204 ms por frame. Apesar dessa melhora, o tempo ainda era elevado. Para otimizar, implementou-se um gerenciamento de memória mais eficiente, incluindo o uso do garbage collector e a aplicação de NMS (Non-Maximum Suppression) para eliminar buffers desnecessários. Essas alterações reduziram o tempo de processamento para 241 ms por frame.

Com esse resultado, ficou evidente que, para o sistema funcionar de efetiva em ambientes de perigo, é crucial ter um servidor com internet local com internet dedicada, especialmente se o objetivo é evitar altas taxas de respostas. Para aplicações em ambientes cotidianos, o testado, é necessário um computador com hardware potente. Uma placa integrada, por exemplo, não se mostra suficiente para aumentar as taxas de frames, sendo que a internet desempenha uma parte do processo e é totalmente irrelevante quando se trata de modelos executados localmente.

O acesso ao usuário se mostrou eficaz, prevenindo a possibilidade de acesso não autorizado à câmera, o que poderia representar uma vulnerabilidade grave no sistema. O Django provou ser uma ferramenta eficiente para implementar essa segurança, garantindo que apenas usuários autenticados e com permissões adequadas possam acessar determinadas funcionalidades críticas do sistema.

6. CONCLUSÃO

Neste trabalho, foi apresentado o desenvolvimento e a implementação de um sistema de detecção e segurança baseado em visão computacional para um ambiente monitorado em tempo real. A metodologia abordou a criação de uma solução para garantir a eficiência e a segurança em um sistema de monitoramento com câmera, incluindo a otimização do processamento de frames e a gestão de dados detectados.

Durante o desenvolvimento, ficou claro que a eficiência do sistema é altamente dependente da infraestrutura de hardware e da qualidade da conexão de internet. Em ambientes críticos, como os industriais, é recomendado ter uma infraestrutura dedicada, enquanto para ambientes cotidianos, a escolha de hardware de qualidade é crucial para garantir o desempenho esperado.

A implementação do sistema demonstrou que, embora o desenvolvimento de soluções de segurança em visão computacional possa ser desafiador, a adaptação e a otimização são fundamentais para alcançar resultados eficazes e a possibilidade de ser aplicado em ambientes industriais.

A conclusão deste trabalho, demonstrou uma solução para ambientes de monitoramento em tempo real, destacando a importância de escolhas tecnológicas adequadas e otimizações contínuas para garantir a eficácia do sistema.

6.1. TRABALHOS FUTUROS

Como trabalho futuro, devido ao alto custo, não foi implementado um servidor dedicado para aumentar a eficiência da aplicação. A proposta é realizar a implementação em nuvem, utilizando GPUs, o que tornaria o sistema mais robusto e, ao mesmo tempo, leve o suficiente para ser acessado a partir de qualquer computador ou dispositivo móvel. Portanto, desenvolver uma abordagem eficaz para a implementação em nuvem é essencial para otimizar o desempenho do software.

REFERÊNCIAS

DE SOUZA, L. C.; DE MELO, F. X. **A Importância do uso de EPI na prevenção de acidentes.** Diálogos Interdisciplinares, v. 9, n. 1, p. 200-215, 23 maio 2020.

EPIs desempenham papel fundamental na luta pela redução de acidentes de trabalho. Tribunal Superior do Trabalho, 27 jul 2021. Notícias Disponível em: https://www.tst.jus.br/-/epis-desempenham-papel-fundamental-na-luta-pela-redu %C3%A7%C3%A3o-de-acidentes-de-trabalho-1. Acesso em: 13 out. 2023.

DE LIMA, D. C. F.; DA CUNHA, J. C. **Inteligência artificial em equipamentos de proteção individual.** Brazilian Applied Science Review, [S. l.], v. 7, n. 1, p. 265–288, 2023. DOI: 10.34115/basrv7n1-013. Disponível em: https://ojs.brazilianjournals.com.br/ojs/index.php/BASR/article/view/57128. Acesso em: 16 oct. 2023.

BANOULA, Mayank. **What is Perceptron: A Begginers Guide for Perceptron.** SimpliLearn. Disponível em: https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron#perceptron. Acesso em: 17 mar 2024.

Data Science Academy. **Deep Learning Book**, 2022. Disponível em: <https://www.deeplearningbook.com.br/funcao-de-ativacao/>. Acesso em: 21 Mar. 2024.

HAYKIN, Simon. **Redes Neurais: Princípios e prática**, 2. Ed. Tradução de Paulo Martins Engel., Porto Alegre: Editora Bookman, 2001.

FLECK, Leandro et al. **Redes neurais artificiais: Princípios básicos**. Revista Eletrônica Científica Inovação e Tecnologia, v. 1, n. 13, p. 47-57, 2016.

FERNANDES, Marcelo Augusto Costa. **Redes neurais artificiais aplicadas à deteção inteligente de sinais.** p.37. Dissertação(mestrado) - Universidade Federal do Rio Grande do Norte, Natal, 1999.

ROSENBLATT, Frank. **The perceptron A perceiving and recognizing automaton (Project PARA).** Relatório Técnico – Cornell Aeronautical Laboratory, Inc. Buffalo, New York, 1957.

ROSENBLATT, Frank. **The Perceptron: A probabilistic model for information storage and organization in the brain.** Psychological Review, Vol. 63, No.6, p. 386-408, 23 abr 1958.

FURTADO, Maria Inês Vasconcellos. **Redes Neurais Artificiais: Uma abordagem para sala de aula**. Ponta Grossa: Editora Atena, 2019.

KARPATHY, Andrej, TODERICI, George, SHETTY, Sanketh, LEUNG, Thomas, SUKTHANKAR, Rahul, FEI-FEI, Li. **Large-Scale Video Classification with Convolutional Neural Networks.** 2014 IEEE Conference on Computer Vision and Pattern Recognition, jun. 2014.

KRIZHEVSKY, Alex, SUTSKEVER, Ilya, HINTON, Geoffrey E . **ImageNet Classification with Deep Convolutional Neural Networks**. Conference on Neural Information Processing Systems. 26ª; 2012; Lake Tahoe, Nevada, USA.

SERMANET, P. et al. OverFeat: **Integrated Recognition, Localization and Detection using Convolutional Networks**. 21 dez. 2013.

SERMANET, Pierre, CHINTALA, Soumith, LECUN, Yann. **Convolutional Neural Networks Applied to House Numbers Digit Classification**. arXiv (Cornell University), 17 abr. 2012.

LECUN, Yann, BENGIO, Yoshua. HINTON, Geoffrey. **Deep Learning**. Nature, v. 521, n. 7553, p. 436–444, maio 2015.

O'SHEA, Keiron, NASH, Ryan. **An Introduction to Convolutional Neural Networks**. arXiv (Cornell University), 26 nov. 2015.

ZAZO, Javier. PROTOPAPAS, Pavlos. **Exemples of Convolutional Neural Networks**. Disponível em: <https://scholar.harvard.edu/files/javierzazo/files/cs109b_asec3_notes_convnets.pdf>. Acesso em: 24 mar. 2024.

JAMES, Gareth, WITTEN, Daniela, HASTIE, Trevor, TIBSHIRANI, Robert et al. **An Introduction to Statistical Learning**. [s.l.] Springer Nature, 2023.

FIGUEIREDO, Raphael Silva de. **Perceptron Multicamadas com Backpropagation para Análise de Crédito**. Dissertação(pos-graduação) - Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

CUNHA, Cristiane Paim. CONTEUDISTA COORDENADORA. **Noções básicas de SST para pequenos negócios**. Disponivel em: https://www.escolavirtual.gov.br/, 1 jan. 2022. Acesso em: 28 mar. 2024.

ULTRALYTICS. **YOLO Performance Metrics**, 2023. Disponível em: <https://docs.ultralytics.com/guides/yolo-performance-metrics/#case-3>. Acesso em: 28 mar. 2024.

GALLO, Gionatan, DI RIENZO, Francesco, GARZELLI, Federico, DUCANGE, Pietro, VALLATI, Carlo. **A Smart System for Personal Protective Equipment Detection in Industrial Environments Based on Deep Learning at the Edge.** IEEE Access, v. 10, p. 110862–110878, 2022.

LI, Feng. ZHANG, Hao, XU, Huaizhe, LIU, Shilong, ZHANG, Lei, NI, Lionel M., SHUM, Heung-Yeung. **Mask DINO: Towards A Unified Transformer-based Framework for Object Detection and Segmentation**. Disponível em: <https://arxiv.org/abs/2206.02777>. Acesso em: 30 mar. 2024.

JOCHER, G.; CHAURASIA, A.; QIU, J. **YOLOv8 by Ultralytics**. Disponível em: <https://github.com/ultralytics/ultralytics>. Acesso 1 ago. 2024

ZOPH, B. et al. **Learning Data Augmentation Strategies for Object Detection.** arXiv (Cornell University), 26 jun. 2019.

FLÔRES, Douglas Souza. **Detecção de epis através de uma cnn**. *In*: FLÔRES, Douglas Souza. Detecção de epis através de uma cnn. 2023. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação com Ênfase em Engenharia da Computação) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2023. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/272129/001188438.pdf?sequence=1&isAllowed=y. Acesso em: 1 jul. 2024.

ROSEBROCK, A. **A visual equation for Intersection over Union (Jaccard Index).** 7 nov. 2016. Disponível em: <https://commons.wikimedia.org/wiki/File:Intersection_over_Union_-_visual_equation.png>. Acesso em: 6 ago. 2024.

IBM. **What is Overfitting? | IBM**. Disponível em: <https://www.ibm.com/topics/overfitting>. Acesso em: 6 ago. 2024.

SIMPLILEARN. **Django Vs. Flask: Understanding The Major Differences**. Disponível em: <https://www.simplilearn.com/flask-vs-django-article#:~:text=Flask%3A%20Ideal%20for%20smaller%20projects>. Acesso em: 7 ago. 2024.

BUCZYŃSKI, R. **MVT Architecture in Django: Introduction and Comparison with MVC**. Disponível em: <https://python.plainenglish.io/mvt-architecture-in-django-introduction-and-comparison-with-mvc-37cd617b542e>. Acesso em: 7 ago. 2024.

contrib packages | Django documentation. Disponível em: <https://docs.djangoproject.com/en/5.1/ref/contrib/>. Acesso em: 8 ago. 2024.

10. Brief Tour of the Standard Library. Disponível em: <https://docs.python.org/3/tutorial/stdlib.html#tut-batteries-included>. Acesso em: Acesso em: 8 ago. 2024.

What to Use – SQLite or MySQL? In-Depth Analysis For Your Convenience. Disponível em: <https://www.hostinger.com/tutorials/sqlite-vs-mysql-whats-the-difference/>. Acesso em: 8 ago. 2024.

venv — Creation of virtual environments — Python 3.8.1 documentation. Disponível em: <https://docs.python.org/3/library/venv.html>. Acesso em: Acesso em: 9 de ago. 2024.

BRASIL. Ministério do Trabalho. Portaria nº 2.175, 28 de julho de 2022. Aprova a NR 6 (Equipamentos de proteção individual – EPI). **Diário Oficial da União**, Brasília, DF, 27 dez. 2006. Disponivel em: https://www.gov.br/trabalho-e-emprego/pt-br/acesso-a-informacao/participacao-social/conselhos-e-orgaos-colegiados/comissao-tripartite-partitaria-permanente/arquivos/normas-regulamentadoras/nr-06-atualizada-2022-1.pdf. Acesso em: 01 set. 2024.

3M (ED.). **Perguntas e Respostas mais Frequentes Materiais retorrefletivos e a NBR 15292: Artigos Confeccionados – Vestimenta de Segurança de Alta visibilidade**. Disponível em: <https://multimedia.3m.com/mws/media/994340O/perguntas-frequentes-nbr-15292.pdf>. Acesso em: 5 set. 2024.