UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

BEATRIZ FRÉCCIA AMANTE

RASTREAMENTO DE PESSOAS POR TÉCNICAS DE APRENDIZAGEM DE MÁQUINA

GUARAPUAVA

2025

BEATRIZ FRÉCCIA AMANTE

RASTREAMENTO DE PESSOAS POR TÉCNICAS DE APRENDIZAGEM DE MÁQUINA

Person tracking using machine learning techniques

Proposta de Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Drª Kelly Lais Wiggers

GUARAPUAVA

2025

RESUMO

Este trabalho propõe o desenvolvimento de uma API *RESTful* acessível e personalizável para rastreamento e reidentificação de pessoas em vídeos, utilizando técnicas de aprendizado de máquina e visão computacional. A proposta busca democratizar o uso de sistemas baseados em inteligência artificial (IA) ao integrar modelos de detecção, rastreamento e extração de características visuais em uma interface funcional e intuitiva. Tais tecnologias, embora amplamente estudadas, ainda enfrentam barreiras de adoção devido à complexidade técnica e à escassez de ferramentas públicas e bem documentadas. A solução desenvolvida permitirá identificar indivíduos em transmissões ao vivo ou vídeos enviados, com suporte à adição de identificadores únicos e proteção da privacidade por meio de técnicas de criptografia. A iniciativa visa preencher lacunas existentes entre bibliotecas de pesquisa e aplicações práticas, oferecendo uma plataforma modular aplicável em contextos como segurança urbana, automação, controle de acesso e projetos sociais. O projeto utiliza bibliotecas consolidadas como YOLO e OSnet aliadas a *frameworks web* como *FastAPI*, para construir um *pipeline* de rastreamento *end-to-end* com foco em desempenho e acessibilidade por desenvolvedores e pesquisadores.

Palavras-chave: inteligencia artificial; deep learning; people tracking; re-identification; real-time video.

ABSTRACT

This project proposes the development of an accessible and customizable RESTful API for person tracking and re-identification in videos, using machine learning and computer vision techniques. The goal is to democratize the use of AI-based systems by integrating object detection, tracking, and feature extraction into a functional and user-friendly interface. Despite significant advances in this area, adoption remains limited due to technical complexity and the lack of public, well-documented tools. The proposed solution identifies individuals in both live video streams and uploaded recordings, supports unique identifiers, and includes data privacy measures such as encryption. The system bridges the gap between research libraries and practical applications, offering a modular platform suitable for urban security, automation, access control, and social projects. Built with established libraries such as YOLO, OSnet, and web frameworks like FastAPI, this end-to-end pipeline emphasizes performance and accessibility for developers and researchers.

Keywords: artificial intelligence; deep learning; people tracking; re-identification; real-time video.

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da era da informação, a sociedade tem se adaptado de forma contínua às inovações tecnológicas. Essas, por sua vez, permeiam desde aspectos cotidianos, como a substituição de linhas telefônicas por dispositivos móveis, até aplicações complexas de Inteligência Artificial (IA) voltadas à automação e otimização de processos em escala.

Assistentes virtuais, *chatbots* e sistemas inteligentes tornaram-se cada vez mais comuns no cotidiano da população, contribuindo para automatizar tarefas, reduzir a necessidade de intervenção humana e minimizar falhas operacionais. Muitos sistemas são baseados em arquiteturas de redes neurais artificiais inspiradas no funcionamento do cérebro humano, compostas por múltiplas camadas ocultas (*hidden layers*) que simulam sinapses neurais e realizam cálculos matemáticos em paralelo para transformar entradas (como imagens ou textos) em saídas interpretáveis. O aprendizado ocorre a partir de grandes volumes de dados previamente rotulados, utilizando algoritmos de retropropagação e otimização, de modo que o sistema passe a inferir padrões e tomar decisões com base em novas informações (GOODFELLOW; BENGIO; COURVILLE, 2016).

Entre os maiores desafios da IA, está o campo da visão computacional, mais especificamente o reconhecimento e o rastreamento de objetos em vídeos. Essa subárea da IA busca capacitar máquinas a interpretarem o conteúdo visual de imagens e vídeos, com o objetivo de detectar e acompanhar, ao longo do tempo, objetos ou indivíduos em movimento. Suas aplicações abrangem áreas como segurança pública, monitoramento urbano, controle de acesso (WEI *et al.*, 2020), varejo e marketing comportamental (JUNIOR; MARTINI, 2019), sistemas de transporte inteligentes e saúde pública (SILVA, 2022), por exemplo, para análise de fluxo em hospitais ou acompanhamento de pacientes com distúrbios de mobilidade (YADAV *et al.*, 2022).

Modelos modernos permitem, além da detecção de indivíduos, seu reconhecimento com base em características visuais, mesmo diante de ambientes com múltiplas câmeras, iluminação variável ou alta densidade populacional. Técnicas como detecção de objetos (por exemplo, o algoritmo *YOLO — You Only Look Once*, que divide imagens em grades e identifica objetos em tempo real) (REDMON *et al.*, 2016), rastreamento multi-objetos (como o DeepSORT, que combina informações espaciais com *features* visuais para acompanhar indivíduos ao longo de sequências (WOJKE; BEWLEY; PAULUS, 2017)) e reidentificação de pessoas (*person re-identification*), que utiliza vetores numéricos chamados *embeddings* para reconhecer a mesma pessoa em diferentes contextos, são frequentemente utilizadas em conjunto para garantir que uma mesma pessoa seja reconhecida ao longo de diferentes momentos e cenas (MCLAUGHLIN; RINCON; MILLER, 2016).

Apesar do avanço dessas soluções, sua adoção ainda é limitada pela complexidade técnica envolvida em sua implementação. Muitos dos *frameworks* existentes exigem conhecimentos especializados em aprendizado profundo (*deep learning*), engenharia de software e manipulação de vídeo em tempo real, o que restringe sua aplicação a empresas com equipes

altamente capacitadas. Além disso, o processo de treinamento de modelos próprios demanda poder computacional significativo e acesso a grandes bases de dados anotadas, o que inviabiliza o uso por desenvolvedores independentes ou instituições de pequeno porte.

O sistema utilizará modelos previamente treinados, com o intuito de facilitar sua aplicação em contextos diversos, como segurança urbana, empresas e ambientes domésticos. A proposta visa democratizar o acesso a essa tecnologia, unindo conceitos de processamento de vídeo, aprendizado de máquina e desenvolvimento *web* em uma solução funcional e intuitiva.

Adicionalmente, nota-se uma escassez de interfaces e APIs públicas que ofereçam suporte eficiente para reconhecimento de pessoas por imagem ou vídeo. Quando existentes, essas ferramentas costumam ser mal documentadas, desatualizadas ou de difícil integração com sistemas modernos. Este projeto busca preencher essa lacuna, utilizando modelos pré-treinados e *frameworks* de código aberto, como YOLOv11 (REDMON *et al.*, 2016), OSnet (ZHOU *et al.*, 2019) e FastAPI (RAMíREZ, 2023), para simplificar a adoção da tecnologia mesmo por usuários sem formação em IA.

Cabe destacar que, por lidar com dados sensíveis como imagens de pessoas, questões como desempenho, acurácia e segurança são cruciais. A aplicação proposta incluirá criptografia de identificadores (por meio de *hashes* gerados com funções seguras como *SHA-256* (STANDARDS; TECHNOLOGY, 2015) a fim de proteger a privacidade dos usuários e mitigar riscos de ataques ou uso indevido das informações. A interface será concebida de forma clara e segura, promovendo facilidade de uso sem comprometer a proteção dos dados.

Ao final, espera-se obter um protótipo funcional de API que permita rastrear indivíduos em vídeos com o uso de IA, com possibilidade de extensão futura para funcionalidades como alertas em tempo real, reconhecimento corporal, múltiplos alvos e *dashboards* analíticos.

1.1 Objetivos

1.1.1 Objetivo geral

Desenvolver uma API funcional para rastreamento de pessoas em vídeos, com suporte tanto para transmissões em tempo real quanto para *uploads*, utilizando técnicas de aprendizado de máquina e visão computacional.

1.1.2 Objetivos específicos

- Investigar e selecionar bases de dados e um modelo de detecção, rastreamento e re-identificação de pessoas que ofereça bom desempenho em tempo real.
- Integrar o modelo selecionado em uma aplicação acessível via API.

- Permitir a inserção de identificadores personalizados para reconhecimento individual.
- Criar uma interface mínima para interação com o sistema, com foco em usabilidade e segurança.
- Avaliar o desempenho do sistema em diferentes contextos (qualidade de vídeo, iluminação, número de indivíduos, métricas estatísticas).
- Implementar mecanismos de criptografia para garantir a privacidade dos dados dos usuários.
- Documentar a API de forma clara, visando sua reutilização por outros pesquisadores e desenvolvedores.

1.2 Justificativa

O tema proposto é de relevância no cenário contemporâneo, especialmente no contexto da crescente urbanização, da digitalização de serviços e do aumento da demanda por soluções tecnológicas voltadas à segurança, automação e gestão inteligente de espaços. O uso de sistemas de rastreamento e reconhecimento de pessoas por vídeo tem potencial para impactar diretamente áreas críticas como segurança pública, cidades inteligentes, controle de acesso, logística, varejo, saúde e ambientes corporativos (MELO; SERRA, 2022).

Em meio a esse panorama, destaca-se a dificuldade de acesso a ferramentas que possibilitem a implementação de tais sistemas de forma prática e acessível. As soluções atualmente disponíveis para rastreamento e identificação de pessoas por vídeo são, em sua maioria, restritas a grandes empresas ou instituições com equipes altamente especializadas, uma vez que exigem domínio técnico em aprendizado profundo, redes neurais convolucionais (RNCs), visão computacional e manipulação de fluxos de vídeo em tempo real. Essa barreira tecnológica restringe a adoção mais ampla da IA, impedindo que pesquisadores, pequenas empresas e até mesmo desenvolvedores experientes — mas não especializados em IA — consigam criar suas próprias soluções personalizadas (ALMASAWA; ELREFAEI; MORIA, 2019).

Além disso, falta no mercado um ecossistema sólido de APIs (termo refere-se à Interface de Programação de Aplicação, ou seja, um programa intermediário que se comunica com solicitações e respostas) públicas bem documentadas que permitam integração rápida com sistemas já existentes, o que limita o uso prático de IA em vídeo em contextos fora do meio corporativo. Este projeto responde diretamente a essa lacuna, propondo uma API robusta, adaptável e fácil de usar, que permita a qualquer pessoa — com ou sem conhecimento profundo em IA — implementar sistemas de rastreamento com segurança e eficiência.

Ao desenvolver uma API acessível, customizável e de fácil integração, este trabalho visa democratizar o acesso à tecnologia de rastreamento por vídeo baseada em IA. A proposta é permitir que diferentes perfis de usuários — incluindo pesquisadores, desenvolvedores *web* e

profissionais de TI — possam aplicar essas técnicas sem a necessidade de treinar modelos do zero ou compreender toda a complexidade dos algoritmos subjacentes. O projeto também se destaca por abordar um ponto sensível no desenvolvimento de sistemas de monitoramento: a proteção da privacidade e a segurança da informação. Por isso, o sistema proposto incorpora práticas como criptografia de identificadores (*hashes*) e uma interface clara e funcional, com foco em confiabilidade e proteção dos dados sensíveis dos usuários.

Adicionalmente, o projeto está alinhado aos princípios de viabilidade técnica e aplicabilidade prática. Existem bibliotecas e *frameworks* consolidados no ecossistema de código aberto — como OpenCV, YOLO, OSnet e Flask/FastAPI — que podem ser aproveitados para estruturar uma solução funcional dentro do tempo e dos recursos disponíveis no desenvolvimento de um trabalho de conclusão de curso.

Por fim, o projeto é também motivado por um interesse pessoal da autora nas áreas de aprendizado de máquina e desenvolvimento *web*. Trata-se, portanto, de uma oportunidade de aplicar conhecimentos adquiridos ao longo da formação acadêmica em um desafio real, que une teoria e prática e contribui tanto para o avanço pessoal quanto para o debate e a produção de conhecimento dentro da área de computação aplicada.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Contextualização Geral

Atualmente, um dos maiores desafios enfrentados por profissionais e pesquisadores que desejam utilizar inteligência artificial para rastreamento de pessoas em vídeo é a ausência de soluções completas e acessíveis que conduzam todo o processo de ponta a ponta (ALMASAWA; ELREFAEI; MORIA, 2019). Desta forma, falta no mercado e na comunidade científica uma ferramenta integrada que vá desde a captura do vídeo até a entrega dos resultados processados de forma compreensível e utilizável.

Um dos elementos centrais dessa lacuna está na geração automática de regiões de interesse (ROIs). Regiões de interesse são áreas específicas dentro de uma imagem ou vídeo onde se presume haver informação relevante — neste caso, a presença de uma pessoa. Para que sistemas automatizados funcionem corretamente, é necessário que consigam identificar, sem intervenção humana, onde essas pessoas estão no vídeo. Essa tarefa é realizada por modelos de detecção, como o YOLO (REDMON *et al.*, 2016), que segmentam os quadros do vídeo em tempo real, indicando com precisão onde os objetos — pessoas, neste caso — estão localizados.

Após a detecção, o segundo passo necessário é o processamento com inteligência artificial para a extração de características únicas dessas pessoas detectadas. Essas características, chamadas de *features* ou *embeddings*, são vetores numéricos que codificam informações visuais relevantes de cada indivíduo — como tipo de roupa, cor, estrutura corporal, e outros padrões que, juntos, permitem distinguir uma pessoa de outra. Esses vetores são fundamentais para realizar o que se chama de reidentificação de pessoas, ou seja, reconhecer o mesmo indivíduo em câmeras diferentes, em momentos distintos, mesmo que ele não esteja sempre na mesma posição ou iluminação.

Diversas abordagens têm sido desenvolvidas para gerar *embeddings* eficazes para tal objetivo, e dentro elas, incluem-se:

- Aprendizado Métrico Profundo:
  - Utiliza redes neurais treinadas com funções de perda específicas, como a *triplet loss*, para mapear imagens de pessoas em um espaço vetorial onde indivíduos semelhantes estão próximos e diferentes estão distantes.
  - É eficaz na captura de relações de similaridades e diferenças entre indivíduos, (LI *et al.*, 2023)
- Aprendizado de Características Locais:

  - Foca na extração de partes específicas do corpo (por exemplo, cabeça, ombro, joelho e pé) para gerar *embeddings* mais robustos e variações de pose e oclusões.
  - Modelos como o PCB (*Part-based Convolutional Baseline*) dividem a imagem em segmentos horizontais e extraem características de cada parte, (WU *et al.*, 2024).

- Redes Adversarias Generativas (GANs):
  - Treina duas redes neurais generativas para competirem entre si e gerar novos dados mais autênticos a partir de um determinado conjunto de dados de treinamento.
  - Utilizadas para gerar imagens sintéticas ou transformar imagens existentes, ajudando a modelar variações de aparência e melhorar a generalização dos *embeddings*, (Amazon Web Services, 2023).

- Aprendizado de Características Sequenciais:
  - Explora informações temporais em sequências de vídeo, utiliza redes recorrentes ou mecanismos de atenção para capturar padrões dinâmicos de movimento e comportamento.
  - Essa abordagem é valiosa para reidentificação em vídeos, onde o movimento pode fornecer pistas adicionais, (AL-JABERY *et al.*, 2020).

Enquanto o YOLO (*You Only Look Once*) é amplamente utilizado para detecção de objetos em tempo real, incluindo pessoas, ele não é projetado para reidentificação. O YOLO divide a imagem em uma grade e prevê *bounding boxes* e classes para cada célula, permitindo a detecção rápida de objetos. No entanto, para reidentificar indivíduos, é necessário extrair *embeddings* que capturem características únicas além da simples detecção.

Portanto, após a detecção inicial com o YOLO, é comum empregar modelos especializados em reidentificação para extrair *embeddings* e realizar a correspondência entre indivíduos ao longo do tempo e em diferentes câmeras.

O terceiro aspecto crítico é a entrega dos resultados em um sistema *web* funcional, que permita que esses dados processados sejam utilizados por outros sistemas ou por usuários humanos sem conhecimento técnico em IA. Isso envolve não apenas a construção de uma API pública, que permita que sistemas externos enviem vídeos e recebam respostas com as identificações, mas também a criação de uma interface gráfica amigável, acessível por navegador, onde se possa testar e visualizar o funcionamento da solução. É nesse ponto que se torna clara a importância da arquitetura *end-to-end* — ou seja, que conecte todos os estágios do processo de forma integrada e fluída.

Tornar tecnologias avançadas de rastreamento de pessoas por vídeo acessíveis e reutilizáveis possui grande impacto social e tecnológico. A segurança pública e privada, por exemplo, frequentemente depende da análise de horas de gravações de câmeras, o que consome tempo e recursos humanos. Ferramentas automatizadas podem agilizar esse processo, reduzindo falhas humanas e otimizando respostas em tempo real. Em ambientes corporativos ou comerciais, como shoppings, supermercados ou escolas, o monitoramento de fluxo de pessoas permite, além da segurança, a análise de comportamento de clientes, otimização de rotas e melhoria da experiência do usuário. Em prédios públicos ou privados, o controle de acesso por meio de reidentificação permite sistemas mais inteligentes e adaptativos, promovendo automação predial e redução de custos com pessoal.

O projeto também se mostra essencial do ponto de vista da acessibilidade e inclusão digital, pois oferece uma alternativa gratuita, aberta e de baixo custo para uma tecnologia que atualmente é majoritariamente proprietária e cara. Isso possibilita que organizações de menor porte, iniciativas sociais e pesquisadores em início de carreira possam acessar e explorar o potencial da inteligência artificial aplicada ao vídeo, mesmo sem uma grande infraestrutura computacional.

2.2 Funcionamento de uma Inteligência Artificial

A inteligência artificial é um campo da ciência da computação que busca desenvolver sistemas capazes de executar tarefas que normalmente exigiriam inteligência humana, como percepção visual, reconhecimento de padrões, tomada de decisões e processamento de linguagem natural. Entre os principais subcampos da IA, destaca-se o aprendizado de máquina (AP), que permite que os sistemas aprendam padrões e tomem decisões com base em dados, sem serem explicitamente programados para cada tarefa específica (GOODFELLOW; BENGIO; COURVILLE, 2016).

O aprendizado de máquina, por sua vez, possui uma vertente ainda mais especializada, o aprendizado profundo, que se baseia em arquiteturas conhecidas como redes neurais convolucionais (RNC). Essas redes são compostas por múltiplas camadas interconectadas, (GOODFELLOW; BENGIO; COURVILLE, 2016).

No contexto da visão computacional, as redes neurais convolucionais são as arquiteturas mais utilizadas. Elas foram projetadas para processar dados que possuem uma estrutura de matriz, como imagens, e são compostas por filtros capazes de capturar padrões locais, como bordas, texturas e formas (GOODFELLOW; BENGIO; COURVILLE, 2016). À medida que a informação percorre as camadas da rede, essas representações locais são combinadas em padrões mais complexos, permitindo que o modelo reconheça objetos inteiros ou identifique características específicas de uma pessoa.

O funcionamento de uma IA para tarefas como detecção e reidentificação de pessoas ocorre, portanto, em etapas bem definidas:

1. Na primeira etapa, chamada de detecção, um modelo como o YOLO analisa cada quadro de um vídeo e localiza objetos de interesse, como pessoas, retornando *bounding boxes* com suas respectivas classificações.

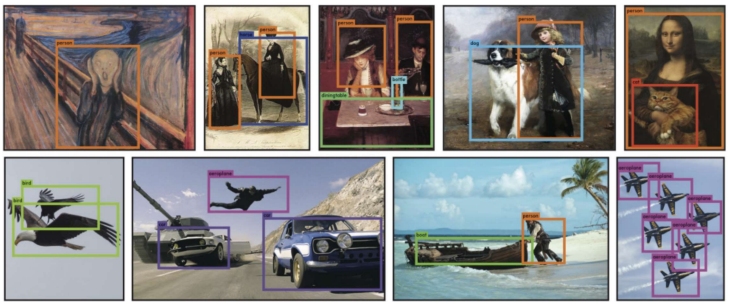

Figura 1 – Exemplo de detecção de objetos com YOLO
Fonte: You Only Look Once: Unified, Real-Time Object Detection (REDMON *et al.*, 2016).

2. Na segunda etapa, chamada de extração de *embeddings*, os modelos de reidentificação aplicam uma RNC sobre as imagens recortadas das pessoas detectadas. O resultado é um vetor numérico que representa matematicamente as características visuais do indivíduo, como cor da roupa, proporção corporal, padrões e texturas (GOODFELLOW; BENGIO; COURVILLE, 2016).

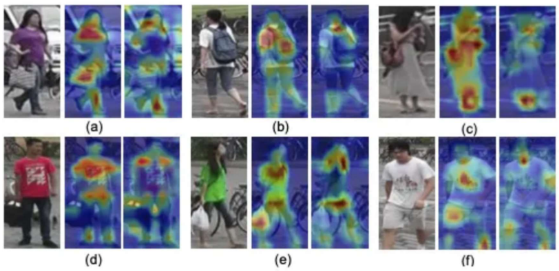

Figura 2 – Exemplo de mapeamento de *embeddings*.
Fonte: Omni-Scale Feature Learning for Person Re-Identification. (ZHOU *et al.*, 2019)

3. Por fim, a etapa de comparação vetorial consiste em medir a distância entre os *embeddings* de uma imagem de referência e os *embeddings* extraídos dos quadros do vídeo. Distâncias menores indicam maior similaridade, permitindo ao sistema concluir se se trata do mesmo indivíduo observado anteriormente.

Esse processo ilustra como uma IA não "enxerga" uma pessoa da mesma forma que a visão humana, mas sim como um conjunto de números que representam características extraídas das imagens. A qualidade dessa representação é diretamente dependente do modelo escolhido, da diversidade do conjunto de dados utilizado no treinamento e das técnicas aplicadas, como *Deep Metric Learning*, *Local Feature Learning* e o uso de GANs para aumento de dados.

Além disso, para que esse tipo de IA seja útil no mundo real, é necessário que o modelo treinado seja encapsulado em uma infraestrutura capaz de recebê-lo como serviço, geralmente

por meio de APIs RESTful, com suporte a comunicação em tempo real via *WebSockets*, e que exponha funcionalidades acessíveis tanto para sistemas externos quanto para usuários finais não técnicos.

2.2.1 Deep Learning

O *Deep Learning*, ou aprendizado profundo, é uma subárea do aprendizado de máquina que tem como objetivo construir modelos computacionais capazes de aprender representações hierárquicas de dados, utilizando arquiteturas compostas por múltiplas camadas de processamento não linear, conhecidas como redes neurais profundas (GOODFELLOW; BENGIO; COURVILLE, 2016).

O termo "profundo" deve-se a presença de várias camadas sucessivas de transformação dos dados, onde cada camada aprende um representação progressivamente mais abstrata de informação.

Historicamente, o conceito de "redes neurais" existe desde a década de 1950, com o modelo de *perceptron* desenvolvido por Frank Rosenblatt. Porém, as primeiras arquiteturas de redes neurais convolucionais apresentavam várias limitações práticas, como por exemplo, a falta de recursos computacionais precedentes à época (GOODFELLOW; BENGIO; COURVILLE, 2016). Foi apenas a partir de 2006, com o desenvolvimento de técnicas como a pré-treinamento não supervisionado e, posteriormente, com a popularização de unidades de processamento gráfico (GPUs) e de algoritmos como a retropropagação, que o aprendizado profundo passou a alcançar resultados expressivos em problemas reais.

O aprendizado profundo se tornou uma abordagem central em diversos ramos da inteligência artificial, especialmente em visão computacional, processamento de linguagem natural, reconhecimento de fala e jogos, devido à sua capacidade de aprender diretamente dos dados, dispensando engenharia manual de *features*, aplicando suas próprias regras para medir o peso dessas *features*.

2.2.2 Arquitetura de uma Rede Neural Profunda

Uma rede neural profunda é composta por um conjunto de unidades computacionais chamadas de neurônios artificiais ou nós, organizados em múltiplas camadas. Essas camadas podem ser divididas em três tipos principais: camada de entrada (*input*), camadas ocultas (*hidden layers*) e camada de saída (*output*).

A Figura 3 ilustra uma arquitetura típica de rede neural profunda com múltiplas camadas ocultas.

Cada neurônio artificial recebe um conjunto de entradas, as quais são multiplicadas por pesos associados a cada conexão. Em seguida, essas entradas ponderadas são somadas à um

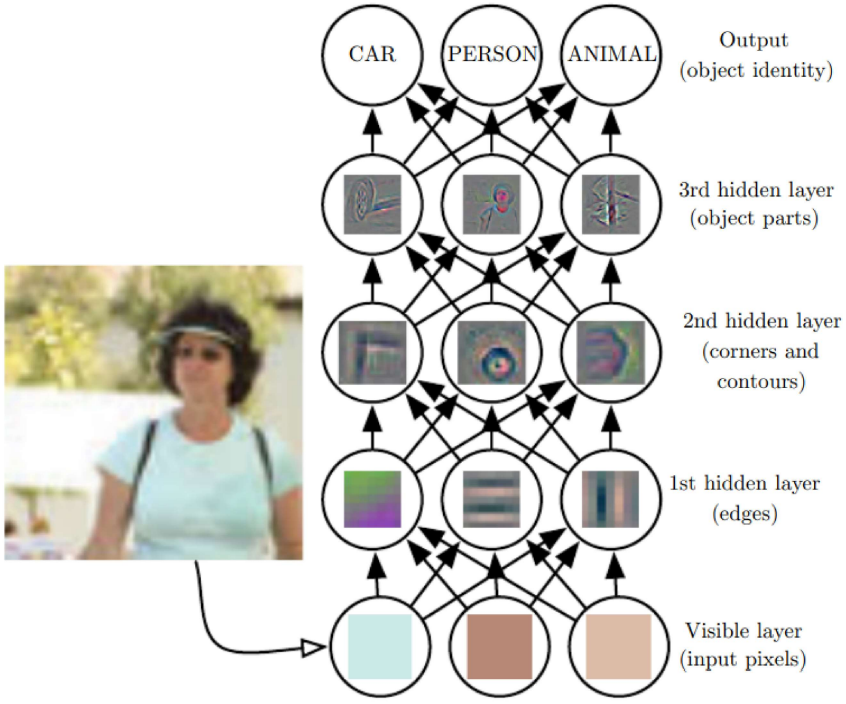

Figura 3 – Exemplo de arquitetura de rede neural profunda com múltiplas camadas ocultas.
Fonte: *Deep Learning*, (GOODFELLOW; BENGIO; COURVILLE, 2016).

viés e passam por uma função de ativação, como a ReLU (*Rectified Linear Unit*), Sigmoid ou Tanh, que introduz não-linearidade ao sistema (GOODFELLOW; BENGIO; COURVILLE, 2016). Essa característica permite que a rede seja capaz de modelar relações complexas entre os dados.

2.2.3 Parâmetros e Hiperparâmetros

O desempenho e o comportamento de uma rede neural profunda durante o treinamento e a inferência dependem de diversos parâmetros e hiperparâmetros que influenciam desde o ajuste dos pesos até a estabilidade do aprendizado. A seguir, são descritos os principais hiperparâmetros utilizados no desenvolvimento de modelos de aprendizado profundo.

- **Número de épocas:** Quantidade de vezes que o modelo percorrerá o conjunto de treinamento completo. Durante cada época, o modelo realiza sucessivas atualizações dos seus parâmetros (pesos e viés), com o objetivo de reduzir o erro em relação aos valores esperados. Normalmente, o treinamento de redes profundas requer múltiplas épocas para que o modelo consiga generalizar bem os padrões aprendidos. No entanto, um número excessivo de épocas pode levar ao sobreajuste, situação em que o modelo passa a memorizar os dados de treinamento e perde capacidade de generalização para novos dados (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Viés:** Parâmetro adicional presente em cada neurônio de uma rede neural, que tem como função ajustar a saída do neurônio mesmo quando todas as entradas são zero. O viés também é ajustado na função de gradiente descendente, e é inicializado com valores pequenos e aleatórios, comumente (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Taxa de aprendizado:** A taxa de aprendizado é um hiperparâmetro fundamental que controla a magnitude da atualização dos pesos da rede a cada iteração do processo de treinamento. Ela determina o quão rápido ou devagar o modelo ajusta os seus parâmetros em resposta ao gradiente da função de perda. Uma taxa de aprendizado muito alta pode fazer com que o modelo oscile ou até mesmo divirja, nunca alcançando um mínimo da função de perda. Por outro lado, uma taxa muito baixa pode tornar o treinamento extremamente lento, além de aumentar o risco de o modelo ficar preso em mínimos locais (GOODFELLOW; BENGIO; COURVILLE, 2016).

- ***Batch size*:** O *batch size* refere-se ao número de amostras de dados utilizadas para calcular o gradiente e atualizar os pesos da rede a cada iteração de treinamento. Em vez de atualizar os pesos após cada amostra individual (*stochastic gradient descent*) ou apenas ao final de uma época (*batch gradient descent*), utiliza-se o *mini-batch gradient descent*, que realiza as atualizações após processar pequenos lotes de amostras. Essa abordagem equilibra o custo computacional e a estabilidade do gradiente, sendo uma prática comum o uso de tamanhos de lote entre 16, 32, 64 ou 128, dependendo dos recursos computacionais disponíveis e do tamanho do *dataset* (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Função de perda:** A função de perda é responsável por quantificar a discrepância entre a saída prevista pelo modelo e os valores reais. Durante o treinamento, o objetivo da rede neural é minimizar essa discrepância. Existem diferentes funções de perda, e a escolha depende do tipo de tarefa. Para problemas de classificação, por exemplo, é comum utilizar a *Cross-Entropy Loss*, enquanto para regressão, funções como *Mean Squared Error* (MSE) são mais adequadas. A função de perda orienta o cálculo dos gradientes durante o processo de retropropagação, influenciando diretamente na atualização dos parâmetros do modelo (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Otimização:** O algoritmo de otimização é o método utilizado para minimizar a função de perda ajustando os pesos e viés da rede de forma eficiente. O mais tradicional é o *Gradient Descent*, mas versões mais avançadas são amplamente utilizadas devido à sua capacidade de acelerar a convergência e lidar melhor com problemas como platôs de gradiente e ravinas. Entre os otimizadores mais populares destacam-se:

  - **SGD *Stochastic Gradient Descent***: Realiza atualizações com base em lotes pequenos, sendo simples, mas eficiente.

  - **Adam (*Adaptive Moment Estimation*)**: Ajusta automaticamente a taxa de aprendizado de cada parâmetro com base em momentos do gradiente, proporcionando uma convergência mais rápida e estável.
  - **RMSprop**: Uma variação que normaliza os gradientes pelo quadrado da média das magnitudes recentes, sendo eficaz em problemas com gradientes esparsos (GOODFELLOW; BENGIO; COURVILLE, 2016).

  A escolha do otimizador influencia diretamente a eficiência do treinamento e a qualidade da solução final.

- **Função de ativação:** A função de ativação é responsável por introduzir não-linearidades no modelo, permitindo que a rede neural seja capaz de aprender relações complexas entre as entradas e as saídas. Sem uma função de ativação não-linear, a rede seria apenas uma combinação linear de suas entradas, independente da quantidade de camadas, impedindo o aprendizado de padrões complexos, como para reconhecimento de imagens.

  As funções de ativação mais comuns incluem:

  - **ReLU (*Rectified Linear Unit*):** Define a saída como zero para valores negativos e linear para positivos, sendo a mais utilizada em redes profundas devido à sua simplicidade e eficácia em evitar o problema do gradiente desvanecente.
  - **Sigmoid:** Comprimi os valores de saída para um intervalo entre 0 e 1, muito usada em problemas de classificação binária, mas com risco de saturação para entradas muito altas ou muito baixas.
  - **Tanh (Tangente Hiperbólica)**: Similar à sigmoid, mas com saída entre -1 e 1, o que pode proporcionar melhor centralização dos dados em algumas situações (GOODFELLOW; BENGIO; COURVILLE, 2016).

Esses conceitos serão explorados em detalhes na seção de metodologia, onde serão definidos os valores específicos utilizados em cada etapa do desenvolvimento da solução proposta.

2.2.4 Processo de Treinamento

O treinamento de uma rede neural profunda consiste em dois ciclos principais: a passagem direta e a retropropagação.

Durante o passagem direta, os dados de entrada são processados camada por camada, da entrada até a saída da rede. Cada neurônio artificial realiza uma combinação linear das entradas recebidas, ponderadas pelos seus respectivos pesos, e adiciona um termo de viés. O resultado dessa combinação é então passado por uma função de ativação, como a ReLU ou a

Sigmoid, que introduz não-linearidade ao modelo, permitindo que a rede aprenda representações complexas (GOODFELLOW; BENGIO; COURVILLE, 2016).

Matematicamente, o funcionamento de cada camada pode ser descrito pela Equação 1:

$$a^{(l)} = f\left(W^{(l)} \cdot a^{(l-1)} + b^{(l)}\right) \tag{1}$$

Onde:

- a(l) representa a saída da camada l;
- W(l) são os pesos da camada l;
- b(l) é o vetor de vieses;
- f é a função de ativação escolhida para a camada.

Ao final da última camada, a rede produz uma predição ou saída (ŷ), que é comparada ao valor real (y) por meio de uma função de perda, como *Cross-Entropy Loss* para tarefas de classificação ou *Mean Squared Error* para regressão. Essa função calcula o erro entre a predição e o valor esperado.

Em seguida, inicia-se o retropropagação, onde o erro calculado é propagado de volta através das camadas da rede. O algoritmo de retropropagação, proposto originalmente por Rumelhart, Hinton e Williams (1986), calcula os gradientes parciais da função de perda em relação aos pesos de cada camada, utilizando o método de diferenciação conhecida como regra da cadeia do cálculo diferencial (GOODFELLOW; BENGIO; COURVILLE, 2016).

Esses gradientes são então utilizados por um algoritmo de otimização, como o SGD, Adam ou RMSprop, para ajustar os pesos da rede, reduzindo gradualmente o erro nas próximas iterações (BERLYAND; JABIN, 2023).

O processo de treinamento repete o passagem direta e o retropropagação para múltiplos *mini-batches* até completar uma época, que corresponde a uma varredura completa por todo o conjunto de treinamento.

O ciclo completo é repetido ao longo de múltiplas épocas, enquanto o desempenho da rede é monitorado por meio de métricas como acurácia, precisão, *recall* e perda de validação. Durante o treinamento, hiperparâmetros como a taxa de aprendizado, o *batch size*, o número de épocas e o tipo de otimizador têm influência direta na velocidade de convergência e na qualidade do modelo final.

Durante o processo de treinamento, técnicas de regularização também são essenciais para evitar o fenômeno conhecido como sobreajuste, no qual a rede aprende padrões muito específicos dos dados de treinamento e apresenta baixo desempenho em dados não vistos anteriormente.

Uma das estratégias mais utilizadas para mitigar esse problema é o *Dropout*, proposto por Srivastava *et al.* (2014). O *Dropout* consiste em desativar aleatoriamente uma porcentagem

dos neurônios durante cada iteração de treinamento, forçando a rede a não depender excessivamente de nenhum caminho específico para a propagação de informação. Isso incentiva a formação de representações mais robustas e generalizáveis. O valor típico para o parâmetro de *Dropout* varia entre 0,2 e 0,5, significando que entre 20% e 50% dos neurônios podem ser temporariamente desativados por iteração (SRIVASTAVA *et al.*, 2014).

Além do *Dropout*, outra prática comum durante o treinamento de redes neurais profundas é o uso de *Data Augmentation*. Esta técnica visa aumentar a diversidade dos dados de entrada por meio de transformações artificiais aplicadas ao conjunto de dados original. Entre as formas mais comuns de *Data Augmentation* em visão computacional estão: rotações, inversões horizontais, variações de brilho, recortes aleatórios (*random cropping*) e alterações de escala (*scale jittering*) (GOODFELLOW; BENGIO; COURVILLE, 2016). O objetivo é tornar o modelo mais robusto a variações que podem ocorrer nas condições reais de uso, como mudanças de iluminação, perspectiva ou oclusões parciais.

Combinadas, as estratégias de otimização, regularização (como *Dropout*) e *Data Augmentation* contribuem para o desenvolvimento de um modelo mais preciso e com melhor capacidade de generalização ao lidar com novas amostras de dados.

2.2.5 Detecção de Pessoas com YOLO (*You Only Look Once*)

O algoritmo YOLO é uma das arquiteturas mais conhecidas e utilizadas na tarefa de detecção de objetos em tempo real. Proposto por Redmon *et al.* (2016), o YOLO surgiu com o objetivo de resolver limitações de modelos anteriores que, embora apresentassem alta acurácia, sofriam com elevado tempo de processamento, tornando-os inviáveis para aplicações com restrições de latência, como vídeos ao vivo.

A Figura 4 ilustra uma arquitetura da rede neural introduzida no YOLO.

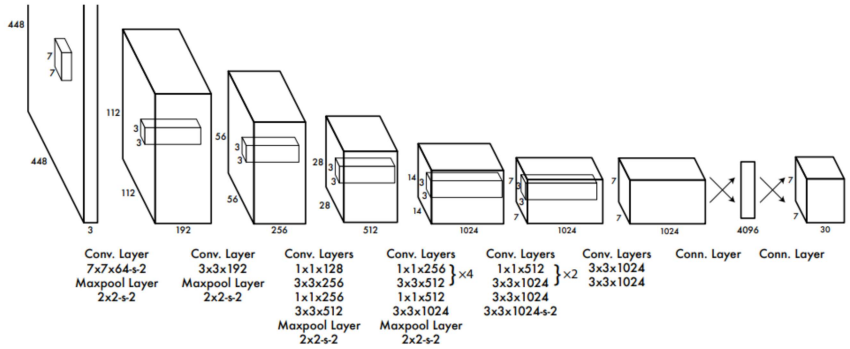

Figura 4 – Arquitetura da rede convolucional usada no modelo YOLO.
Fonte: You Only Look Once: Unified, Real-Time Object Detection (REDMON *et al.*, 2016)

Enquanto arquiteturas tradicionais de detecção de objetos operam por meio de um *pipeline* de múltiplas etapas — geralmente incluindo a geração de regiões propostas, extração de características para cada uma dessas regiões e posterior classificação — o YOLO adota uma

abordagem de regressão direta, tratando a detecção como um único problema de mapeamento de entrada para saída (REDMON *et al.*, 2016).

O funcionamento do YOLO pode ser descrito da seguinte forma: a imagem de entrada é dividida em uma *grid* de células de tamanho fixo. Para cada célula da grade, o modelo prevê simultaneamente:

- As coordenadas das *bounding boxes* — incluindo posição (x, y) e dimensões (largura e altura);
- A classe (ou tipo) do objeto contido naquela caixa (ex.: pessoa, carro, bicicleta);
- A confiança da detecção — um valor numérico que expressa o grau de certeza da rede de que a caixa contém um objeto e que esse objeto pertence à classe prevista.

Essa estrutura permite que o YOLO realize a detecção de todos os objetos de uma imagem em uma única execução da rede neural, sem a necessidade de percorrer múltiplas regiões de forma sequencial. Em termos práticos, isso resulta em uma redução significativa do tempo de processamento, permitindo a detecção em tempo real, mesmo em vídeos com altas taxas de quadros por segundo.

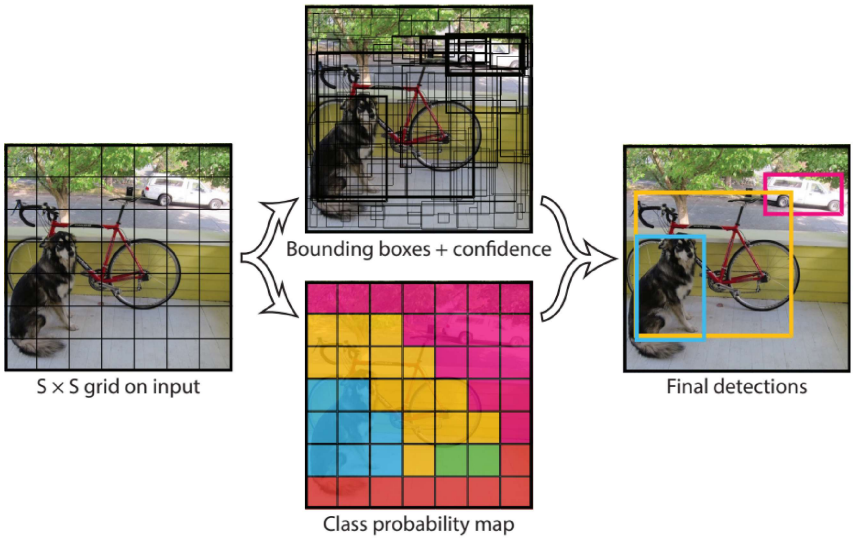

Figura 5 – Formação de *Bounding Boxes* e classificação de imagens com YOLO.
Fonte: You Only Look Once: Unified, Real-Time Object Detection (REDMON *et al.*, 2016)

A saída final do YOLO consiste em uma série de caixas delimitadoras com suas respectivas classes e níveis de confiança, permitindo ao sistema identificar múltiplos objetos em diferentes posições de uma única imagem ou quadro de vídeo.

Essa característica faz com que o YOLO seja uma das principais escolhas em aplicações que exigem rapidez, como vigilância por vídeo, direção autônoma e análise de fluxo de pessoas em tempo real (REDMON *et al.*, 2016).

Desde a sua primeira versão, o YOLO passou por diversas atualizações significativas. Versões como o YOLOv5, YOLOv8 e YOLOv11 apresentam melhorias tanto em desempenho quanto em acurácia, incorporando avanços como camadas de atenção, arquiteturas mais profundas e técnicas modernas de otimização (REDMON *et al.*, 2016).

2.2.6 Reidentificação de Pessoas (Person Re-Identification)

A Reidentificação de Pessoas é uma tarefa dentro da visão computacional que consiste em reconhecer o mesmo indivíduo em diferentes cenas, câmeras ou momentos, mesmo que a pessoa esteja com variações de ângulo, iluminação ou postura (ALMASAWA; ELREFAEI; MORIA, 2019).

O processo de Re-ID geralmente é composto por três etapas principais:

1. **Extração de *Embeddings*:** Após a detecção da pessoa, a imagem recortada (ROI) é processada por uma rede neural convolucional especializada, como a OSNet, para extrair um vetor numérico de alta dimensão (*embedding*) que representa as características visuais únicas daquele indivíduo (ZHOU *et al.*, 2019).

2. **Comparação Vetorial:** Através de uma métrica de distância (como a Euclidiana ou *Cosine Distance*), o sistema compara o *embedding* da pessoa detectada com o e*mbedding* de referência fornecido pelo usuário. *Embeddings* com menor distância são considerados mais similares, permitindo a identificação da pessoa alvo (ALMASAWA; ELREFAEI; MORIA, 2019).

3. **Decisão de Identidade:** Caso a distância calculada esteja abaixo de um limiar previamente definido, o sistema considera que a detecção corresponde ao indivíduo de interesse.

2.2.6.1 Arquitetura OSNet para Re-ID

O OSNet é uma arquitetura de rede neural convolucional especializada na extração de *embeddings* de imagens de pessoas. Seu diferencial está na capacidade de capturar informações em múltiplas escalas espaciais de forma simultânea, por meio de blocos chamados *Omni-Scale Blocks*, que integram diferentes campos receptivos de convolução. Isso permite ao modelo ser eficiente tanto na captura de detalhes locais (como textura de roupa) quanto de contextos globais (como estrutura corporal) (ZHOU *et al.*, 2019).

A Figura 6 ilustra a arquitetura típica do OSNet.

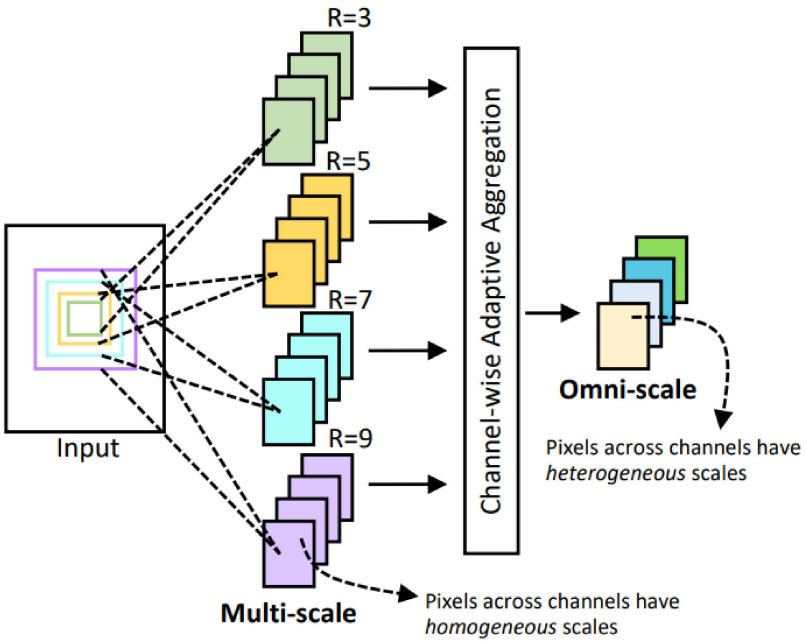

Figura 6 – Arquitetura convolucional usada no modelo Omni Scale.
Fonte: Omni-Scale Feature Learning for Person Re-Identification (ZHOU *et al.*, 2019)

2.3 Funcionamento de Aplicação *Web*

Nesta sessão, será explicado um pouco sobre o funcionamento por trás de uma aplicação *web* e como a integração de uma (ou mais) inteligência artificial se insere no contexto geral desse projeto.

2.3.1 API RESTful

Uma API RESTful (*Representational State Transfer*) é um estilo arquitetural para desenvolvimento de aplicações *web* que seguem os princípios do protocolo HTTP, ou seja, é enviado uma solicitação de um cliente (ex.: um navegador) para um servidor, pedindo uma ação ou informação. O termo foi inicialmente definido por Roy Fielding em sua tese de doutorado em Fielding (2000). APIs baseadas em REST permitem que diferentes sistemas se comuniquem através de requisições HTTP, utilizando métodos como GET, POST, PUT e DELETE.

Entre as principais características de uma API RESTful estão:

- **Comunicação *Stateless*:** Cada requisição do cliente ao servidor é independente e não requer armazenamento de contexto entre as requisições.
- **Uso de Recursos:** Cada recurso (como imagens, vídeos, dados de usuário) é identificado por uma URL única.
- **Formato Padrão de Dados:** A troca de informações geralmente é realizada em formatos como JSON ou XML.

2.3.2 *WebSocket*

O protocolo *WebSocket*, padronizado pela RFC 6455 (FETTE; MELNIKOV, 2011), é uma tecnologia que permite comunicação bidirecional, persistente e em tempo real entre cliente e servidor através de uma única conexão TCP. Diferentemente das requisições HTTP tradicionais, nas quais cada mensagem necessita abrir e fechar uma conexão, o *WebSocket* mantém uma conexão aberta, permitindo que o servidor envie dados ao cliente assim que novos eventos ocorrem.

Essa característica faz com que o *WebSocket* seja especialmente adequado para aplicações que necessitam de baixa latência e comunicação em tempo real, como sistemas de transmissão de vídeo ao vivo ou monitoramento de dados contínuos.

2.3.3 Integração entre API RESTful e *WebSocket* no Sistema Proposto

A combinação de uma API RESTful e de um canal *WebSocket* proporciona uma arquitetura flexível e escalável para o sistema de rastreamento de pessoas. Enquanto a API RESTful permitirá operações pontuais, como o envio de imagens de referência ou vídeos armazenados, o *WebSocket* garantirá a comunicação contínua e em tempo real para análise de transmissões ao vivo.

Essa arquitetura permitirá que usuários de diferentes perfis — desde desenvolvedores até operadores de segurança — interajam com o sistema de forma eficiente, acessível e com baixo tempo de resposta.

2.4 Estudos Relacionados

Este trabalho se insere dentre múltiplas linhas de pesquisa consolidadas na área de ciência da computação e engenharia de software. Está diretamente relacionado à área de reconhecimento facial e reidentificação de pessoas, que estuda formas de identificar indivíduos por meio de características visuais, com ou sem uso de biometria direta. Assim também com o campo da vigilância inteligente, onde algoritmos analisam vídeos em tempo real para tomar decisões ou alertar operadores sobre eventos de interesse.

Um exemplo significativo é o trabalho de Chen *et al.* (2018), que aplicou *Deep Metric Learning* com a função de perda *triplet loss* para gerar *embeddings* discriminativos em tarefas de reidentificação. Os autores demonstraram que esse método pode atingir alta acurácia ao separar vetorialmente indivíduos distintos com eficácia, mesmo em grandes conjuntos de dados.

Já o modelo PCB, proposto por Sun *et al.* (2021), adota uma abordagem de aprendizado local, segmentando a imagem em partes horizontais e extraindo características de cada uma

delas. Essa estratégia mostrou-se particularmente eficaz em cenários com variações de pose e oclusão parcial, frequentemente encontradas em ambientes reais.

No contexto de melhoria de generalização e aumento da variabilidade visual, Karmakar e Mishra (2021) introduziram o uso de GANs para gerar amostras sintéticas de pessoas. Isso permitiu treinar modelos mais robustos mesmo com conjuntos de dados limitados, ampliando a capacidade dos *embeddings* em generalizar para novos indivíduos.

Quanto à reidentificação em vídeos, McLaughlin, Rincon e Miller (2016) propuseram uma rede recorrente convolucional (RRC) capaz de capturar padrões temporais para melhorar a identificação de indivíduos em sequências contínuas. O modelo demonstrou que o uso de informações temporais melhora significativamente o desempenho em comparação com abordagens baseadas apenas em imagens estáticas.

Além disso, há outras pesquisas que merecem ser mencionadas em rastreamento de multi-objetos , uma vertente da inteligência artificial que busca acompanhar simultaneamente várias entidades em movimento ao longo do tempo, como o estudo de Bergmann, Meinhardt e Leal-Taixe (2019) com o *Tracktor++*, que combina detecção com rastreamento baseado em *tracking by regression*. Outro elo importante está com a detecção de anomalias em vídeo, como ações suspeitas ou comportamentos fora do padrão, que utilizam muitos dos mesmos fundamentos técnicos aqui abordados.

Por fim, estes estudos podem também tangenciar áreas mais aplicadas como engenharia de *software*, especialmente no que diz respeito à criação de infraestrutura moderna de APIs e interfaces *web* funcionais, que são essenciais para tornar as descobertas da inteligência artificial utilizáveis no mundo real, fora do ambiente restrito de laboratórios e grupos de pesquisa.

Além das contribuições acadêmicas em reidentificação e rastreamento, este trabalho também tangencia discussões recentes em engenharia de software para sistemas baseados em IA, especialmente no que se refere à produção de APIs modernas. Ramachandran (2024) propõe um modelo de "*API AI-first*", que integra o processamento vetorial, infraestrutura e práticas de segurança "*zero-trust*", otimizando o uso de modelos de aprendizado de máquina em produção.

3 MATERIAIS E MÉTODOS

Este Trabalho de Conclusão de Curso tem por objetivo o desenvolvimento de uma ferramenta *end-to-end* de rastreamento de pessoas baseada em inteligência artificial, com o intuito de possibilitar a detecção e identificação de indivíduos em vídeos transmitidos em tempo real ou por meio de arquivos enviados. A solução combina técnicas modernas de visão computacional, como detecção de regiões de interesse (ROIs), extração de *embeddings* e comparação vetorial, de forma a reconhecer um indivíduo específico a partir de uma imagem de referência fornecida pelo usuário. Uma visão geral do desenvolvimento pode ser visto na Figura 7.

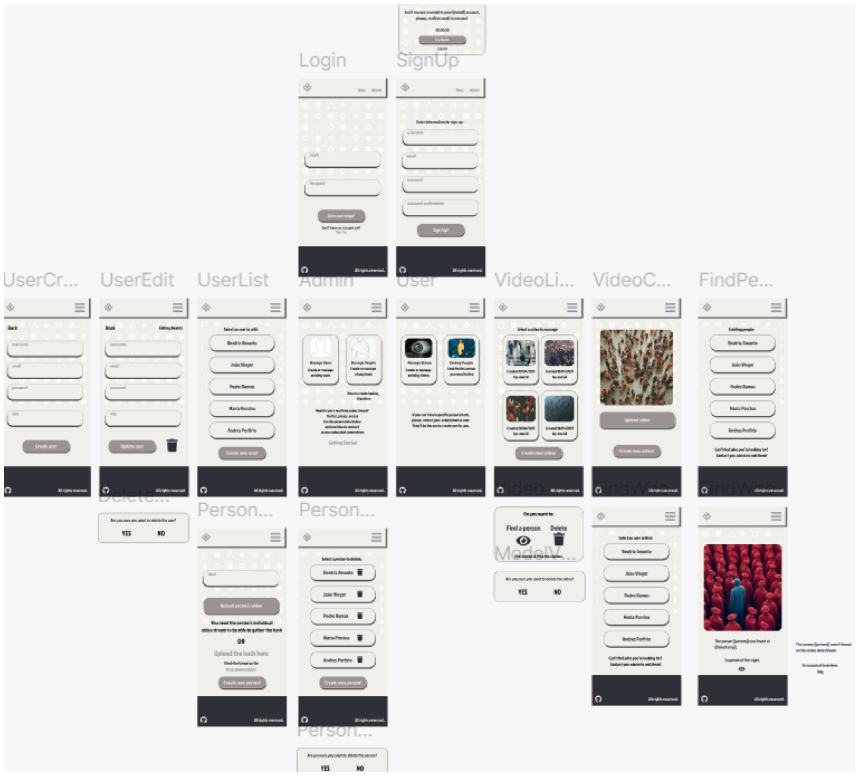

Figura 7 – Visão geral do sistema *Watch Me*.
Fonte: Autoria própria.

A ferramenta será estruturada como uma API acessível, com conexão via protocolo *websocket* e suporte a requisições RESTful, além de oferecer um *dashboard* interativo para visualização dos resultados. Por meio dessa interface, o usuário poderá carregar uma imagem de uma pessoa de interesse e conectar uma fonte de vídeo — seja por *upload* direto ou por *streaming*. A API será responsável por identificar se, onde e quando o indivíduo-alvo aparece nos vídeos processados, retornando as informações em formato estruturado (JSON) e visualizando os dados em tempo real por meio do painel *web*.

3.1 Materiais

O desenvolvimento da solução proposta neste Trabalho de Conclusão de Curso exige a utilização de um conjunto diversificado de ferramentas, *frameworks*, linguagens de programação e serviços de infraestrutura. A seguir, são apresentados os principais materiais que serão empregados ao longo das etapas de desenvolvimento e implementação da API para rastreamento de pessoas com inteligência artificial.

- **Figma:** O Figma é uma ferramenta de prototipação e design de interfaces amplamente utilizada no desenvolvimento de aplicações *web* e *mobile*. Ele permite a criação de um *design system* estruturado, proporcionando melhor organização visual e facilitando a implementação da experiência do usuário (UX) e da interface do usuário (UI). Entre suas funcionalidades, destaca-se o modo *Dev Mode*, que gera informações úteis ao desenvolvedor, como propriedades CSS, valores de espaçamento, cores e tipografias, a partir dos elementos do protótipo. O Figma é multiplataforma e pode ser acessado via navegador, além de possuir versões para Windows e MacOS (FIGMA, 2025).

- **DBDiagram:** Para a modelagem inicial do banco de dados relacional, será utilizada a ferramenta online DBDiagram. Essa plataforma permite a criação rápida de diagramas entidade-relacionamento (ER) com sintaxe simplificada, possibilitando a exportação para SQL ou integração com outras ferramentas de desenvolvimento de banco de dados (DBDIAGRAM, 2025).

- **Git e GitHub:** O controle de versão do código será realizado por meio do Git, utilizando o repositório remoto hospedado no GitHub. Essa combinação de ferramentas permitirá o versionamento eficiente do projeto, o trabalho colaborativo, e a implementação de fluxos de desenvolvimento como *branching* e *pull requests*, além de garantir o histórico de alterações e rastreabilidade das modificações (CHACON; STRAUB, 2014).

- **Notion:** Para o gerenciamento das *sprints* e organização das tarefas, será utilizado o Notion. Essa ferramenta permite a criação de quadros Kanban, listas de tarefas e a documentação de decisões técnicas, promovendo a gestão ágil do desenvolvimento (NOTION, 2025).

- **Docker:** Para garantir a portabilidade e a reprodutibilidade do ambiente de desenvolvimento e produção, será utilizada a tecnologia de *containers Docker*. A aplicação será containerizada, possibilitando sua implantação em diferentes servidores ou máquinas virtuais de forma rápida e consistente (DOCKER, 2025).

- **Sphinx:** Para a documentação técnica da API e dos módulos de IA, será utilizado o Sphinx, um gerador de documentação em formato estático, compatível com o padrão *"reStructuredText"*. O Sphinx permitirá a geração de documentos navegáveis, incluindo

exemplos de uso, *endpoints* disponíveis e descrições detalhadas do funcionamento interno da aplicação (SPHINX. . . , 2025);

- **OpenAPI(Swagger):** O *Swagger* permite que a estrutura de cada *endpoint* da API — incluindo parâmetros de entrada, formatos de resposta, códigos de status HTTP e exemplos de requisição — seja descrita de forma legível tanto por humanos quanto por máquinas. Ao adotar o *Swagger*, será possível gerar automaticamente uma interface interativa via navegador, onde desenvolvedores podem testar os *endpoints* diretamente, enviar requisições e visualizar as respostas em tempo real, sem a necessidade de criar um cliente manual (OPENAPI. . . , 2025).

3.1.1 Desenvolvimento da API RESTful

Para o desenvolvimento da API RESTful, serão usados os seguintes *frameworks* e ferramentas:

3.1.1.1 *Frontend*

A interface visual da aplicação será desenvolvida utilizando:

- **React Native**: Permite o desenvolvimento de aplicações móveis multiplataforma com código único (REACT. . . , 2025);
- **Expo:** Simplifica o processo de construção, empacotamento e execução do aplicativo.
- **Tailwind CSS:** Através do pacote *NativeWind*, que permite aplicar estilos utilitários de forma programática em *React Native* (REACT. . . , 2025; EXPO, 2025; NATIVEWIND, 2025).

3.1.1.2 *Backend*

Para a criação da API e do *backend* do projeto, serão usados os *frameworks*:

- ***TypeScript:*** Adiciona tipagem estática ao *JavaScript*, tornando o desenvolvimento mais seguro e reduzindo erros em tempo de compilação (TYPESCRIPT, 2025);
- ***Fastify:*** *Framework* principal para o desenvolvimento da API RESTful. Reconhecido por sua alta performance e baixo consumo de recursos (FASTIFY. . . , 2025);
- **Objection.js e Knex:** Respectivamente ORM baseado em Modelos e construtor de consultas SQL. Essa combinação facilitará a manipulação de dados no banco relacional escolhido, garantindo flexibilidade e escalabilidade (OBJECTION. . . , 2025; KNEX. . . , 2025);

- **Zod:** Ferramenta de validação de *schemas* baseada em TypeScript, que permite definir, de forma tipada e declarativa, as estruturas esperadas de entrada para cada *endpoint* (ZOD. . . , 2025).

- **PostgreSQL:** Armazenamento das informações de identificação, *embeddings* de referência, logs de processamento e dados de usuários da aplicação (POSTGRESQL, 2025).

3.1.2 Desenvolvimento e Treinamento da IA

Ferramentas específicas para manipulação e desenvolvimento das IAs.

3.1.2.1 Bases de Dados para Treinamento e Testes

Para o treinamento e avaliação dos modelos, serão utilizadas bases de dados públicas amplamente reconhecidas na área de reidentificação de pessoas e detecção de objetos, como:

- **DukeMTMC-reID**: Para treinamento e avaliação dos modelos de reidentificação (ZHENG; ZHENG; YANG, 2017). Ela foi derivada do DukeMTMC, um *dataset* original de múltiplas câmeras para vigilância, com anotações adaptadas para tarefas de Re-ID. O DukeMTMC-reID contém cerca de 1.812 identidades únicas, distribuídas em 36.411 imagens, capturadas por 8 câmeras diferentes. O conjunto de dados é dividido em 16.522 imagens para treinamento, 2.228 para consulta e 17.661 para a galeria de teste. Uma das principais características do DukeMTMC-reID é a grande variação de ângulos de câmera, iluminação e poses, o que o torna um *benchmark* desafiador e amplamente adotado em pesquisas recentes de Re-ID.

- **COCO**: Para tarefas relacionadas à detecção de pessoas no vídeo (LIN *et al.*, 2014). O COCO (*Common Objects in Context*) é uma das bases de dados mais utilizadas no treinamento de modelos de detecção de objetos, incluindo pessoas. Ele contém mais de 330.000 imagens, das quais cerca de 200.000 são anotadas com mais de 1,5 milhão de instâncias de objetos. Ao todo, o *dataset* abrange 80 categorias de objetos. O COCO oferece anotações detalhadas em formato JSON, incluindo as coordenadas de *bounding boxes*, máscaras de segmentação e pontos-chave, sendo amplamente utilizado em *benchmarks* de detecção, segmentação e reconhecimento de pessoas.

Essas bases foram escolhidas por sua diversidade, qualidade e ampla adoção em *benchmarks* acadêmicos.

3.1.2.2 Bibliotecas e *Frameworks* de desenvolvimento de IA

Serão utilizadas as seguintes bibliotecas e frameworks de IA:

- **YOLOv11:** Detecção de pessoas nos vídeos (ULTRALYTICS, 2024);
- **PyTorch**: Framework de aprendizado profundo para treinamento e inferência (PYTORCH, 2025);
- **Ultralytics YOLO**: Implementação popular e otimizada do YOLOv11, com suporte a Python (ULTRALYTICS. . . , 2025);
- **CUDA e cuDNN (opcional)**: Caso o treinamento seja realizado em GPU, serão necessárias as bibliotecas CUDA Toolkit e cuDNN para aceleração computacional (NVIDIA Corporation, 2025a; NVIDIA Corporation, 2025b).
- **OSNet (Omni-Scale Network):** Extração de *embeddings* e reidentificação de indivíduos (ZHOU *et al.*, 2019);
- **Torchreid**: Biblioteca *open-source* para experimentos de Re-ID, contendo implementações pré-treinadas de modelos como OSNet, PCB e AGW (TORCHREID. . . , 2025);
- **PyTorch**: Suporte ao treinamento e inferência de redes neurais (PYTORCH, 2025);
- **NumPy e SciPy**: Bibliotecas auxiliares para manipulação de vetores, arrays e cálculo de distâncias (NUMPY, 2025; SCIPY, 2025).

3.1.2.3 Ferramentas para desenvolvimento da API da IA:

- **Python:** Linguagem amplamente utilizada na área de inteligência artificial e aprendizado profundo, devido à sua extensa gama de bibliotecas especializadas para aprendizado de máquina (Python Software Foundation, 2025);
- **FastAPI:** *Framework* leve e de alta performance em Python, usado para a exposição da API de IA e para a conexão *Websocket* (RAMíREZ, 2023);
- **OpenCV:** Biblioteca para processamento de vídeo e manipulação de imagens em tempo real (OPENCV, 2025);
- ***Google Cloud TPU:*** Ferramenta fornecida pela *Google Cloud* para acelerar o processo de treinamento de modelos (GOOGLE. . . , 2025).

3.2 Métodos

O desenvolvimento da solução será dividido em etapas técnicas bem definidas, contemplando desde o treinamento e configuração dos modelos de inteligência artificial até a implementação da API, interface *web* e medidas de segurança de dados.

3.2.1 Detecção de Regiões de Interesse (ROIs)

Será empregado um modelo de detecção de objetos baseado em aprendizado profundo, o YOLOv11. Esse modelo é treinado para localizar e gerar *bounding boxes* ao redor das pessoas em cada *frame* do vídeo. Esses recortes serão posteriormente processados para extração de características, constituindo o ponto de partida do *pipeline* de reidentificação. A base para o treinamento será a COCO, já descrita na seção **3.1.2.1.**

- Parâmetros de Treinamento do YOLOv11:

    1. **Taxa de aprendizado:** Definida inicialmente como 0,001. Este valor foi escolhido para permitir atualizações graduais dos pesos, evitando oscilações durante o treinamento.
    2. ***Batch Size*:** Um lote de 64 amostras será utilizado, equilibrando entre velocidade de processamento e estabilidade da atualização dos gradientes.
    3. **Número de épocas:** Estima-se um ciclo de 100 a 150 épocas, com monitoramento da perda de validação para evitar sobreajuste.
    4. **Divisão de Dados**: O conjunto de dados será separado em 70% para treinamento, 15% para validação e 15% para teste, seguindo práticas comuns na literatura (REDMON *et al.*, 2016).
    5. **Função de perda:** Serão utilizadas as funções padrão da arquitetura YOLOv11: *bounding box regression loss*, *objectness loss* e *classification loss*.
    6. ***Data Augmentation*:** Técnicas como *random horizontal flip*, *scale jittering*, *color jitter* e *random crop* serão aplicadas para aumentar a acurácia do modelo diante de variações de iluminação, ângulo e posição (GONZALEZ; WOODS, 2008).
    7. ***Dropout:*** Embora a arquitetura YOLOv11 minimize o uso de *dropout*, técnicas de regularização como *early stopping* serão empregadas.

Esses parâmetros serão ajustados conforme os resultados preliminares de validação. O objetivo é maximizar a métrica *mAP (mean Average Precision)* na detecção de pessoas. Eles foram escolhidos com base em recomendações da documentação oficial do YOLOv11 e de experimentos descritos na literatura recente (ULTRALYTICS, 2024).

3.2.2 Extração de Embeddings

Para cada pessoa detectada, será aplicado um modelo leve de reidentificação (Re-ID), como o OSNet (ZHOU *et al.*, 2019), que transforma a imagem recortada da pessoa em um vetor numérico (*embedding*).

Como o sistema não parte de um banco de dados pré-existente, será possível fazer *upload* de um *embedding* de referência previamente extraído a partir de uma imagem fornecida pelo usuário final, por exemplo, uma captura de tela, uma imagem frontal da câmera ou uma foto da pessoa a ser rastreada;

Para o treinamento e validação do OSNet, será utilizada a base pública DukeMTMC-reID, já descrita na seção 3.1.2.1.

- Parâmetros de Treinamento da OSNet

    1. **Taxa de aprendizado:** Inicialmente definido como 0,0003 com decaimento progressivo, utilizando o *scheduler* StepLR para reduzir a taxa em um fator de 0,1 a cada 40 épocas. (ZHOU *et al.*, 2019).
    2. **Otimizador:** Função Adam com B1=0.9 de decaimento exponencial da media dos gradientes e B2=0.999 de decaimento exponencial da média dos quadrados dos gradientes, proporcionando melhor convergência em problemas de classificação complexos.
    3. ***Batch Size*:** Um lote de 64 amostras será utilizado, equilibrando entre velocidade de processamento e estabilidade da atualização dos gradientes.
    4. **Número de épocas:** Estima-se um ciclo de 100 a 150 épocas, com monitoramento da perda de validação para evitar sobreajuste.
    5. ***Divisão de Dados***: O conjunto de dados será separado em 70% para treinamento, 15% para validação e 15% para teste, como comentado anteriormente.
    6. ***Data Augmentation*:** Técnicas como *random horizontal flip*, *random erasing* e *random crop* serão aplicadas para aumentar a acurácia do modelo diante de variações de iluminação, ângulo e posição.

A imagem de referência fornecida pelo usuário será processada pelo OSNet para gerar o *embedding*, que será salvo como *blob* e depois utilizado para comparação na análise do vídeo.

3.2.3 Comparação Vetorial

O usuário da aplicação (*client*), será responsável por fornecer a imagem de referência do indivíduo que se deseja rastrear, como comentado anteriormente. A cada detecção de pessoa no vídeo, o sistema calculará a distância entre o vetor de referência (utilizando métricas como

distância euclidiana) e os vetores extraídos das pessoas detectadas no vídeo. As menores distâncias indicarão maior similaridade, permitindo identificar recorrências do mesmo indivíduo ao longo do tempo e de diferentes cenas;

A etapa de correspondência (*matching*) entre a imagem de referência e as detecções no vídeo será realizada utilizando métricas de distância vetorial. A principal métrica adotada será a distância euclidiana (L2), calculada conforme a Equação 2:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2}$$

Onde (x) representa o vetor de *embedding* da imagem de referência, e (y) o vetor de uma detecção no vídeo.

3.2.4 Avaliação do desempenho

A avaliação da acurácia da detecção de pessoas e da reidentificação será realizada com base nas seguintes métricas:

- Para Detecção (YOLO) (ULTRALYTICS, 2024):

  1. **mAP (*mean Average Precision*):** Média das precisões em múltiplos níveis de *Intersection over Union* (IoU). Mede a qualidade das caixas geradas.

$$\text{AP} = \int_0^1 p(r)\, dr \tag{3}$$

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \tag{4}$$

     Onde:

     - p(r) é a precisão em função do *recall*;
     - N é o número total de classes.

  2. **IoU *(Intersection over Union*):** Métrica que avalia a sobreposição entre a caixa prevista e a real.

$$\text{IoU} = \frac{\text{Área da Interseção}}{\text{Área da União}} \tag{5}$$

  3. ***Precision* e *Recall*:** Avaliação da taxa de verdadeiros positivos e cobertura de detecção.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7}$$

Onde:

- TP são os verdadeiros Positivos;
- TN são os falsos positivos;
- FP são os falsos positivos.

- Para Re-ID (OSNet) (ZHOU *et al.*, 2019):

1. ***Rank-k Accuracy***: Indica a frequência com que a pessoa correta aparece nas (k) primeiras posições da lista de resultados.

$$\text{Rank-k Accuracy} = \frac{\text{Nº de consultas positivas nas k primeiras posições}}{\text{Número total de consultas}} \tag{8}$$

2. **mINP (*mean Inverse Negative Penalty*):** Mede a robustez do modelo considerando variações nos resultados negativos.

$$\text{INP} = \frac{1}{P} \sum_{i=1}^{P} \frac{1}{r_i} \tag{9}$$

Onde:

- P é o número total de consultas;
- r(i) é o Rank da primeira ocorrência correta para a consulta.

3. **CMC (*Cumulative Matching Characteristic*):** Avalia o desempenho cumulativo da reidentificação por Rank (ZHOU *et al.*, 2019).

As fórmulas dessas métricas estão descritas na literatura (ZHENG *et al.*, 2015; ZHOU *et al.*, 2019) e serão implementadas como *scripts* de validação paralelos ao sistema.

3.2.5 Integração do modelo via API

A comunicação com o sistema será viabilizada por meio de uma API desenvolvida com o *framework* FastAPI. Serão implementados *endpoints* específicos para as seguintes funções:

- /upload-hash: Recebimento da imagem de referência, que será convertida em *embedding*;
- /video-stream: Conexão com transmissões ao vivo, via protocolo *websocket*;

- /upload-video: Envio de arquivos de vídeo para processamento;
- /find/:personId: Requisições de busca por aparições da pessoa de interesse no vídeo.

3.2.6 Desenvolvimento da Interface Web

Nesta etapa, será desenvolvida uma interface gráfica minimalista, conectada a uma API escalável, com o objetivo de viabilizar testes, demonstrações e a utilização prática do sistema por usuários finais. A interface permitirá a visualização dos vídeos processados com marcações visuais nas *bounding boxes*, indicação dos *timestamps* das ocorrências e identificação dos indivíduos reconhecidos no vídeo.

O desenvolvimento da API e da interface seguirá os princípios de *Domain-Driven Design* (DDD), uma abordagem arquitetural proposta por Eric Evans (EVANS, 2003), cujo foco é alinhar a estrutura de software ao domínio de negócio, promovendo um código mais organizado, desacoplado e de fácil manutenção.

O DDD tem como conceitos centrais os seguintes:

- **Interface (*Presentation Layer*):** Essa camada será responsável por receber as requisições dos usuários (via interface *web* ou chamadas de API RESTful) e entregar as respostas apropriadas. Inclui os controladores de rotas, validações iniciais de entrada (em conjunto com o Zod) e os adaptadores que traduzem as requisições externas em comandos compreendidos internamente pela aplicação.
- **Camada de Aplicação (*Application Layer*):** A camada de aplicação será responsável por coordenar o fluxo de execução das operações de negócio, sem implementar regras de negócio propriamente ditas. Ela orquestra os casos de uso, chamando os serviços do domínio, gerenciando transações e controlando o fluxo de dados entre as outras camadas.
- **Camada de Domínio (*Domain Layer*):** Esta camada conterá o núcleo das regras de negócio da aplicação. Aqui estarão definidas as entidades, *value objects*, serviços de domínio, regras de validação específicas e as interfaces dos repositórios. No contexto deste projeto, conceitos como "Pessoa", "Vídeo"e "Usuário" serão modelados como entidades ou objetos de valor.
- **Camada de Infraestrutura (*Infrastructure Layer*):** A camada de infraestrutura será responsável por implementar os detalhes técnicos da aplicação. Isso inclui o acesso ao banco de dados (via Knex e Objection), serviços de armazenamento de arquivos (como os vídeos e *embeddings*) e comunicação com a IA (incluindo o disparo das detecções via Python). Essa camada também será responsável pela integração com bibliotecas externas, como o Sphinx para documentação e ferramentas de criptografia para proteção de dados sensíveis.

Ao adotar o DDD, o projeto garantirá que suas regras de negócio sejam expressas de forma clara e independente da camada de infraestrutura, facilitando futuras expansões, como a adição de novas fontes de vídeo, suporte a múltiplos algoritmos de reidentificação ou a integração com sistemas externos.

3.2.7 Criptografia

Para garantir a segurança e a privacidade dos dados dos usuários, será implementado um sistema de *hashing* dos *embeddings* de referência, utilizando o algoritmo **SHA-256.** Dessa forma, os *embeddings* nunca serão armazenados em formato bruto.

3.2.8 Criar a documentação

A documentação técnica da API será gerada utilizando o Sphinx, com formato de saída em HTML estático, integrando exemplos de uso, explicações de cada *endpoint* e detalhes sobre o funcionamento interno dos módulos de IA. Além disso, será criado um *OpenAPI Specification (Swagger)* para facilitar o consumo da API por desenvolvedores externos.

4 RESULTADOS PARCIAIS

4.1 Organização da Base de dados

4.1.1 Base de Dados para Detecção de Pessoas – COCO

A detecção de pessoas será treinada utilizando o COCO (*Common Objects in Context*) (LIN *et al.*, 2014), um dos maiores e mais utilizados *datasets* de detecção de objetos em visão computacional.

A base foi organizada em três conjuntos:

1. 70% para Treinamento: Imagens usadas para atualização dos pesos do YOLOv11.
2. 15% para Validação: Usadas para ajuste de hiperparâmetros e evitar overfitting.
3. 15% para Teste: Exclusivas para avaliação final da acurácia da detecção.

4.1.2 Base de Dados para Reidentificação – DukeMTMC-reID

Para o módulo de reidentificação de pessoas, a base DukeMTMC-reID foi escolhida por sua ampla adoção na comunidade científica e por refletir cenários de vigilância com múltiplas câmeras (ZHENG; ZHENG; YANG, 2017).

A base foi organizada em três conjuntos:

1. 70% para Treinamento: Imagens usadas para atualização dos pesos do YOLOv11.
2. 15% para Validação: Usadas para ajuste de hiperparâmetros e evitar overfitting.
3. 15% para Teste: Exclusivas para avaliação final da acurácia da detecção.

4.1.3 Protótipo de telas

Nesta seção, são apresentados os protótipos de interface elaborados para a aplicação *web* e o painel administrativo do sistema de rastreamento de pessoas. O foco principal foi criar um fluxo de interação simples, seguro e com baixo nível de complexidade operacional, considerando usuários com diferentes níveis de conhecimento técnico. As telas serão projetadas no Figma, seguindo princípios de usabilidade e *design* responsivo.

4.1.3.1 Autenticação e Controle de Acesso

A primeira interação do usuário com o sistema será a tela de autenticação, na qual ele deverá inserir suas credenciais (usuário e senha) para obter acesso ao sistema. O fluxo de

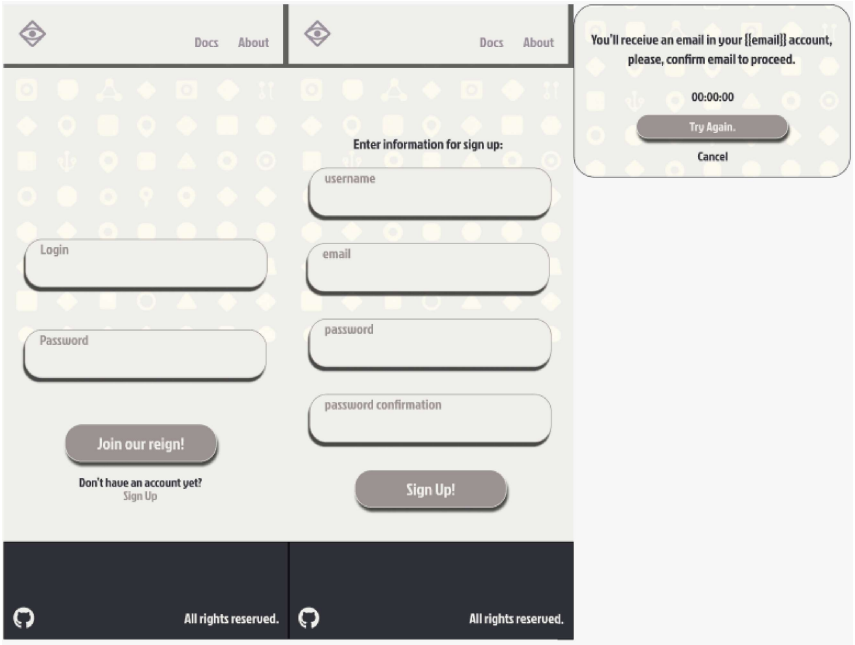

Figura 8 – Telas de Registro e Autenticação.
Fonte: Autoria Própria

autenticação será baseado em um mecanismo de *login* com JWT (*JSON Web Tokens*) armazenado de forma segura em *HTTP-only cookies*, visando mitigar riscos de ataques por XSS. A expiração do *token* será curta, necessitando de renovação periódica através de um *endpoint* específico para *refresh tokens*, aumentando a segurança da sessão.

Além disso, o sistema implementará um esquema de controle de acesso baseado em papéis (*Role-Based Access Control* - RBAC), com as seguintes permissões:

- Usuário do tipo "admin":

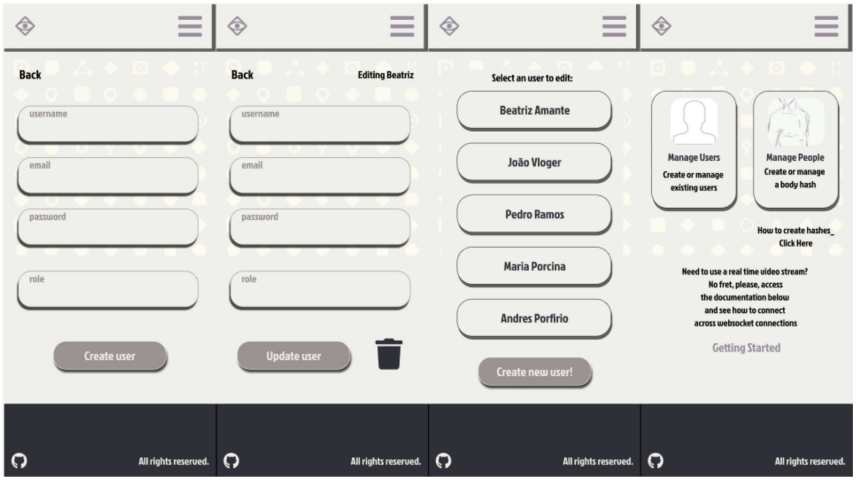

Figura 9 – Telas acessíveis ao Administrador.
Fonte: Autoria Própria

1. Gerenciar todos os usuários cadastrados (criar, editar, excluir);
2. Visualizar todos os vídeos e *embeddings* no sistema;

3. Remover *embeddings* associados a indivíduos.

- Usuário do tipo "user":

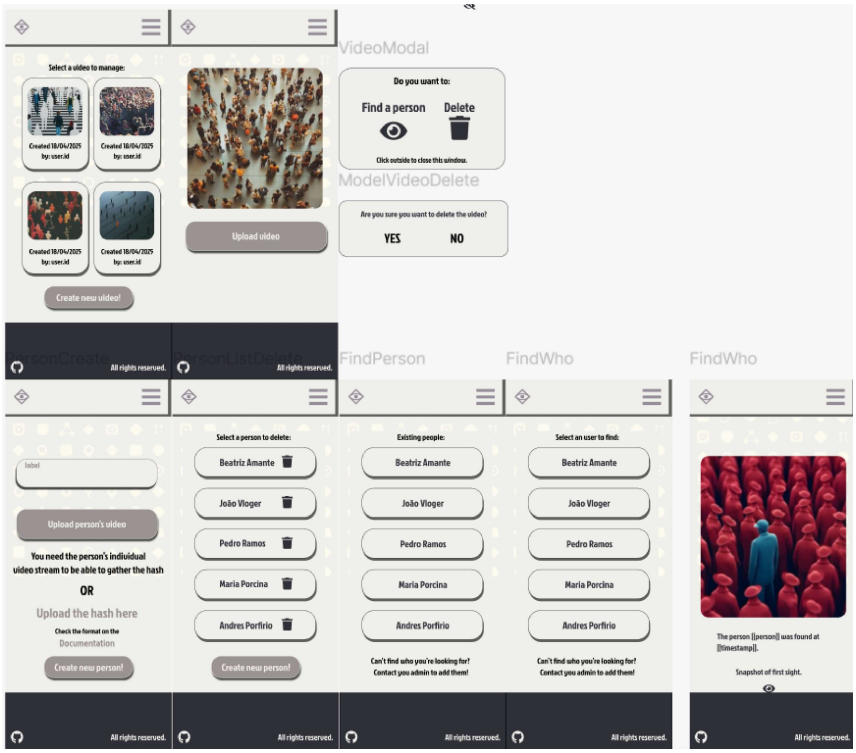

Figura 10 – Telas acessíveis ao Usuário.
Fonte: Autoria Própria

1. Editar apenas seu próprio perfil;
2. Criar e excluir seus próprios vídeos de referência;
3. Adicionar, deletar e buscar pessoas (gerenciar *embeddings* próprios);
4. Realizar buscas por indivíduos em vídeos já enviados ou via transmissão ao vivo.

O sistema garantirá que campos sensíveis, como o tipo de perfil (*role*), não sejam acessíveis para usuários comuns no *frontend*, evitando escalonamentos indevidos de privilégio.

4.1.3.2 Gerenciamento de Vídeos

Após a autenticação, o usuário será redirecionado para a interface principal, onde poderá realizar o *upload de vídeos* ou estabelecer uma conexão via *websocket* para processar transmissões em tempo real.

Usuários poderão:

- Listar os vídeos já enviados;

- Selecionar um vídeo específico para análise de detecção/reidentificação;

A interface permitirá, de maneira intuitiva, associar um vídeo a um processo de detecção ou reidentificação com base nos *embeddings* previamente cadastrados.

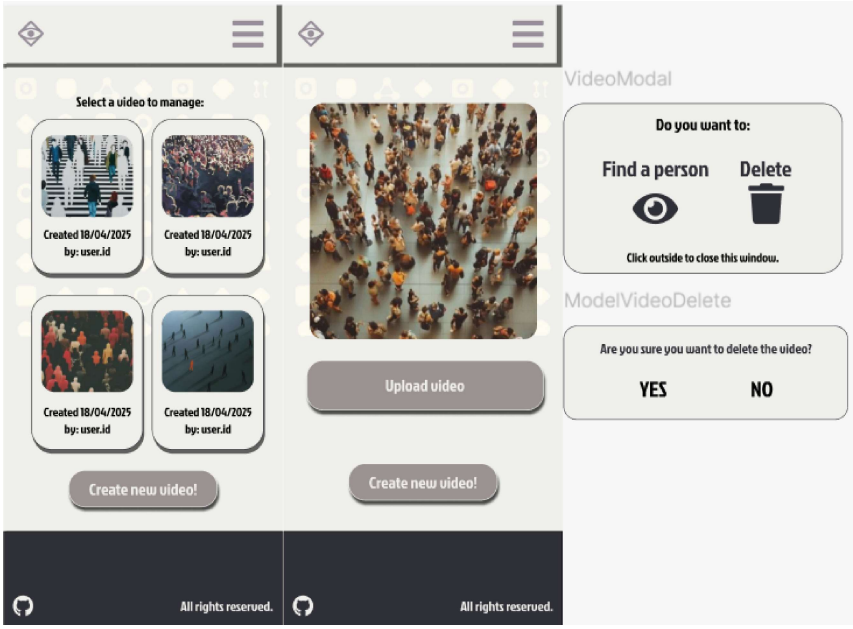

Figura 11 – Gerenciamento de vídeos.
Fonte: Autoria Própria

4.1.3.3 Gerenciamento de Pessoas *(embeddings)*

Nesta tela, o usuário poderá:

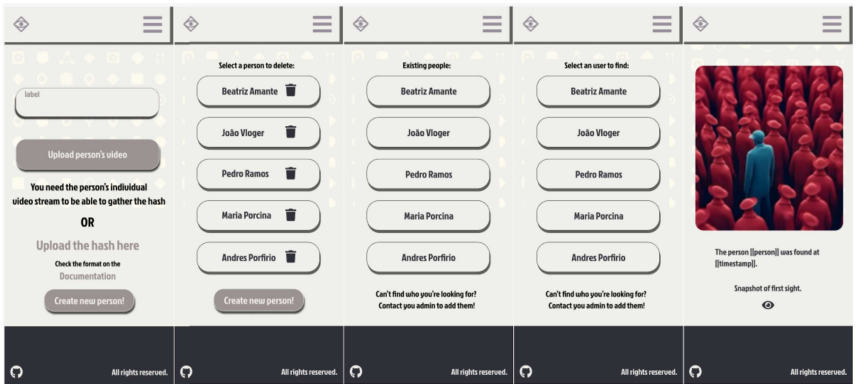

Figura 12 – Gerenciamento de pessoas.
Fonte: Autoria Própria

- Realizar o *upload* de uma imagem de referência (que será convertida em *embedding*);
- Visualizar os *embeddings* cadastrados relacionadas ao usuário logado em questão;
- Excluir imagens de referência já existentes relacionadas ao usuário logado em questão.

A busca por indivíduos será feita informando selecionando o *embedding* de interesse de uma lista, que será comparado com as detecções presentes nos vídeos selecionados ou na transmissão ao vivo.

4.1.4 Modelagem de banco

A partir dos requisitos funcionais levantados durante a fase de planejamento, foi elaborada a modelagem lógica do banco de dados para suportar as operações essenciais da aplicação de rastreamento de pessoas. O objetivo principal desta modelagem foi garantir uma estrutura que atendesse tanto às regras de negócio relacionadas ao gerenciamento de usuários e vídeos quanto às necessidades específicas da manipulação de *embeddings* para reidentificação de pessoas.

A estrutura foi dividida nas seguintes tabelas principais:

- users: Responsável pelo armazenamento de informações dos usuários do sistema;
- people: Essa tabela armazena os registros das pessoas que o usuário deseja rastrear. Cada entrada é associada a um userId, garantindo que cada usuário só tenha acesso às suas próprias pessoas cadastradas (exceto no caso de admins);
- profilePicture: Tabela destinada a armazenar o caminho para a imagem de referência de cada pessoa, vinculada ao userId;
- videos: Responsável por armazenar os vídeos enviados pelos usuários. Inclui o *path* do vídeo no servidor, o userId de quem realizou o *upload* e os campos de data de criação e atualização. Essa separação é essencial para manter a rastreabilidade de quais usuários enviaram quais vídeos, bem como para futuras consultas de análise.

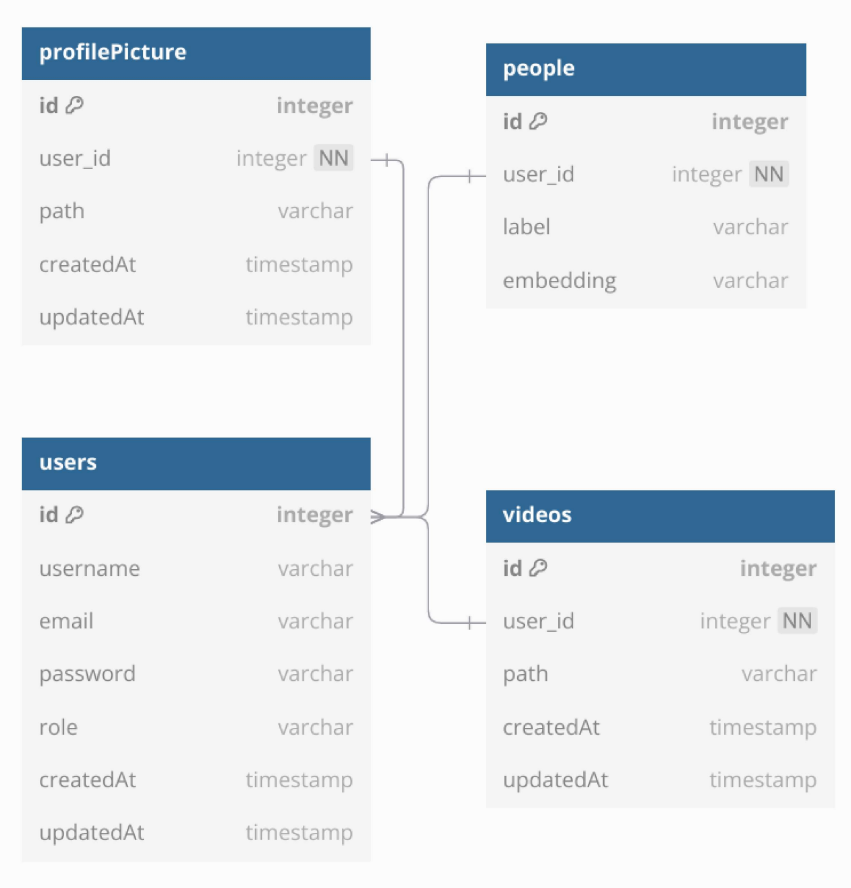

Figura 13 – Modelagem do banco de dados.
Fonte: Autoria Própria

5 CONSIDERAÇÕES FINAIS

Conclui-se que este projeto busca entregar uma solução funcional, acessível e escalável para o rastreamento de pessoas por vídeo com o uso de inteligência artificial, por meio do desenvolvimento de uma API *end-to-end*. A proposta se fundamenta na construção de uma ponte entre a complexidade técnica do reconhecimento visual baseado em redes neurais e a necessidade de interfaces simplificadas que viabilizem o uso por profissionais não especialistas em IA.

Especificamente, pretende-se disponibilizar uma API bem documentada, com *endpoints* REST organizados, exemplos de requisição e retorno em formato estruturado (JSON), além de uma interface visual intuitiva, que permita o teste e a validação das funcionalidades sem exigir conhecimento avançado em programação ou aprendizado de máquina.

O projeto procura superar dificuldades recorrentes na adoção de tecnologias de rastreamento automatizado, como a ausência de ferramentas acessíveis, a falta de documentação clara em muitos *frameworks* existentes e a complexidade na integração entre modelos de detecção, rastreamento e reidentificação. Ao focar em modularidade, privacidade e facilidade de uso, o sistema proposto visa democratizar o acesso a essas tecnologias, viabilizando sua aplicação em contextos diversos — da segurança pública ao uso doméstico.

Além disso, espera-se que o protótipo desenvolvido desperte o interesse da comunidade acadêmica e técnica para novas aplicações e extensões do sistema, como a inclusão de múltiplos alvos, alertas em tempo real, integração com câmeras IP, e *dashboards* analíticos para monitoramento contínuo. Em um cenário onde a privacidade dos dados é cada vez mais relevante, a proposta também levanta discussões importantes sobre ética e segurança, reforçando a necessidade de ferramentas tecnológicas aliadas a boas práticas de proteção da informação.

REFERÊNCIAS

AL-JABERY, K. K. *et al.* 3 - clustering algorithms. *In*: AL-JABERY, K. K. *et al.* (Ed.). **Computational Learning Approaches to Data Analytics in Biomedical Applications**. Academic Press, 2020. p. 29–100. ISBN 978-0-12-814482-4. Disponível em: https://www.sciencedirect.com/science/article/pii/B9780128144824000036.

ALMASAWA, M.; ELREFAEI, L.; MORIA, K. A survey on deep learning based person re-identification systems. **IEEE Access**, PP, p. 1–1, 12 2019.

Amazon Web Services. **O que é uma Rede Generativa Adversária (GAN)?** 2023. Acesso em: 30 abr. 2025. Disponível em: https://aws.amazon.com/pt/what-is/gan/.

BERGMANN, P.; MEINHARDT, T.; LEAL-TAIXE, L. Tracking without bells and whistles. *In*: **2019 IEEE/CVF International Conference on Computer Vision (ICCV)**. IEEE, 2019. p. 941–951. Disponível em: http://dx.doi.org/10.1109/ICCV.2019.00103.

BERLYAND, L.; JABIN, P. **Mathematics of Deep Learning: An Introduction**. Berlin; Boston: De Gruyter, 2023. ISBN 9783111024318.

CHACON, S.; STRAUB, B. **Pro Git**. Apress, 2014. Disponível em: https://git-scm.com/book/en/v2.

CHEN, M. *et al.* Person re-identification by pose invariant deep metric learning with improved triplet loss. **IEEE Access**, PP, p. 1–1, 11 2018.

DBDIAGRAM. 2025. https://dbdiagram.io/. Acesso em: 17 jun. 2025.

DOCKER. 2025. https://www.docker.com/. Acesso em: 17 jun. 2025.

EVANS, E. **Domain-driven design: tackling complexity in the heart of software**. [*S.l.*]: Addison-Wesley Professional, 2003.

EXPO. 2025. https://expo.dev/. Acesso em: 17 jun. 2025.

FASTIFY Web Framework. 2025. https://fastify.dev/. Acesso em: 17 jun. 2025.

FETTE, I.; MELNIKOV, A. **The WebSocket Protocol**. 2011. https://datatracker.ietf.org/doc/html/rfc6455. RFC 6455.

FIELDING, R. T. **Architectural styles and the design of network-based software architectures**. 2000. Tese (Doutorado) — University of California, Irvine, 2000.

FIGMA. 2025. https://www.figma.com/. Acesso em: 17 jun. 2025.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. Upper Saddle River, N.J.: Prentice Hall, 2008. ISBN 9780131687288.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA: MIT Press, 2016. ISBN 9780262035613. Disponível em: https://www.deeplearningbook.org.

GOOGLE Cloud TPU. 2025. https://cloud.google.com/tpu. Acesso em: 17 jun. 2025.

JUNIOR, M. A. d. S.; MARTINI, J. S. C. **Utilização eficiente em larga escala de reconhecimento facial para análise preditiva de segurança em cidades inteligentes**. 2019. Dissertação (Mestrado) — Universidade de São Paulo, 2019.

KARMAKAR, A.; MISHRA, D. **Pose Invariant Person Re-Identification using Robust Pose-transformation GAN.** 2021. Disponível em: https://arxiv.org/abs/2105.00930.

KNEX.JS. 2025. https://knexjs.org/. Acesso em: 17 jun. 2025.

LI, X. *et al.* Deep metric learning for few-shot image classification: A review of recent developments. **Pattern Recognition**, v. 138, p. 109381, 2023. ISSN 0031-3203. Disponível em: https://www.sciencedirect.com/science/article/pii/S0031320323000821.

LIN, T.-Y. *et al.* Microsoft coco: Common objects in context. *In*: SPRINGER. **European conference on computer vision (ECCV).** [*S.l.*], 2014. p. 740–755.

MCLAUGHLIN, N.; RINCON, J. Martinez del; MILLER, P. Recurrent convolutional network for video-based person re-identification. *In*: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).** [*S.l.*: *s.n.*], 2016. p. 1325–1334.

MELO, P. V.; SERRA, P. Tecnologia de reconhecimento facial e segurança pública nas capitais brasileiras: Apontamentos e problematizações. **Comunicação e Sociedade**, n. 42, 2022. Postado online em 16 dez. 2022. Acesso em: 30 abr. 2025. Disponível em: http://journals.openedition.org/cs/8111.

NATIVEWIND. 2025. https://www.nativewind.dev/. Acesso em: 17 jun. 2025.

NOTION. 2025. https://www.notion.so/. Acesso em: 17 jun. 2025.

NUMPY. 2025. https://numpy.org/. Acesso em: 17 jun. 2025.

NVIDIA Corporation. **CUDA Toolkit Documentation.** 2025. https://developer.nvidia.com/cuda-toolkit. Acesso em: 17 jun. 2025.

NVIDIA Corporation. **NVIDIA cuDNN: GPU Accelerated Deep Learning.** 2025. https://developer.nvidia.com/cudnn. Acesso em: 17 jun. 2025.

OBJECTION.JS. 2025. https://vincit.github.io/objection.js/. Acesso em: 17 jun. 2025.

OPENAPI Specification (Swagger). 2025. https://swagger.io/specification/. Acesso em: 17 jun. 2025.

OPENCV. 2025. https://opencv.org/. Acesso em: 17 jun. 2025.

POSTGRESQL. 2025. https://www.postgresql.org/. Acesso em: 17 jun. 2025.

Python Software Foundation. **Python Programming Language.** 2025. https://www.python.org/. Acesso em: 17 jun. 2025.

PYTORCH. 2025. https://pytorch.org/. Acesso em: 17 jun. 2025.

RAMACHANDRAN, G. R. Modern api design: Ai-first architecture, event-driven patterns, and zero-trust security. **International Journal of Computer Trends and Technology**, v. 72, n. 11, p. 220–227, 2024.

RAMíREZ, S. **FastAPI: Modern, fast (high-performance), web framework for building APIs with Python 3.7+.** 2023. Acesso em: 30 abr. 2025. Disponível em: https://fastapi.tiangolo.com.

REACT Native. 2025. https://reactnative.dev/. Acesso em: 17 jun. 2025.

REDMON, J. *et al.* You only look once: Unified, real-time object detection. *In*: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).** [*S.l.*: *s.n.*], 2016.

SCIPY. 2025. https://scipy.org/. Acesso em: 17 jun. 2025.

SILVA, M. L. S. **As tecnologias de reconhecimento facial para Segurança Pública no Brasil: perspectivas regulatórias e a garantia de Direitos Fundamentais.** 2022. Monografia (Graduação em Direito) - Universidade Federal do Rio Grande do Norte, Natal, 87f.

SPHINX Documentation Generator. 2025. https://www.sphinx-doc.org/. Acesso em: 17 jun. 2025.

SRIVASTAVA, N. *et al.* Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: http://jmlr.org/papers/v15/srivastava14a.html.

STANDARDS, N. I. of; TECHNOLOGY. **Secure Hash Standard (SHS).** [*S.l.*], 2015. Acesso em: 04 maio 2025. Disponível em: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

SUN, Y. *et al.* Learning part-based convolutional features for person re-identification. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 43, n. 3, p. 902–917, 2021.

TORCHREID: Person Re-identification Framework. 2025. https://kaiyangzhou.github.io/deep-person-reid/. Acesso em: 17 jun. 2025.

TYPESCRIPT. 2025. https://www.typescriptlang.org/. Acesso em: 17 jun. 2025.

ULTRALYTICS. **YOLOv11 Documentation.** 2024. https://docs.ultralytics.com/. Acesso em: Junho de 2025.

ULTRALYTICS YOLO. 2025. https://github.com/ultralytics/ultralytics. Acesso em: 17 jun. 2025.

WEI, Y. *et al.* Deep learning for retail product recognition: Challenges and techniques. **Computational Intelligence and Neuroscience**, v. 2020, p. 8875910, 2020.

WOJKE, N.; BEWLEY, A.; PAULUS, D. Simple online and realtime tracking with a deep association metric. *In*: **2017 IEEE International Conference on Image Processing (ICIP)**. IEEE Press, 2017. p. 3645–3649. Disponível em: https://doi.org/10.1109/ICIP.2017.8296962.

WU, J. *et al.* **Segment Anything Model is a Good Teacher for Local Feature Learning.** 2024. Disponível em: https://arxiv.org/abs/2309.16992.

YADAV, S. K. *et al.* Yognet: A two-stream network for realtime multiperson yoga action recognition and posture correction. **Knowledge-Based Systems**, v. 250, p. 109097, 2022. ISSN 0950-7051. Disponível em: https://www.sciencedirect.com/science/article/pii/S095070512200541X.

ZHENG, L. *et al.* Scalable person re-identification: A benchmark. *In*: **2015 IEEE International Conference on Computer Vision (ICCV).** [*S.l.*: *s.n.*], 2015. p. 1116–1124.

ZHENG, Z.; ZHENG, L.; YANG, Y. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. **arXiv preprint arXiv:1701.07717**, 2017.

ZHOU, K. *et al.* Omni-scale feature learning for person re-identification. *In*: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).** [*s.n.*], 2019. p. 3707–3716. Disponível em: https://arxiv.org/abs/1905.00953.

ZOD - TypeScript-first schema validation with static type inference. 2025. https://zod.dev/. Acesso em: 17 jun. 2025.