CONTAGEM E IDENTIFICAÇÃO DE PESSOAS EM SALA DE AULA ATRAVÉS DE VISÃO COMPUTACIONAL E INTERNET DAS COISAS

COUNTING AND IDENTIFICATION OF PEOPLE IN A CLASSROOM USING COMPUTER VISION AND INTERNET OF THINGS

Marlon Gonçalves Duarte *

Prof. Dr. Filipe Fernandes dos Santos Brasil de Matos † Prof. Me. Marciel Barros Pereira. ‡

RESUMO

A frequência escolar é essencial para uma gestão educacional eficiente. Métodos tradicionais são menos eficazes em comparação com tecnologias que, além de poupar tempo e esforço dos professores, permitem a extração de novos conhecimentos a partir dos dados gerados. A Inteligência Artificial, com técnicas como Aprendizado de Máquina, Processamento de Linguagem Natural e Visão Computacional, passou a integrar o cotidiano das pessoas, mas tarefas complexas como essas exigem o *offloading* computacional para dispositivos mais poderosos, como na Computação em Nuvem, devido às limitações dos dispositivos de Internet das Coisas. Este trabalho adota uma abordagem de Visão Computacional para reconhecimento facial de alunos em sala de aula, visando realizar a frequência escolar de forma automatizada. A proposta envolve um sistema baseado em um *Raspberry Pi* com uma câmera que captura imagens, as quais serão processadas pelo próprio dispositivo e/ou enviadas para um dispositivo de borda com maior poder computacional. Uma vez que o conjunto de imagens ou dados característicos esteja no dispositivo de borda, ou ainda no *Raspberry Pi*, o sistema oferecerá diversos modelos de detecção e reconhecimento facial para a realização da frequência e posterior registro em um banco de dados. Neste trabalho, foram utilizados os modelos *Single Shot MultiBox Detector* e *ArcFace*, que trabalham em conjunto para detectar rostos e realizar o reconhecimento facial, respectivamente. Os experimentos foram conduzidos como uma prova de conceito e um teste de estresse computacional. Os resultados obtidos demonstram que o sistema atingiu 85% de acurácia nos reconhecimentos. O *offloading* computacional se mostrou uma opção válida, embora o *Raspberry Pi* tenha apresentado tempo médio de 4,05 segundos para a conclusão da tarefa.

Palavras-chave: Reconhecimento Facial. *Offloading*. Frequência Escolar. Computação de Borda.

* Graduando em Ciência da Computação.
† Orientador
‡ Coorientador.

ABSTRACT

School attendance is essential for efficient educational management. Traditional methods seem less effective than technologies that, in addition to saving teachers' time and effort, allow the extraction of new insights from the data generated. Artificial Intelligence, with techniques such as Machine Learning, Natural Language Processing, and Computer Vision, has become part of people's daily lives. However, complex tasks like these require computational offloading to more powerful devices, such as in Cloud Computing, due to the limitations of Internet of Things devices. This work adopts a Computer Vision approach for facial recognition of students in the classroom, aiming to automate the school attendance process. The proposed system is based on a Raspberry Pi with a camera that captures images, which will be processed by the device itself or sent to an edge device with greater computational power. Once the set of pictures or characteristic data is on the edge device or the Raspberry Pi, the system will offer various facial detection and recognition models to perform attendance and subsequent logging into a database. This work used the Single Shot MultiBox Detector and ArcFace models, which work together to detect faces and perform facial recognition, respectively. The experiments were conducted as a proof of concept and a computational stress test. The results show that the system achieved 85% accuracy in recognitions. Computational offloading proved a valid option, although the Raspberry Pi presented an average time of 4.05 seconds to complete the task.

Keywords: Face Recognition; Offloading. School Attendance. Edge Computing.

1 INTRODUÇÃO

A frequência escolar desempenha um papel importante na educação para alunos, professores e pais. A chamada tradicional, nome a nome, parece desgastante, ineficiente e consome tempo de aula. Além disso, pode interromper o fluxo da aula, desmotivar os alunos pela sensação de desconfiança gerada e, ocasionalmente, expor alunos a situações desconfortáveis, tais como ser chamado por um nome cujo qual não lhe representa ou pelo professor não o saber pronunciar. No Brasil, Paiva e Silva (2020) relataram que, em uma escola profissionalizante, cerca de 31,45% do tempo de aula foi dedicado a atividades não pedagógicas. Apenas 64% do tempo da classe foi usado para transmissão de conteúdo devido a interrupções, como a chamada, resultando na perda de até um dia de ensino por semana. Zamoner (2005) destaca que gestores escolares devem minimizar essas interrupções, implementando campanhas de conscientização e ajustando procedimentos administrativos. Na maioria das vezes, os profissionais de educação ainda fazem uso de papel e caneta para a realização da chamada, dispensando a enorme quantidade de recursos tecnológicos que existem para ajudar nesta tarefa.

O avanço tecnológico permitiu a criação de modelos de Inteligência Artificial (IA). Russell (2010) define IA como a capacidade de um sistema realizar tarefas que seriam consideradas inteligentes se feitas por humanos. A IA usa técnicas como *Machine Learning* (ML), *Natural Language Processing* (NLP), *Evolutionary Algorithms* (EA), *Reinforcement Learning* (RL) e Processamento de Sinais (RUSSELL, 2010). Dentro do conjunto de campos da IA, a visão computacional, que baseia-se em ML, permite ao computador "ver" o mundo processando informações de imagens digitais permitindo aos sistemas reconhecer padrões em *pixels* de imagens (MILANO; HONORATO, 2014).

A visão computacional tem o potencial de transformar a vida das pessoas em diversas aplicações. No entanto, quanto maior a qualidade buscada nesses sistemas, maiores são os custos computacionais, tanto em termos de armazenamento quanto de processamento. Para Saavedra *et al.* (2022), computadores pessoais podem não ser eficientes para certos tipos de computação, especialmente no reconhecimento facial, que demanda respostas instantâneas ou grande capacidade de cálculo durante o treinamento. A precisão de um modelo de reconhecimento facial frequentemente depende de grandes volumes de dados, exigindo armazenamento e processamento robustos (HOSAKI; RIBEIRO, 2021). Esse cenário gera a necessidade de direcionar essas operações para sistemas mais potentes por meio de *offloading*.

O *offloading* transfere tanto dados quanto tarefas complexas para dispositivos mais poderosos, melhorando o desempenho, economizando energia e liberando recursos do dispositivo original.. Sua popularização deve-se a *Cloud Computing* (CC), uma técnica de computação na qual serviços de Tecnologia da Informação (TI) são fornecidos sob demanda por meio de unidades computacionais massivas e de baixo custo. Essas unidades são conectadas por redes *Internet Protocol* (IP) e suas principais características incluem alta escalabilidade, elasticidade e o uso de um pool de recursos compartilhados, combinando infraestrutura virtualizada e física para otimizar o uso (QIAN *et al.*, 2009). Não obstante, também contam com agendamento dinâmico de recursos para alocá-los de forma eficiente conforme as necessidades momentâneas, promovendo a eficiência, reduzindo custos e impulsionando a inovação (CHANDRASEKHARAN, 2014). Embora a CC ofereça grande escalabilidade, apresenta latências críticas em respostas sensíveis ao tempo, como no reconhecimento facial, sendo que tecnologias como *Fog Computing* (FC) ou *Edge Computing* (EC) são cruciais para descentralizar o processamento e posicionar os recursos mais próximos dos dados, acelerando o envio, processamento e resposta, o que reduz latências e garante maior eficiência das aplicações (DE, 2016).

Com as possibilidades de conexões mais eficientes e transferências de trabalhos para

dispositivos mais poderosos, uma nova tecnologia ganha destaque, a Internet das Coisas (IoT). A mesma surgiu dos avanços em sistemas computacionais menores e portáteis, microeletrônica, comunicação e sensoriamento, atraindo atenção pelo seu potencial. A IoT é uma extensão da Internet que permite a conexão de objetos cotidianos com capacidade computacional e de comunicação, possibilitando o controle remoto e a oferta de serviços. Apesar das oportunidades acadêmicas e industriais, as limitações dos computadores pessoais aqui se destacam ainda mais (SANTOS *et al.*, 2016).

A automatização dos processos educacionais pode otimizar o tempo de aula, com a aplicação de IoT em tarefas como o gerenciamento de projetores e o controle de luzes, entre outros. O reconhecimento facial surge como uma solução promissora para automatizar o registro de presença dos alunos. Em ambientes universitários, essa tecnologia pode ser utilizada para contar os alunos em sala e realizar o controle de frequência de forma inteligente. No entanto, dispositivos menos robustos, como os da IoT, podem não ser poderosos o suficiente para essa tarefa, dada a complexidade do processamento de imagens e a necessidade de respostas em tempo real. Isso indica a necessidade de implementar estratégias como EC e *offloading*. Assim, é importante o desenvolvimento de um sistema que auxilie na gestão escolar, contribuindo para um uso mais eficiente do tempo de aula.

Este trabalho teve como objetivo desenvolver um sistema de frequência escolar com visão computacional através de um pipeline moderno de detecção facial, integrando recursos de IoT e EC para garantir eficiência em termos de tempo para consolidação da frequência. Para tanto, conduziu-se uma análise do estado atual dos sistemas de visão computacional, especialmente aqueles voltados para a frequência escolar. Em seguida, foi proposta uma modelagem de um sistema que integrasse de forma eficaz os recursos de EC. A implementação envolveu um sistema de visão computacional baseado em um dispositivo *Raspberry Pi 4 Model B*, com ênfase na coleta de imagens em ambientes de borda e na transferência eficiente de dados para um dispositivo de borda adicional. Por fim, o sistema foi testado em um ambiente com interação de pessoas, além de submetido a testes de estresse computacional. O sistema demonstrou-se viável quanto à execução em um dispositivo de borda levando em torno quatro segundos para um reconhecimento, além de apresentar resultados promissores no uso de um dispositivo de borda para ajuda no reconhecimento facial. Este último apresentou tempo de resposta, para até 5 salas de aula simultâneas, menores que o tempo gasto pelo dispositivo IoT.

As principais contribuições deste trabalho são:

1. A criação de um sistema para realização da frequência escolar de forma a aliviar a carga laboral do professor e otimizar o processo a fim de uma maior eficiência em termos de documentação do ambiente de sala de aula;
2. Avaliação da capacidade de um dispositivo IoT realizar a tarefa de reconhecer rostos de pessoas e processar esses dados internamente. Bem como, benefícios de utilizar offloading para transferir a tarefa de reconhecimento facial a um dispositivo de borda.

2 FUNDAMENTAÇÃO TEÓRICA

A presente seção apresentará uma base necessária ao entendimento do sistema proposto neste trabalho. Dentre os assuntos que serão abordados nesta parte do trabalho estão: IA, *Computer Vision* (CV), IoT e *Mobile Cloud Computing* (MCC).

2.1 Inteligência Artificial (IA)

A conceptualização da inteligência apresenta desafios substanciais em termos de definição precisa, refletindo-se, consequentemente, na complexidade inerente à delimitação da IA. Coppin (2004) aborda que é o campo que se concentra na pesquisa de sistemas que se comportam de tal forma que, para um observador, suas ações seriam percebidas como inteligentes. Já para Russell (2010), a IA é um campo de estudo que busca compreender e criar entidades inteligentes com base no que a inteligência é e proporciona aos seres humanos. É através da inteligência humana que se pode manipular o mundo e facilitar a realização de diversas tarefas. A IA é um campo relativamente recente, com origens pós-Segunda Guerra Mundial, e tem atraído o interesse de profissionais de diversas áreas do conhecimento, sobretudo nos últimos anos, com o surgimento de diversas plataformas online que oferecem serviços através de IA (BARBOSA, 2020).

Atualmente, a IA abrange uma ampla gama de subcampos e é aplicável a tarefas intelectuais diversas, tornando-se um campo universal em constante expansão. Ela está intrinsecamente ligada à ação racional, na qual um agente inteligente busca tomar a melhor decisão possível com base na situação. Esse agente pode ser qualquer entidade capaz de perceber seu ambiente por meio de sensores e agir sobre ele usando atuadores (RUSSELL, 2010). Por exemplo, um agente humano utiliza órgãos sensoriais, como olhos e ouvidos, e partes do corpo, como mãos e pernas, como atuadores. Algumas técnicas foram desenvolvidas para tornar os agentes inteligente e capazes de reagir de acordo com o que se espera. Uma dessas técnicas é o Aprendizado de Máquina.

2.2 Aprendizado de Máquina

Segundo Russell (2010), Aprendizado de Máquina envolve treinar sistemas para generalizar (enfatizar aquilo que mais comumente se apresenta com um padrão) a partir de dados rotulados e classificar novos dados. Não é muito diferente da definição de Coppin (2004) quando defende que é a capacidade de um sistema computacional aprender e melhorar seu desempenho em uma tarefa com base em dados, sem ser explicitamente programado para tal. A IBM (2023) define Aprendizado de Máquina como um ramo da inteligência artificial e ciência da computação que se concentra no uso de dados e algoritmos para imitar a forma como os humanos aprendem, melhorando gradualmente sua precisão. O Aprendizado de Máquina é um componente importante da crescente área de ciência de dados. Por meio do uso de métodos estatísticos, os algoritmos são treinados para fazer classificações ou previsões, e descobrir informações fundamentais em projetos de mineração de dados. Certos tipos de tarefas necessitam de treinamentos mais complexos assim como o cérebro humano faz para reconhecer padrões. Dentro da IA, as Redes Neurais Artificiais (RNA) tentam realizar essas tarefas.

2.3 Redes Neurais Artificiais

As RNA são modelos computacionais inspirados nas redes de neurônios biológicos presentes no cérebro humano. Elas consistem em unidades simples de processamento interconectadas, os neurônios artificiais, que se comunicam uns com os outros através de conexões ponderadas para a maioria dos projetos (COPPIN, 2004). Os atuais projetos de RNA englobam estruturas complexas que incluem milhões de neurônios dispostos em dezenas a até centenas de camadas, capacitadas a realizar uma vasta variedade de tarefas avançadas de Aprendizado de Máquina (MAGALHÃES *et al.*, 2020).

A rede neural é construída conectando múltiplos neurônios em camadas. Redes neurais de propagação para frente apresentam conexões unidirecionais entre suas camadas de entrada, ocultas e de saída. Por outro lado, redes neurais recorrentes ou de realimentação caracterizam-se por conexões arbitrárias, permitindo retroalimentação entre unidades neurais (RAUBER, 2005).

As redes neurais são robustas e têm a capacidade de generalizar a partir de exemplos, modelando relações complexas não-lineares entre entradas e saídas. Elas encontram aplicações em reconhecimento de padrões, classificação, aproximação de funções, previsão de séries temporais e otimização. Sua inspiração biológica e propriedades de aprendizado as tornam particularmente adequadas para problemas de IA. O uso de RNA permitiu à computação realizar tarefas mais difíceis como o Reconhecimento Facial.

2.4 Reconhecimento Facial

O reconhecimento facial pertence a área da IA e visa identificar ou verificar a identidade de uma pessoa através de características únicas do seu rosto (THORAT *et al.*, 2010). O processo envolve várias etapas, começando com a detecção da face em uma imagem ou vídeo. Em seguida, ocorre o alinhamento da face detectada, geralmente para uma posição frontal*, utilizando técnicas como o alinhamento 3D proposto por Taigman *et al.* (2014), que emprega um modelo 3D genérico para ajustar a posição da face. Na Figura 1, é apresentado o processo do pipeline de detecção e alinhamento segundo Taigman *et al.* (2014), da seguinte forma:

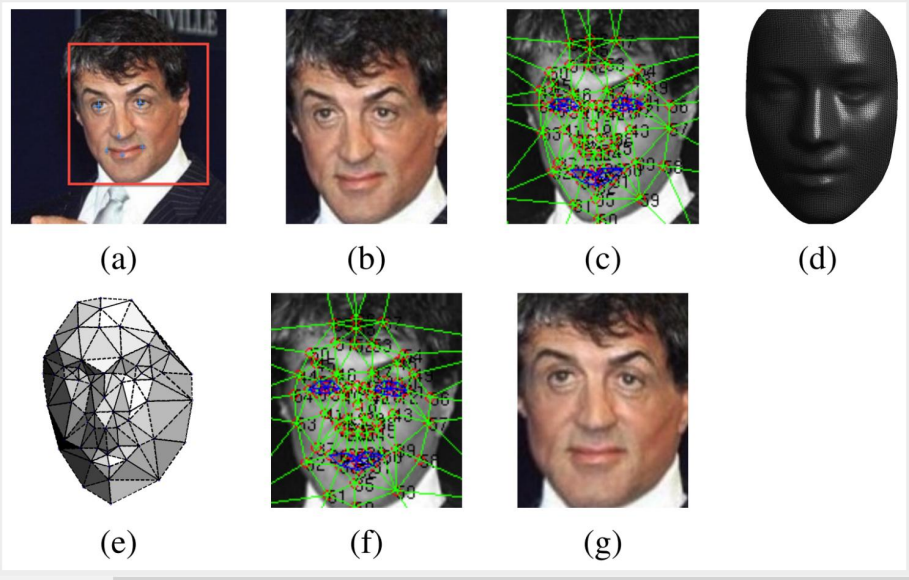
- (a) Detecção da face, com 6 pontos de referência iniciais nos olhos, nariz e boca;
- (b) O corte alinhado em 2D induzido;
- (c) Expande-se então os pontos para 67 no corte alinhado em 2D adicionamos triângulos no contorno para evitar descontinuidades;
- (d) A forma 3D de referência transformada a partir de (b);
- (e) Visualização dos triângulos. Triângulos mais escuros são menos visíveis;
- (f) Os 67 pontos de referência induzidos pelo modelo 3D que são usados para direcionar a deformação;
- (g) O corte final com o rosto alinhado.

Após o alinhamento, o passo sequinte é a extração de características faciais, que pode ser realizada de diferentes maneiras. Taigman *et al.* (2014) utiliza uma *Deep Neural Network* (DNN) com múltiplas camadas, incluindo camadas convolucionais e de *pooling*, para extrair um vetor de características de alta dimensionalidade. Já Schroff *et al.* (2015), emprega uma *Convolutional Neural Networks* (CNN) treinada com uma *Triplet Loss* para mapear diretamente as imagens faciais para um espaço euclidiano compacto. O *Triplet Loss* é uma função de perda que tem como objetivo, minimizar a distância entre uma imagem base e uma classificação correta, ao mesmo tempo em que se maximiza a distância entre a imagem base e a erroneamente classificada (HERMANS *et al.*, 2017). Ambas as abordagens visam capturar detalhes sutis e características discriminativas das faces, e impressiona ao conseguir uma representação mais compacta de apenas 128 bytes por face.

A etapa final do reconhecimento facial envolve a classificação ou verificação da identidade. Isso pode ser feito comparando o vetor de características extraído com um banco de dados de faces conhecidas, utilizando técnicas como *K-Nearest Neighbors* (KNN) ou outros métodos de classificação. A eficácia desses sistemas é geralmente avaliada em conjuntos de dados padronizados, como o *Labeled Faces in the Wild* (LFW), que testam o desempenho em cenários

* Em algumas abordagens pode se encontrar o termo frontalização, por isso o mesmo foi empregado em algumas seções deste trabalho

Figura 1 – Pipeline de detecção e alinhamento facial com a face frontalizada na última etapa



Fonte: Adaptado de (TAIGMAN *et al.*, 2014)

de verificação facial não-restrita (SCHROFF *et al.*, 2015). O treinamento desses modelos requer grandes volumes de dados faciais rotulados, como o conjunto *Social Face Classification* (SFC) usado por Taigman *et al.* (2014), que contém milhões de imagens de faces. Nas próximas subseções serão abordados outros conceitos necessários ao entedimento deste trabalho.

2.5 Internet das Coisas (IoT)

Para Santos *et al.* (2016), a IoT é um conceito que se refere à interconexão de dispositivos físicos, objetos e sistemas por meio da Internet. Essa interconexão permite a coleta, troca e análise de dados, possibilitando a comunicação entre os dispositivos e a realização de ações automatizadas. Em outras palavras, a IoT transforma objetos comuns, permitindo que eles coletem e compartilhem informações para melhorar a eficiência, a conveniência e a tomada de decisões em diversos setores, como saúde, agricultura, indústria, transporte, entre outros Sobin (2020). Essa rede de dispositivos conectados pode incluir desde sensores industriais e dispositivos médicos até eletrodomésticos e veículos, criando um ambiente onde a informação é coletada e utilizada para otimizar processos e resolver problemas do cotidiano. Devido a baixa capacidade de processamento e armazenamento comum a estes dispositivos, as tarefas mais complexas a serem realizadas devem, quase sempre, migrarem para outros dispositivos mais robustos através da técnica de *offloading*.

2.6 *Offloading*

O *offloading* se refere à transferência de dados ou computação de um dispositivo móvel para a nuvem com o objetivo de otimizar a vida útil da bateria e superar as limitações de recursos do dispositivo. Essa ideia que também é, muitas vezes, denominada *cyber foraging*, teve sua origem com (SATYANARAYANAN *et al.*, 2009). Pode acontecer enviando apenas partes selecionadas de um programa para a nuvem. Mas também existe a possibilidade de migrar o programa completo. Essa diferenciação nas estratégias de *offloading* visa maximizar a eficiência do processo. Ao enviar tarefas mais pesadas para a nuvem, a técnica não apenas aprimora o

desempenho, mas também economiza energia, dado que a computação local tende a consumir mais recursos. Dados também podem ser migrados para a nuvem aliviando a carga da memória do dispositivo de origem.

O *offloading* pode ser impulsionado por diferentes objetivos, dependendo das necessidades específicas da aplicação. Para Rego *et al.* (2019) quando a prioridade é melhorar os tempos de processamento, a abordagem é direcionado para reduzir o tempo de execução das aplicações. Se o foco é economia de energia, o *offloading* concentra-se na eficiência energética, buscando reduzir o consumo de energia durante o processamento. Há também situações em que a abordagem é motivada por outros fatores que não estão centrados no tempo de processamento ou na eficiência energética (WANG *et al.*, 2020). Em casos assim, o *offloading* pode ser impulsionado pelo desejo de aumentar a colaboração, estender a capacidade de armazenamento ou reduzir custos financeiros. Essas considerações adicionais destacam a versatilidade desta técnica, que pode ser adaptado de acordo com uma variedade de metas e requisitos específicos.

Figura 2 – Cenário geral de *offloading*

Fonte: O autor com base em Costa (2014).

Dispositivos móveis aprimoram o desempenho de aplicações ao utilizar recursos remotos através de técnicas de *offloading* como mostrou Gomes *et al.* (2017). Essa estratégia pode ser aplicada tanto em uma nuvem pública remota quanto em outro dispositivo dentro da mesma rede local sem fio, conhecido como *Cloudlet*, o qual está conectado à nuvem por meio de uma rede cabeada (SATYANARAYANAN *et al.*, 2009). Para Gomes *et al.* (2017), os uso do *offloading* também otimiza o consumo de energia, importante para garantir que as baterias de alguns dispositivos possam duram mais tempo sem recarga.

A abordagem que incorpora o uso de *Cloudlets* oferece benefícios significativos, resultando em uma melhor qualidade de serviço e na redução do tempo percebido de execução. Esse aprimoramento se deve à maior velocidade de transmissão de dados dentro de uma rede *Wi-Fi* de um único salto, em comparação com a execução em uma nuvem pública remota (SANTOS, 2017). Nesse cenário, os dispositivos móveis realizam *offloading* para recursos remotos através da rede *Wi-Fi*, e esses recursos podem estar hospedados em computadores próximos dedicados

(*Cloudlets*) ou em uma nuvem pública conforme pode ser visualizado na Figura 2. Na seção seguinte, serão apresentados autores que pesquisaram abordagens que se assemelham à deste trabalho.

3 TRABALHOS RELACIONADOS

Khan *et al.* (2019), apresenta um algoritmo de detecção e reconhecimento facial baseado em CNN, que apresenta desempenho superior em comparação às técnicas tradicionais. Para validar a eficácia do algoritmo, foi construído um sistema de sala de aula inteligente, com o objetivo de automatizar o processo de chamada dos presentes. A metodologia empregou a rede *Faster Region-based Convolutional Neural Network* (RCNN), que utiliza dois fluxos: um para propor regiões candidatas a faces detectadas e outro para reconhecer essas faces. O primeiro fluxo faz uso de uma Rede de Proposta de Regiões (RPR) que desenha âncoras nas imagens, selecionando as 2000 com maior probabilidade de conter faces. Essas âncoras são caixas delimitadoras que indicam áreas a serem verificadas quanto à presença de rostos. O segundo fluxo passa cada região de interesse por uma CNN para extrair características faciais, como a localização de olhos, nariz e boca, textura e cor da pele, expressões faciais, entre outros, classificando as faces. O sistema foi treinado no conjunto de dados LFW, que contém mais de 13.000 imagens de rostos coletadas na web, utilizando técnicas como *dropout* e *early stopping* para evitar *overfitting*. O modelo alcançou 97,9% de acurácia nos dados de validação. Para a produção do *dataset* de características dos rostos dos alunos, foi exigida a submissão de seis imagens de boa qualidade no momento do cadastro, cujas características foram armazenadas para comparação com as imagens capturadas durante a frequência, realizada pelo professor por meio de fotografias da turma. A fim de reduzir a latência, foi utilizada computação de borda, permitindo o processamento mais próximo do usuário. O sistema reconheceu 30 faces em um conjunto de 35 detectadas, com uma precisão média de 94,6% e, em uma avaliação com 3000 pares de faces, obteve uma precisão média de 99,64%, demonstrando que a abordagem baseada em CNN e computação de borda supera as técnicas tradicionais de reconhecimento facial.

Já Khan *et al.* (2020), propõe um sistema automático de chamada escolar sensível ao tempo, baseado em reconhecimento facial. A metodologia envolve a instalação de câmeras nas salas de aula para capturar imagens no início e no final de cada aula. Essas imagens são processadas pelo algoritmo *You Only Look Once* (YOLO) na versão 3, que detecta rostos percorrendo toda a imagem e delimitando cada face encontrada com um retângulo. Além do YOLO, o projeto utiliza uma ferramenta de reconhecimento da *Microsoft (Vision Studio)* para contar os rostos detectados e informar o número de alunos presentes. Para o desenvolvimento do projeto, os autores criaram uma*Grafical User Interface* (GUI) com a biblioteca *TKinter* do *Python* e utilizaram a biblioteca *SQLite* para o armazenamento dos dados, fornecendo recursos básicos de *Structured Query Language* (SQL) para gerenciamento de banco de dados. O sistema realiza duas tarefas principais nas imagens capturadas: a contagem de alunos com o YOLO e o reconhecimento de pessoas. Após a contagem, uma planilha é gerada automaticamente, diferenciando alunos conhecidos e desconhecidos, e o banco de dados é atualizado diariamente. Na etapa de registro dos alunos, foram atribuídos números de registro únicos e os dados foram armazenados na nuvem da *Microsoft Azure*, com a geração de um Identidade (ID) específico. Para criar o banco de dados de reconhecimento facial, várias fotos de cada aluno foram capturadas para treinamento. Cada rosto delimitado nas imagens foi recortado e enviado para reconhecimento pela *Application Programming Interface* (API) de reconhecimento facial, que extraiu características faciais como olhos, nariz e boca, comparando-as com as faces cadastradas no banco de dados. Se um rosto fosse identificado como correspondente a um aluno, ele era marcado como "reconhecido"; caso

contrário, como "não reconhecido". Ao final do processo de reconhecimento, o sistema gerou planilhas de presença, separando os dados dos alunos reconhecidos e dos não reconhecidos. No final de cada mês, a ferramenta gerou um relatório de presença, que foi enviado via *e-mail* a alunos, pais e professores. Os experimentos realizados demonstraram alta precisão na detecção e no reconhecimento facial, sob diferentes condições, alcançando 100% de precisão em muitos casos, o que viabiliza a automatização do monitoramento de presença e elimina o trabalho manual.

O sistema automático de monitoramento de presença em tempo real para salas de reunião proposto por Muttaqin *et al.* (2020), utiliza reconhecimento facial baseado em aprendizado de máquina. A metodologia consistiu na instalação de múltiplas câmeras na sala para capturar imagens no início e no final da reunião, com foco na melhoria da performance do reconhecimento facial, sendo essa a principal contribuição do trabalho. O sistema é composto por três subsistemas: o *Front-End*, que implementa uma interface web desenvolvida em *Vue.js* para exibir os dados de presença capturados; o *Back-End*, que é uma glsAPI *Representational State Transfer* (REST) responsável por armazenar os dados no banco de dados e disponibilizá-los para o *Front-End*; e o Nó, que realiza a captura de vídeo, detecção e reconhecimento facial, utilizando uma rede neural pré-treinada. As imagens capturadas pela *webcam* são processadas pelo *OpenCV*, padronizadas e enviadas para a biblioteca Dlib, que aplica a rede neural para extrair vetores de características faciais e comparar com os dados armazenados, identificando os indivíduos. O sistema gera relatórios de presença dos participantes da reunião, demonstrando desempenho notável com uma taxa de reconhecimento próxima a 100% para indivíduos registrados. Contudo, apresentou desafios ao identificar rostos desconhecidos, com métricas de precisão variando entre 60% e 80% em testes com diferentes combinações de pessoas conhecidas e desconhecidas. O tempo médio para reconhecimento e envio dos dados ao servidor foi de aproximadamente dois segundos, com a detecção facial sendo mais precisa a uma distância mínima de dois metros. O sistema proposto permite a automatização da marcação de presença em reuniões, mas os resultados indicam que ainda é necessária intervenção humana em determinadas situações.

Outro que aborda essa temática, mas não em termos de frequência, é Hussain *et al.* (2022), que propõe um sistema inteligente de controle de acesso médico baseado em reconhecimento facial, utilizando *Deep Learnig* (DL), com o objetivo de aumentar a segurança em ambientes médicos e de saúde de alto risco. Esse sistema evita a disseminação de doenças infecciosas que poderiam se espalhar por meio de ferramentas de autenticação por toque ou que exijam contato físico, além de restringir o acesso a áreas sensíveis. A metodologia adotada pelos autores utiliza um *Raspberry Pi* como controlador principal, conectando-o às portas do ambiente médico, tornando-as inteligentes. O controlador, uma placa bastante compacta e acessível, foi configurado para gerenciar a câmera que captura a imagem facial e a fechadura da porta, utilizando as *General Purpose Input/Output* (GPIO) do dispositivo. O processo de autenticação começa com a captura da imagem facial pela câmera, que é então comparada com as imagens armazenadas em um banco de dados previamente alimentado. A detecção do rosto é feita por meio da técnica *Haar Cascade*, seguida pela extração de características faciais utilizando modelos pré-treinados de CNN (ResNet-50 e VGG-16), em conjunto com o algoritmo *LBP Histogram*. O rosto é, então, classificado como genuíno ou não utilizando *Support Vector Machine* (SVM). Se o rosto for reconhecido como genuíno, a porta é destravada; caso contrário, ela permanece travada, e o sistema notifica o proprietário por *e-mail*, salvando a imagem capturada e as informações no banco de dados para possíveis usos futuros. Os resultados mostraram que a abordagem proposta alcançou 99,56% de acurácia, com o modelo *ResNet-50* destacando-se no desempenho de extração de características e reconhecimento facial. Além disso, o sistema é de baixo custo, fácil de implementar e oferece notificações aos proprietários

sobre possíveis invasores, cumprindo eficazmente o objetivo dos autores.

Após a apresentação de exemplos de trabalhos científicos que exploram uma temática na linha deste, é possível discernir contribuições que cada um desses estudos oferece. O presente trabalho inova ao utilizar a captura da imagem e processamento ainda no dispositivo de IoT para realização da frequência em sala de aula, podendo também transferir o reconhecimento das faces para um dispositivo de borda. O presente trabalho também se mostra inovador por realizar avaliação de uso em ambiente real além de experimentos de estresse computacional. A Tabela 1 mostra as principais contribuições de cada trabalho fortalecendo as características não abordadas por nenhum outro trabalho exceto este.

Tabela 1 – Tabela comparativa entre as contribuições deste trabalho em relação aos outros apresentados.

Autores	Contribuições				
-	1	2	3	4	5
Khan et al. (2019)	X		X		
Khan et al. (2020)	X				X
Muttaqin et al. (2020)	X				
Hussain et al. (2022)		X			
Este trabalho	X	X	X	X	X

1. Chamada em sala sensível ao tempo. 2 - Processamento ainda no *Raspberry Pi*. 3 - Reconhecimento Facial na borda. 4 - Avaliação de estresse computacional do sistema. 5 - Chamada automática a partir do reconhecimento.

4 SISTEMA DE CHAMADA AUTOMÁTICA

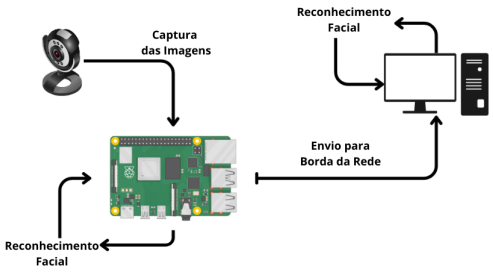
Nesta seção será apresentado o sistema de chamada automática desenvolvido neste trabalho. O objetivo é otimizar o tempo dos professores e gerar dados relevantes no contexto educacional. Para isso, foram utilizadas ferramentas e processos amplamente empregados em TI, como Computação de Borda, mini-computadores, como o *Raspberry Pi*, e visão computacional.

4.1 Arquitetura do Sistema

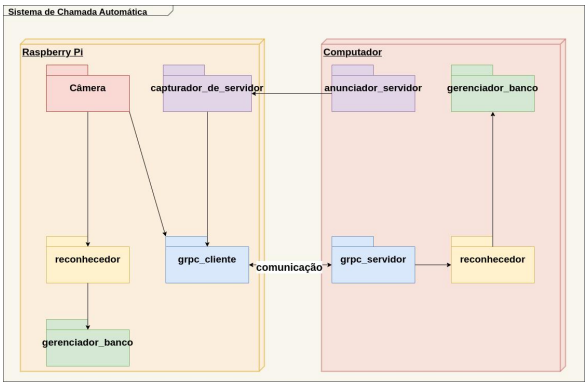
A arquitetura neste trabalho se dividiu em quatro etapas essenciais como pode ser visualizado na Figura 3a: Captura das imagens, reconhecimento no próprio *Raspberry Pi* e/ou transmissão de imagens para reconhecimento no dispositivo de borda. Uma câmera ficou responsável pela aquisição das imagens e as enviar à placa *Raspberry Pi* que, poderá realizar o reconhecimento e frequência na segunda etapa ou envia para um computador (terceira etapa) que realizará o reconhecimento das faces, sendo esta, a última etapa.

O sistema pode funcionar utilizando apenas um dispositivo realizando a frequência, mas pode estar em um ambiente de borda fazendo o *offloading* dos dados para serem processados no dispositivo de borda. Na Figura 3b é apresentado um diagrama de módulos deste sistema. O `anunciador_servidor` envia os dados de endereço do servidor através de um serviço *multicast* e o `capturador_de_servidor` recebe esses dados e seta o sistema com o endereço do servidor. A câmera envia a imagem ao reconhecedor ou ao `grpc_cliente` que envia ao computador. O `reconhecedor`, tando do *Raspberry Pi* quanto do Computador, podem realizar o reconhecimento facial e enviar ao banco de dados.

Figura 3 – Arquitetura geral do sistema e diagrama dos módulos do sistema.



(a) Arquitetura do sistema mostrando uma visão macro.



(b) Módulos que compõem o sistema.

4.1.1 Captura das imagens

Para realizar a captura das imagens, o sistema oferece duas opções aos usuários: o uso de um módulo de câmera específico para projetos com *Raspberry Pi* ou uma câmera USB conectada ao dispositivo. Essas alternativas possibilitam a aplicação do sistema em dispositivos que não possuam um conector próprio para câmeras. Para assegurar o funcionamento adequado de ambas as opções, a etapa de captura de imagens utiliza a biblioteca *PiCamera2*[†] e a biblioteca *OpenCV*, muito empregada em projetos de processamento de imagens (BRAHMBHATT, 2013).

O sistema emprega, independente da ferramenta que captura a imagem, a técnica *Haarcascade* da biblioteca *OpenCV* para detectar a presença de um rosto em frente à câmera, informando ao usuário para aguardar até que a posição seja ideal para a captura da fotografia. Esse procedimento é executado de maneira simples através de um *loop*: após a detecção de um rosto, o sistema itera mais duas vezes analisando assim, mais duas imagens antes de capturar a imagem que será utilizada. Essa abordagem visa evitar que a imagem seja registrada imediatamente ao detectar o rosto, o que poderia resultar em uma fotografia borrada devido ao movimento do usuário ao se posicionar em frente à câmera.

4.1.2 Comunicação dos nós

Devido às limitações de processamento do *Raspberry Pi*, optou-se por implementar também um mecanismo de *offloading* computacional. Quando o reconhecimento facial for custoso ou, por alguma razão o dispositivo não for eficiente ou rápido o suficiente, as imagens capturadas são enviadas para um dispositivo de borda com maior capacidade computacional. Esse dispositivo de borda realiza o reconhecimento de maneira mais eficiente e retorna os resultados ao *Raspberry Pi*.

O envio das imagens foi realizado utilizando a tecnologia *Google Remote Procedure Call* (gRPC), escolhida pela sua alta eficiência na transferência de dados em rede, sem comprometer a segurança (MATOS *et al.*, 2021). O gRPC permite uma comunicação rápida e confiável entre sistemas distribuídos, o que é crucial para o desempenho de aplicações que demandam baixa latência e alta escalabilidade (INDRASIRI; KURUPPU, 2020). Além disso, essa tecnologia oferece suporte a múltiplas linguagens de programação e utiliza a compactação de dados, o que otimiza a transmissão e reduz o consumo de largura de banda. Nesse contexto,

[†] Disponível em https://pypi.org/project/picamera2/

foi desenvolvido um serviço gRPC específico para receber os dados dos rostos recortados e transmiti-los ao dispositivo de borda, além de responder ao cliente (*Raspberry Pi*) também utilizando o mesmo método de comunicação, garantindo uma comunicação segura e eficiente ao longo do processo.

O protocolo de comunicação proposto neste trabalho define duas estruturas de mensagem: `ClienteParaServidor` e `ServidorParaCliente`, que estabelecem o *framework* para a interação entre o cliente e o servidor. A mensagem `ClienteParaServidor`, originada no cliente, incorpora os seguintes elementos: tipo de requisição (`type`, representado por um inteiro de 32 bits), identificadores de turma, disciplina e aluno (todos em formato *string*), um *timestamp* (`time`, preferencialmente um inteiro de 64 bits para maior granularidade temporal), uma imagem em formato binário (`image`), suas dimensões (`shape`, em *string*) e o nome do aluno.

A mensagem `ServidorParaCliente`, gerada como resposta do servidor, replica os campos correspondentes da solicitação, acrescentando o número de faltas do aluno (`num_faltas`, um inteiro) e um repositório de dados binários (`repositorio`) que pode ser utilizado para transacionar pacotes de imagens para compor o banco de dados do cliente. O serviço denominado `EnvioDeMensagens` implementa a funcionalidade de transmissão, permitindo ao cliente enviar uma mensagem `ClienteParaServidor` e receber a resposta `ServidorParaCliente` através do RPC `EnviarMensagem`.

4.1.3 Descoberta de Servidor

O sistema utiliza um servidor *socket* que, por meio de um serviço *multicast*, envia pacotes na rede contendo o endereço IP do servidor gRPC, responsável pela comunicação entre cliente e servidor para a troca de imagens. O cliente monitora a rede em busca de pacotes com endereços de servidores e, caso não receba nenhuma mensagem, entra em repouso por cinco segundos para evitar a sobrecarga da rede. Após receber os dados do corpo da mensagem, os valores iram preencher a configuração de IP do servidor gRPC durante toda a execução do sistema.

4.1.4 Reconhecimento Facial através da biblioteca DeepFace

O reconhecimento facial pode ser realizado tanto no dispositivo de borda quanto no próprio *Raspberry Pi* de forma semelhante. Para a detecção e reconhecimento facial, foi utilizado *DeepFace*[*], uma biblioteca de código aberto disponível no repositório Pypi[†]. Essa biblioteca oferece suporte a múltiplos modelos de reconhecimento facial e extração de características, como *VGG-Face*, *Facenet*, *OpenFace*, *DeepFace*, entre outros. Bem como variados modelos de detecção de rosto como *You Only Look Once* na versão 8 (YOLOv8), *Dlib*, *Multi-task Cascaded Convolutional Networks* (MTCNN), etc. A flexibilidade da biblioteca permite que diferentes modelos sejam testados para avaliar seu desempenho em termos de precisão, velocidade e eficiência no ambiente específico de hardware e software utilizado.

A biblioteca anteriormente chamada *LightFace* e depois *DeepFace* é um *framework* de análise de faces ainda em desenvolvimento, mas que conta com análises que vão desde o reconhecimento facial até a análise de expressões. No entanto, neste trabalho, será abordado apenas aspectos da biblioteca que envolvem o reconhecimento facial. *Deepface* atua dentro do *pipeline* mais utilizado de reconhecimento facial que segue as etapas de: detecção de faces;

[*] Deepface - https://pypi.org/project/deepface/
[†] PyPI - http://pypi.org

Alinhamento; extração de características faciais e comparação dos vetores (TAIGMAN *et al.*, 2014).

A inclusão de algoritmos de alinhamento facial na etapa de pré-processamento das imagens de entrada eleva a acurácia do modelo final de reconhecimento facial de 98,87% para 99,63%. Com isso, o autor do projeto adiciona a possibilidade de usar várias opções de modelo de detecção de face e alinhamento das faces (WANG; DENG, 2021 apud SERENGIL; OZPINAR, 2020). Alguns dos modelos desta etapa ainda são capazes de oferecer a possibilidade de "frontalizar" os rostos, voltado para situações em que uma face vista de forma lateral é transformada de forma que se pareça estar de frente para a câmera. Dentre os modelos suportados pela biblioteca na versão 0.0.92, estão: *opencv, ssd, dlib, mtcnn, fastmtcnn, retinaface, mediapipe, yolov8, yunet, centerface*. Após as etapas citadas anteriormente, será realizada a extração de características das imagens repassadas como parâmetros. Esta etapa é responsável por gerar o vetor de características de cada imagem para posterior comparação e cálculo da distância entre as faces. Para tal atividade, são utilizados alguns dos principais modelos de extração de características que despontam como os mais modernos disponíveis: *Facenet512, Human-beings, Facenet, Dlib, VGG-Face, ArcFace, GhostFaceNet, SFace, OpenFace, DeepFace, DeepID.*

A biblioteca *DeepFace* disponibiliza uma função pré-definida denominada "*find*", que é empregada para executar a tarefa de reconhecimento com base em um banco de imagens. Esta função busca identificar a imagem de entrada no diretório do banco de dados, retornando como resultado uma lista de *DataFrames* gerados pelo pandas. Esses *DataFrames* são compostos por 10 colunas com a posição do rosto na imagem, uma chave *hash*, a distância entre as imagens e a imagem mais próxima da imagem analisada. Simultaneamente, os *embeddings* faciais presentes no banco de dados são armazenados em um arquivo *pickle*[‡]. A utilização de uma arquivo serializado com as características das imagens evita a necessidade de calcular o vetor de todas as imagens a cada vez que o método é invocado, reduzindo o tempo de resposta e aliviando a carga do processador ou placa de vídeo. O tamanho do arquivo pkl será proporcional à quantidade de rostos identificados na imagem de entrada. Outro dado sobre esse método, é que quando invocada, a função "*find*" faz o *download* dos modelos escolhidos nos parâmetros de chamada da função caso não já tenham sido utilizados, tanto o detector quanto o modelo de extração de características. Após o *download*, é realizado uma extração das características faciais dos rostos repassados como referência seguindo o pipeline de reconhecimento como mencionado na Seção 2. As imagens enviadas ao sistema também podem conter múltiplos rostos, o que pode ser utilizado para reconhecer grupo de pessoas também, visto que a saída dessa função pode ser um ou mais *DataFrames*.

4.1.5 Seleção de Modelos

Devido a grande quantidade de modelos disponíveis na biblioteca *Deepface*, foi necessário elencar aqueles modelos que melhor se destacavam no reconhecimento facial em termos de tempo e acurácia, além de funcionar corretamente nos dois dispositivos. Dessa forma, foi elaborado um experimento que visou extrair tais métricas da biblioteca testando todas as possibilidades.

O processo transcorreu utilizando um *dataset* com 98 classes contendo 12 rostos de cada classe, portanto mil cento e setenta e seis (1176) imagens no total. Das 12 imagens de cada classe, 4 foram utilizadas para teste e as outras 8 para treino. Foi criado dois vetores contendo

[‡] Arquivos pickle possuem extensão de arquivo .pkl e são usados em Python para serializar e desserializar objetos. A serialização converte um objeto em um formato armazenável ou transmissível, enquanto a desserialização faz o oposto, o que torna esses formatos mais fáceis de serem manipulados.

os parâmetros de detectores e modelos (e.g. *detector* = ['yolov8', 'mtcnn', 'dlib', 'opencv', ...] e *model* = ['VGG-Face', 'Facenet', 'Dlib', 'ArcFace', ...]) de forma que dois laços aninhados pudesse passar por todas as possíveis combinações de detector/modelo e testá-los com o banco de dados. Essa forma funciona pois os parâmetros de detector e modelo da função "*find*" são entradas no formato de *String*.

Cada detector e modelo passou por todas as imagens treinando e testando de forma que foram gerados cem arquivos *pickle*, dado que a biblioteca trabalha com 10 tipos de detector e 10 tipos de modelos de extração de características faciais. O experimento foi realizando utilizando o serviço de *Jupyter Notebook* hospedado que oferece acesso a recursos de computação chamado Google *Colaboratory*. No entanto, a configuração não contou com *Tensor Processing Units* (TPU)s ou *Graphic Processing Unit* (GPU)s ativadas, apenas os recursos básicos oferecidos pelo sistema. Não foi considerado no experimento o consumo de *Central Processing Unit* (CPU) e nem memória *Random Access Memory* (RAM). Os modelos com acurácia acima de 90%, ordenados a partir da acurácia média resultante, podem ser vistos na Tabela 2.

Tabela 2 – Pontuação das melhores combinações de detector e no que se refere a acurácia nos testes realizados

Detector	Modelo	Acurácia	Média (s)	Variância	Desvio Padrão *s*
yolov8	ArcFace	98%	1.98	1.2130	1.1014
mtcnn	ArcFace	98%	2.56	0.2243	0.4736
retinaface	ArcFace	96%	3.33	1.7798	1.3341
dlib	ArcFace	94%	2.21	0.3414	0.5843
yolov8	VGG-Face	94%	2.41	2.1592	1.4694
mtcnn	VGG-Face	94%	3.1	2.0673	1.4378
yolov8	Facenet	93%	1.89	0.0190	0.1379
mtcnn	Facenet	93%	2.51	0.1945	0.4410
retinaface	Facenet	92%	3.27	0.0892	0.2986
retinaface	VGG-Face	92%	3.84	0.2202	0.4693
dlib	Dlib	91%	2.07	0.3205	0.5661
ssd	Facenet	90%	1.86	1.9656	1.9657
ssd	ArcFace	90%	1.89	0.0147	0.1214

Fonte: O autor.

Os modelos escolhidos para serem utilizados no sistema foram: ['yolov8', 'ArcFace'], ['mtcnn', 'ArcFace'], ['yolov8', 'Facenet'] e ['mtcnn', 'Facenet']. Os critérios para a escolha dos detectores e modelos se deu tomando o tempo de reconhecimento seguido da acurácia. Por exemplo, apesar do par ['mtcnn', 'Facenet'] despontar na oitava posição em termos de acurácia, este par é quinto em termos de tempo médio de detecção. Dado que dentre os 10 todos estão com acurácia acima dos 90%, considerou-se que todos seriam ótima combinações para serem utilizadas e é nesse aspecto que entra a importância das outras métricas como a variância que coloca o par ['mtcnn', 'Facenet'] acima do ['yolov8', 'VGG-Face'], por exemplo. A combinação ['ssd', 'ArcFace'] foi escolhida posteriormente por se atender aos requisitos exigidos para o sistema, ou seja, ter boa pontuação no reconhecimento além de boa estabilidade temporal. Contudo, este par detector/modelo não foi testado nesta outra etapa e não aparece nos gráficos deste pré análise. Apesar do par ['ssd', 'ArcFace'] não ser o mais forte em termos de reconhecimento, 90% de acurácia é um bom percentual. Além disso o mesmo demonstrou bastante estabilidade em termos temporais, o que é interessante se posto que será utilizado por

pessoas.

Após definir pelo menos 4 pares detector/modelo que melhor performaram no sistema, decidiu-se avaliar estes mesmos modelos em uma ambiente de poucos dados como se pretende este trabalho, haja vista a proposta de construir um banco de dados com apenas cinco imagens enviadas pelos próprios alunos. Desta forma, foram realizados novos experimentos com um *dataset* de 98 classes, sendo sete imagens para cada uma, dividido em cinco imagens para treino e duas imagens para teste. A Tabela 3 mostra os resultados obtidos neste experimento. O modelo ArcFace é muito eficiente com qualquer um dos detectores analisados, bem a frente do Facenet que está, no mínimo, 4 pontos percentuais abaixo. Contudo, em termos temporais, o yolov8 torna os dois modelos bem estáveis com ligeiro favorecimento do ArcFace, sobretudo no desvio padrão. Em todos os casos, é possível perceber que os pares detector/modelo conseguem ter boa acurácia mesmo com apenas 5 imagens no banco.

Tabela 3 – Avaliação dos pares detector/modelo com apenas 5 imagens de referência no banco

Detector	Modelo	Acurácia (%)	Tempo Médio (s)	Variância (s)	Desvio Padrão (s)
mtcnn	ArcFace	97%	2.54	0.2497	0.4997
yolov8	ArcFace	96%	1.81	0.0181	0.1348
yolov8	Facenet	92%	1.85	0.0977	0.3126
mtcnn	Facenet	90%	2.62	0.4271	0.6535

Fonte: O autor.

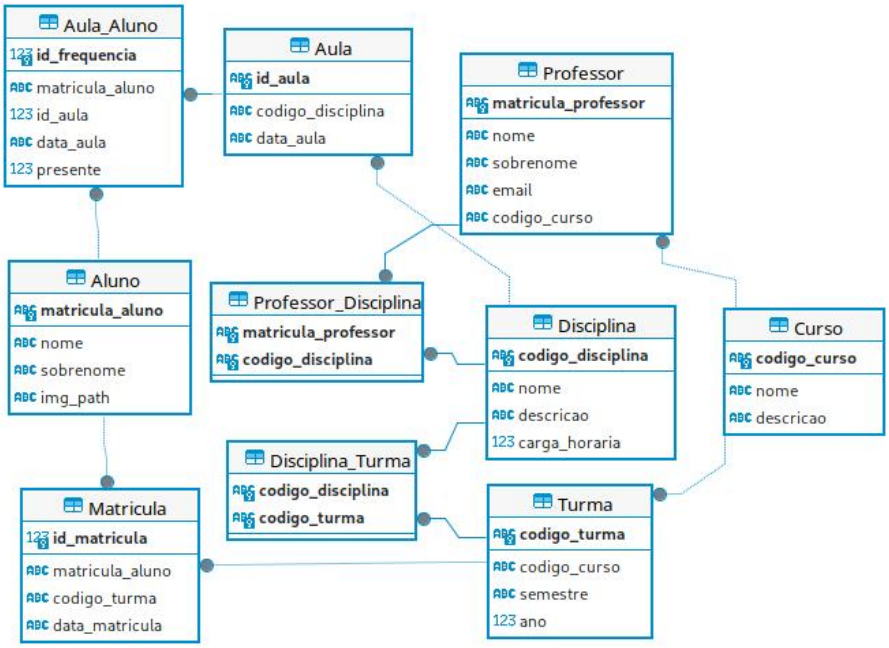
4.1.6 Banco de Dados

Assim como foi utilizado por Khan *et al.* (2020) em seu estudo, o banco de dados empregado para armazenar as informações relevantes ao experimento foi desenvolvido em *SQLite*. Este banco de dados, além de ser compatível com SQL, destaca-se por sua facilidade de uso e por se adequar às limitações de dispositivos como o *Raspberry Pi*, que possuem espaço de disco reduzido. O *SQLite* utiliza um arquivo que ocupa pouco espaço de memória e é acessível ao usuário para operações como backup facilitado (DIANA *et al.*, 2020). O banco de dados foi projetado para manter a integridade referencial por meio de chaves estrangeiras, garantindo que os dados sejam organizados de maneira consistente e relacionável.

A Figura 4 mostra o diagrama obtido no projeto deste banco de dados. A estrutura proposta visa facilitar a gestão de um ambiente acadêmico de forma a permiti o rastreamento de cursos, disciplinas, turmas, professores, alunos e suas respectivas matrículas e presenças. O banco de dados foi estruturado com as seguintes tabelas:

- **Curso**: Esta tabela armazena os detalhes dos cursos oferecidos. Cada curso possui um código único (codigo-curso), que é a chave primária, além do nome e uma descrição.
- **Professor**: Guarda as informações dos professores. Cada professor é identificado por uma matrícula única (matricula-professor), que é a chave primária. Além disso, a tabela contém o nome, sobrenome e *e-mail* do professor, sendo que o *e-mail* deve ser único. Cada professor também está associado a um curso, identificado pelo codigo-curso, que referencia a tabela Curso.
- **Disciplina**: Aqui são armazenadas as informações das disciplinas. Cada disciplina tem um código único (codigo-disciplina), que é a chave primária, além do nome, descrição e carga horária.
- **Turma**: Armazena os dados das turmas criadas para os cursos. Cada turma é identificada por um código único (codigo-turma), que é a chave primária. A turma está associada a um

Figura 4 – Diagrama entidade relacionamento do banco de dados do sistema



Fonte: O autor.

curso, identificado pelo codigo-curso, e contém informações sobre o semestre e ano em que a turma é oferecida.

- **Aluno**: Nesta tabela são armazenados os dados dos alunos. Cada aluno possui uma matrícula única (matricula-aluno), que é a chave primária, além do nome, sobrenome e o caminho da imagem (img-path) do aluno.
- **Aluno-Turma**: Registra as matrículas dos alunos nas turmas. Cada matrícula tem um identificador único (id-matricula), que é a chave primária gerada automaticamente. A tabela também armazena a matrícula do aluno (matricula-aluno), que referencia a tabela Aluno, e o código da turma (codigo-turma), que referencia a tabela Turma, além da data da matrícula.
- **Aula**: Esta tabela registra as aulas ministradas em disciplinas específicas. Cada aula é identificada por um ID único (id-aula), que é a chave primária, e está associada a uma disciplina, identificada pelo codigo-disciplina, que referencia a tabela Disciplina. A tabela também armazena a data da aula.
- **Aula-Aluno**: Nesta tabela são registradas as presenças dos alunos em aulas específicas. Cada registro de frequência tem um ID único (id-frequencia), que é a chave primária gerada automaticamente. A tabela armazena a matrícula do aluno (matricula-aluno), que referencia a tabela Aluno, e o ID da aula (id-aula), que referencia a tabela Aula, além da hora em que a presença foi registrada e um campo numérico que indica se o aluno estava presente.
- **Professor-Disciplina**: Esta tabela cria uma relação entre professores e as disciplinas que eles ensinam. Ela usa a combinação da matrícula do professor (matricula-professor) e do código da disciplina (codigo-disciplina) como chave primária, permitindo uma relação muitos-para-muitos entre as tabelas Professor e Disciplina.
- **Disciplina-Turma**: Esta tabela relaciona as disciplinas com as turmas em que são oferecidas. Assim como a tabela anterior, ela usa a combinação do código da disciplina (codigo-disciplina) e do código da turma (codigo-turma) como chave primária, criando

uma relação muitos-para-muitos entre as tabelas Disciplina e Turma.

4.1.7 Terminal de Usuário

O sistema conta com um terminal para o usuário com um *display* LCD1602 que conta com duas linhas de 16 segmentos para conteúdo alfa numérico. O mesmo foi conectado ao *Raspberry Pi* através das portas GPIO utilizando uma biblioteca chamada lcd1602gpio 0.3.1[§]. Apesar estar na sua última versão e, por se tratar de uma tarefa simples, essa biblioteca não apresenta nenhum problema para ser utilizada e possui um número muito pequeno de dependências. Na montagem do *display* no Raspberry Pi, utilizou-se um potenciômetro *trimpot* para ajuste de contraste do mesmo. A luz de fundo foi mantida em uma intensidade padrão suficiente para visualização.

O *display* tem a função de mediar a interação entre o sistema e o usuário, fornecendo alertas sobre o status do sistema, o endereço IP do dispositivo, o resultado do reconhecimento, entre outros. A linguagem utilizada nas mensagens exibidas no *display* foi simples e acessível, visando facilitar a compreensão por parte dos usuários.

4.2 Ferramentas e Tecnologias Utilizadas

Para o experimento do sistema proposto, foi utilizado um *Raspberry Pi* como dispositivo principal, responsável pela captura de imagens e reconhecimento facial em tempo real. A utilização do dispositivo foi motivada, sobretudo, por ser bastante conhecido na área de IoT e por fazer parte do acervo do Laboratório de Engenharia de Software e Sistemas (EngineLab) da Universidade Federal do Ceará no Campus de Crateús no estado do Ceará. Um esquema pode ser visto na Figura 5, que ilustra a disposição e ligação dos componentes no *Raspberry Pi*. O protótipo de ligação dos componentes foi desenvolvido na ferramenta Fritzing[¶]. Esta ferramenta possui diversos componentes disponíveis após a instalação, mas pode receber novos pacotes de componentes, mas durante a prototipação não foi encontrado um cabo *flat* que refletisse o que foi utilizado no projeto. Portanto, a representação da ligação entre a câmera e a placa foi feita através de um fio comum, colorido com vermelho e branco.

Para acomodar os componentes de hardware, foi desenvolvido um gabinete utilizando uma impressora 3D. O *design* do gabinete foi pensado para permitir a circulação de ar, favorecendo a refrigeração do *Raspberry Pi*, além de proporcionar a visualização do *display*, controle de contraste e posicionamento da câmera. Na parte interna do fundo do gabinete, foram criadas linhas em alto relevo com espaçamento de três milímetros, para a soldagem de porcas que fixariam a placa do *Raspberry Pi*. As aberturas foram projetadas propositalmente maiores que o necessário, facilitando a adição de novos componentes, caso necessário. A Figura 6 apresenta o modelo 3D do gabinete, desenvolvido no *TinkerCad*, uma ferramenta online da *AutoCad*.

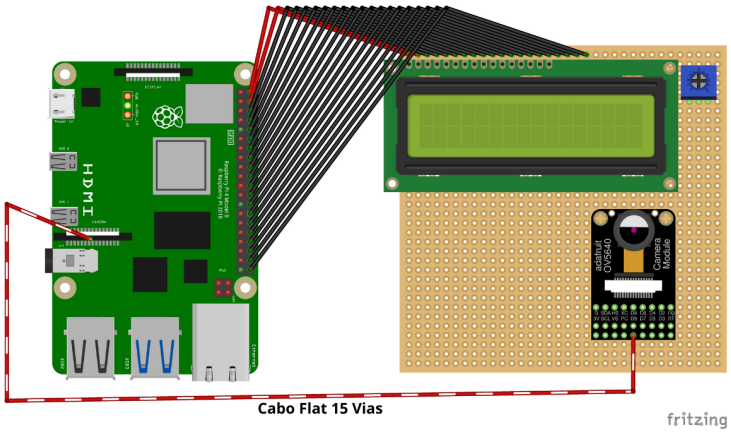
O gabinete foi impresso no laboratório de física do Campus e demorou catorze horas para ficar pronto. Após a impressão, foi afixados porcas nas grades do fundo para acomodar o *Raspberry Pi* e o resultado pode ser visto na Figura 7. Passagens foram deixadas para conexões com a placa. A tampa frontal do gabinete foi propositalmente feita com uma abertura maior possibilitando a instalação de novos componentes no projeto.

---

§ Disponível em https://pypi.org/project/lcd1602gpio/

¶ Disponível em https://fritzing.org/

Figura 5 – Esquema físico de ligação dos componentes desenvolvido com a ferramenta Fritzing



Fonte: O autor.

Figura 6 – Protótipo do gabinete desenvolvido no TinkerCad para acomodar os componentes de hardware. Posteriormente impresso em 3D

Fonte: O autor.

4.2.1 Preparação do sistema

Para o experimento, foi solicitado aos alunos do laboratório que compartilhassem imagens pessoais que pudessem ser utilizadas no experimento e que fossem registros de oca-

Figura 7 – Fixação do *Raspberry Pi* no gabinete impresso em 3D.

Fonte: O autor.

siões, ângulos, lugares e condições diferentes. Aos que utilizam óculos, foi recomendado que enviassem imagens também com o adereço para enriquecer a base de dados. As mesmas foram compartilhadas pelo WhatsApp e possuem resoluções que variam de `340x470` até `3113x4160`. A maioria das imagens está no formato *Joint Photographic Experts Group* (JPG), mas algumas estavam no formato *Portable Network Graphic* (PNG) também. Coletadas as imagens, para cada aluno foi criada uma base de dados contendo cinco imagens distintas. As imagens foram salvas em diretórios nomeados com o código da turma de forma a facilitar a busca por dados de uma aluno a partir da turma. Todas as imagens foram nomeadas com a matrícula do referido aluno sufixadas por X, onde X é um número qualquer que diferencia o nome das imagens. Na sequência, os nomes dos alunos e suas respectivas matrículas foram todos cadastrados no banco de dados para que esse dados fossem exibidos no *display* após o reconhecimento, além de armazenar a frequência dos alunos.

4.2.2 Dispositivo principal

O *Raspberry Pi* 4 Modelo B[‖], com 4GB de RAM, foi o modelo utilizado no projeto. Trata-se de um microcomputador versátil, ideal para diversas aplicações, bem como todos os outros modelos *Raspberry Pi*. São utilizados em diversos projetos desde simples microcontroladores até como computadores de pequeno porte. O modelo utilizado possui um processador *quad-core Cortex-A72* de 1.8GHz e conectividade *Wi-Fi*, *Bluetooth* e *Ethernet Gigabit*. A placa suporta até dois monitores 4K e conta com algumas portas USB para expansão e conectividade. Para seu funcionamento, exige uma fonte de alimentação estável de no mínimo 3 amperes e,

[‖] Mais informações em https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

ainda assim, pode apresentar limitações em tarefas que demandam alto poder de processamento gráfico.

4.2.3 Alimentação do Raspberry Pi

A alimentação do *Raspberry Pi* foi realizada utilizando um carregador de celular modelo LE-1 Turbo de 20W, capaz de fornecer até 4 amperes de corrente a 5 volts, que é a tensão recomendada para o *Raspberry Pi*. O cabo utilizado para conectar a fonte ao dispositivo foi o mesmo fornecido pelo fabricante do carregador, o qual afirma suportar 4 amperes a 5 volts; no entanto, não foi possível avaliar a capacidade real do cabo, tampouco a capacidade do carregador.

4.2.4 Cartão SD e Sistema Operacional do Raspberry Pi

Foi utilizado o sistema operacional *Raspbian*[**] que é uma distribuição Linux desenvolvida para operar nos *Raspberry Pi* (GAY, 2018). A mesma deriva de outra distribuição chamada *Debian* e é considerada o sistema operacional padrão desses dispositivos. O sistema operacional foi instalado dentro de um cartão SD Classe 4 da SanDisk com capacidade igual a 32GB, que já é bastante para um sistema que ocupa pouco espaço, porém é pouco para se trabalhar com imagens caso se deseje armazenar todas as imagens de uma turma além dos arquivos *pickle* e dos modelos que são baixados no sistema para extração de características.

4.2.5 Câmera

A câmera utilizada foi o modelo OV5647, que possui foco fixo e foi especialmente projetada para o *Raspberry Pi*. Este modelo é capaz de capturar fotos com resolução de até 5 megapixels e gravar vídeos com taxas de até 90 *Frames per Second* (FPS) em resolução *Video Graphics Array* (VGA). O módulo de câmera utiliza uma interface de conexão que permite a transferência rápida de dados para o processador BCM2835 por meio de um cabo flat de 15 vias (EJIMA; HATANO, 2018).

4.2.6 Monitor do usuário

Para exibir informações ao usuário, foi utilizado um *display* LCD de 16 segmentos e duas linhas com luz de fundo azul, identificado pelo código LCD1602. Este *display* pode operar com 4 ou 8 bits de comunicação, sendo o modo de 4 bits o que utilizado neste trabalho. A interface de comunicação entre o *Raspberry Pi* e o *display* foi estabelecida através das portas GPIO do microcomputador, na qual foi utilizada um cabo de 20 vias ligando o lado par dos pinos GPIO a placa do *display*.

4.2.7 Dispositivo de borda

O dispositivo de borda utilizado neste experimento é um computador *Notebook Avell A52 Hyb*, equipado com um processador Intel de décima-segunda geração modelo Core i5-12450H de 8 núcleos e 12 *threads*. A configuração gráfica inclui uma placa de vídeo integrada Intel UHD Graphics de décima-segunda geração. O sistema operacional instalado é o Linux Kubuntu 24.04 com *Kernel 6.8.0-41-generic (64-bit)*, baseado no Ubuntu 24.04. O dispositivo possui 24GB de memória RAM DDR4 3200 MHz e um *Non-Volatile Memory Express* (NVMe)

[**] http://www.raspbian.org/

A-Data de 512GB, com mais de 100GB de capacidade disponível. Devido a instabilidade da rede da universidade, foi criado uma rede com um celular *Samsung* A21s utilizando a conexão 2.4GHz por não conter um roteador disponível no momento do experimento.

4.2.8 Bibliotecas utilizadas no lado servidor

Todo o sistema foi desenvolvido em *Python* rodando a versão 3.12 dessa linguagem. A biblioteca responsável pelo reconhecimento estava em sua versão 0.0.92, sendo que a mesma já se encontra na versão 0.0.93. As bibliotecas de recorte de rosto, ou seja, que ficam na parte inicial da extração de características, *mtcnn* e *yolov8* estavam nas versões 0.1.1 e 8.2.60, respectivamente. Contudo, apenas a 'yolov8' foi testada no experimento com usuário, devido a necessidade de reconfiguração do sistema para utilizar outro modelo. Como o par detector/modelo escolhido foi *['ssd', 'ArcFace']*, o sistema baixou o arquivo *Hierarchical Data Format version 5* (HDF5) de 131MB com os pesos resultantes do treinamento do modelo ArcFace. A escolha deste par detector/modelo se deve ao fato de o mesmo ter ocupado o primeiro lugar que pode ser visto na Tabela 2. Outras bibliotecas que foram necessárias foram: grpcio na versão 1.65.1, grpcio-tools na versão 1.62.2 e Flask na 3.0.3 para a parte de comunicação. A biblioteca numpy, utilizada para a parte de conversão de dados estava na versão 1.26.4. A biblioteca pandas, na 2.2.2, serviu para o gerenciamento dos dados produzidos durante o reconhecimento.

4.2.9 Bibliotecas utilizadas no Raspberry Pi

A versão da biblioteca responsável pelo reconhecimento facial, *deepface*, utilizada no experimento foi a 0.0.92. Esta biblioteca já tem versões mais recentes, porém, a versão 0.0.93 saiu depois da realização dos experimentos. Para a parte de interface web e comunicação, foi utilizada a biblioteca *Flask* na versão 2.2.2, enquanto *grpcio* e *grpcio-tools* estavam ambas na versão 1.65.0. O sistema também fez uso de *tensorflow* e *keras* nas versões 2.17.0 e 3.4.1, respectivamente, responsáveis pela execução do modelo de reconhecimento facial. Além disso, foi necessário o download de arquivos de pesos em formato *HDF5* com média de 135MB, resultado do treinamento do modelo utilizado, para o correto funcionamento da arquitetura de redes neurais. Outras bibliotecas que desempenharam papel fundamental no projeto incluem *numpy* (1.24.2) e *pandas* (2.2.2), encarregadas da manipulação e conversão dos dados gerados durante o processo de reconhecimento. Para visualizações gráficas e plotagem de resultados, utilizou-se *matplotlib* na versão 3.9.1.

Para controle de hardware, as bibliotecas *RPi.GPIO* (0.7.1a4) e *lgpio* (0.2.2.0) foram essenciais no gerenciamento das interações com os pinos GPIO do Raspberry Pi. Além disso, a captura de imagens foi realizada por meio da biblioteca *picamera2* (0.3.19), enquanto *lcd1602gpio* (0.3.1) permitiu a integração de um display LCD para visualização das saídas. Por fim, o sistema contou ainda com *torch* (2.4.0) e *torchvision* (0.19.0), usados para suporte ao treinamento e otimização de redes neurais adicionais, bem como com *scipy* (1.14.0) e *opencv-python* (4.10.0.84), que auxiliaram em operações matemáticas e de processamento de imagem, respectivamente. A comunicação entre diferentes componentes do sistema foi realizada utilizando *protobuf* (4.25.4) como protocolo de serialização de dados.

5 RESULTADOS E EXPERIMENTOS

O experimento foi conduzido em duas etapas distintas, cada uma focada em diferentes aspectos do desempenho do sistema:

1. **Etapa de Implementação em Ambiente Real**: O sistema foi testado em um laboratório da Universidade Federal do Ceará no Campus de Crateús com participantes reais e com a simulação de seu funcionamento real como uma prova de conceito. O dispositivo foi fixado em um quadro branco a uma altura de 1.5m de forma que o ângulo da câmera permitia boa visibilidade dos rostos dos alunos desde que os mesmos respeitassem uma distância superior a um metro, cuja qual foi informada aos participantes no início do experimento. O ambiente estava iluminado por luzes artificiais e luz natural. Os alunos se posicionavam em frente ao dispositivo que exibia a mensagem para que os mesmo ficassem parados aguardando a captura da imagem. Em, aproximadamente, 3 segundos o sistema já capturava a imagem para realizar o reconhecimento. O sistema foi testado tanto para reconhecimento local, no próprio *Raspberry Pi*, quanto em um dispositivo de borda que se tratava de um computador com maior poder computacional em relação ao *Raspberry Pi*. Este último, realizava o reconhecimento facial em tempo real o que incluia a captura da imagem, recorte de face, extração de características e comparação com as métricas já contidas no banco. No caso do reconhecimento utilizando um dispositivo de borda, o *Raspberry Pi* apenas realizava a captura da imagem e informava aos usuários os resultado da captura. As imagens, independentemente de serem processadas pelo *Raspberry Pi* ou pelo dispositivo de borda, foram todas armazenadas para a Etapa 2 dos experimentos.
2. **Etapa de Estresse Computacional e Análise de Desempenho**: Utilizando as imagens coletadas na primeira etapa, foram realizados testes exaustivos para avaliar o desempenho do sistema. O mesmo par detector/modelo utilizado no *Raspberry Pi* foi utilizado nesta etapa para que se pudesse observar os dados e comparar os resultados. Este teste foi feito tanto no próprio *Raspberry Pi* quanto no dispositivo de borda, porém, escalando em quantidade de clientes realizando as requisições. Durante os testes, o *Raspberry Pi* e o dispositivo de borda processaram as imagens, e os dados de desempenho foram registrados.

Na primeira etapa, execução em um ambiente real, foi necessário preparar o sistema para fazer o reconhecimento dos alunos que iriam participar do experimento. Para preparar o ambiente foi seguido os seguintes passos:

- **Cadastro de Alunos** - Inicialmente, foi realizado o cadastro dos alunos participantes do experimento com a coleta de informações como: Nome, Matrícula, etc. Além de solicitado o envio das imagens.
- **Configuração do Banco de Dados** - Após o cadastro dos alunos, procedeu-se à configuração do banco de dados. Esta etapa envolveu a preparação de um ambiente para o armazenamento das informações dos alunos durante os experimentos.
- **Preparação do ambiente** - Para a realização do experimento foi realizada a seleção de uma sala do laboratório que poderia ser isolada sem ônus dos demais projetos. Para isso, foi escolhido a sala de reuniões. O dispositivo contendo a placa *Raspberry Pi* e os componentes foi fixado em um quadro branco próximo a uma tomada utilizando uma fita dupla face posicionado a uma altura entre 1,4 e 1,5 metros.
- **Informes para o Experimentos** - Os usuários foram instruídos a observar as mensagens exibidas do *display*. Além disso, foram informados que caso diante do dispositivo e não fosse gerada a imagem, os mesmo deveriam se afastar até a exibição da mensagem: `Estou te reconhecendo.` A distância considerada para uma boa captura foi informada aos participantes e avizinhava um metro ou pouco mais que isso. Não foi marcado um ponto fixo no chão pois a distância dita ideal poderia variar de acordo com a altura dos participantes, sobretudo para os casos de pessoas de mais alta estatura.

5.1 Experimento em Ambiente Real

O experimento foi iniciado usando o servidor às 13:00 horas da tarde do dia 27 de agosto de 2024 no laboratório EngineLab. Na oportunidade, foi testado tanto com o dispositivo *Raspberry Pi* realizando requisições ao dispositivo de borda, quanto os testes nele prórpio. Os alunos foram informados sobre a fase em que se encontrava o experimento, se usando o dispositivo de borda ou se estava usando apenas o dispositivo local, o *Raspberry Pi*. Na oportunidade foi produzido um vídeo demonstrando a utilização do sistema[††].

O uso do sistema transcorre com o usuário se aproximando do dispositivo e interagindo com o *display* do mesmo. Não havendo usuários diante do dispositivo o mesmo exibe uma mensagem: `Sem rostos a reconhecer`. Quando o usuário se aproxima, o mesmo passa a exibir uma mensagem que pede ao usuário para se manter parado e aguardar a captura da melhor imagem. Quando a imagem é captura o sistema exibe uma mensagem própria para a situação. A Figura 8 mostra como essa informação era exibida aos usuários.

Figura 8 – *Display* exibindo uma mensagem aos usuários. Mensagem exibe a matricula do aluno e o tempo gasto para o reconhecimento

Fonte: O autor.

As imagens capturadas foram submetidas a uma análise de qualidade de captura e reconhecimento humano de forma que se pudesse garantir que a mesma foi classificada de forma correta. A análise das imagens revelou que o sistema apresentou uma média de tempo de 4,5 segundos para a análise no próprio *Raspberry Pi* e de 1,97 segundos utilizando dispositivo de borda. O sistema reconheceu 12 rostos dos 14 participantes do experimento. O mesmo aconteceu na utilização do dispositivo de borda, os acertos ficaram em 85.71% e os erros em 14,29%. A Figura 9 apresenta uma das imagens que foi recebida pelo dispositivo de borda e que foi corretamente reconhecida pelo sistema. Na imagem é possível perceber a qualidade que chega ao servidor, com boa iluminação, resolução, porém com outros rostos que podem ser enquadrados na hora da captura.

Tanto no Raspberry Pi como no dispositivo de borda, o par detector/modelo utilizado foi o ['*ssd*', '*ArcFace*'], devido à melhor compatibilidade observada em comparação com outros

[††] O vídeo demonstrativo está disponível em: https://youtu.be/Hyrcst-8hx4

Figura 9 – Imagem capturada pelo sistema e enviada ao servidor.

Fonte: O autor.

pares de modelos testados nos dois dispositivos. Antes da realização da prova de conceito com o teste com pessoas em um ambiente real, o sistema foi submetido a pré-testes de execução para sanar quaisquer problemas que pudessem acontecer durante o experimento com os alunos. Durante os pré-testes utilizando o detector "*yolov8*" foi identificado um erro com a mensagem "Instrução Ilegal", que, apesar de ser possivelmente solucionável, não havia tempo hábil para correção durante o experimento. Ao utilizar o "*mtcnn*", a biblioteca iniciava um processo de treinamento a cada tentativa de reconhecimento, gerando um custo computacional e temporal desnecessário, já que o arquivo de treinamento "*pkl*" já havia sido gerado anteriormente.

5.1.1 Observações e impressões do usuários

Os participantes do experimento foram alunos do próprio laboratório. Um aspecto observado pelos participantes foi o reconhecimento de rostos de alunos que não estavam se posicionando intencionalmente para a captura de frequência conforme visto na Figura 9. Alunos que passavam em frente à câmera foram fotografados em movimento ou em poses inadequadas para um bom reconhecimento justamente por não esterem tentando realizar a frequência. Isso ocorreu devido ao funcionamento do sistema de captura de imagens apontado na subseção 4.1.1. Nas três iterações do laço haviam rostos, mas na iteração que geraria a imagem não realmente tinha rostos a serem registrados. Em um dos casos, apesar do rosto ter ficado ligeiramente borrado devido ao movimento durante a captura, o sistema conseguiu realizar o reconhecimento do aluno de forma correta.

Durante essa etapa, o aluno A conseguiu visualizar facilmente as informações exibidas no *display*, inclusive o tempo de processamento para o reconhecimento facial. Em contrapartida, o aluno B relatou dificuldades no que se refere a capacidade da câmera, pois, ao se posicionar em frente ao dispositivo, o sistema apresentava repetidamente a mensagem "Sem rostos a registrar". Essa mensagem é exibida quando não há rostos na frente da câmera. O problema foi identificado como resultado do uso de um boné pelo aluno B, cuja aba cobria parcialmente os olhos, impedindo que a biblioteca *OpenCV* reconhecesse o rosto. Após remover o boné, o sistema foi capaz de realizar o reconhecimento corretamente. A aluna C, que utilizava

óculos, teve seu rosto identificado corretamente pelo sistema, apesar do adereço. No entanto, a aluna D, que também era usuária de óculos de grau, relatou um erro de reconhecimento, acreditando ter sido identificada como outro aluno. Ao revisar os dados, verificou-se que o reconhecimento havia sido efetuado corretamente. A confusão ocorreu, possivelmente, devido ao tamanho reduzido do *display*, que dificultou a leitura correta das informações pela aluna, destacando um aspecto a ser melhorado em termos de acessibilidade.

A aluna D, que havia reportado problemas na etapa de teste no dispositivo de borda, novamente teve dificuldades quando o teste aconteceu no próprio *Raspberry Pi*, sendo seu rosto erroneamente classificado pelo sistema. Decidiu-se experimentar o reconhecimento sem os e constatou-se que a remoção dos óculos permitiu o reconhecimento correto, indicando que a presença dos óculos poderia estar interferindo na precisão do modelo de reconhecimento. Ao realizar uma análise das imagens capturadas para o reconhecimento durante o processo, percebeu-se que as imagens não reconhecidas da aluna D ou erroneamente classificadas, continham um reflexo luminoso nas lentes dos óculos devido ao padrão das lentes que a mesma usava, o que poderia estar impedindo a captura dos olhos da mesma.

Ao final do experimento, o interesse e o envolvimento dos alunos aumentou, levando-os a realizar experimentos adicionais, como o uso de bonés e óculos, embora tais variações não fossem o foco principal deste trabalho. Os alunos demonstraram satisfação com os resultados e poucas vezes relataram erros no reconhecimento. Essas observações são úteis visto que, em um cenário real de utilização, o sistema deverá lidar com uma diversidade de situações e condições como: utilização de óculos de grau e escuros, acessórios, barba, máscara, tatuagem, etc.

5.2 Experimento com simulação de múltiplos clientes

A segunda etapa dos experimentos foi realizada mediante a construção de cenários de uso em que o sistema estivesse sob grande estresse de utilização. Observando a UFC no Campus de Crateús, estimou-se que a instituição possui cerca de vinte salas disponíveis para os alunos. Dessa forma, uma análise que represente um cenário de estresse no sistema, considerando a instituição, deveria simular aproximadamente vinte salas realizando o controle de frequência simultaneamente.

Para esta etapa, foi utilizado o mesmo dispositivo servidor mencionado anteriormente além de um notebook Acer Aspire construído com um processador Intel i3 7100U 4 GB de memória RAM no lado cliente. O computador do lado cliente rodou as *threads* que simularam as salas de aula. A rede utilizada para o experimento foi uma rede doméstica com conexão de 2GHz e 5GHz. Todos os dois computadores permaneceram conectados a rede elétrica durante todo o experimento.

Foram definidos seis cenários distintos de utilização do sistema, desde o melhor caso possível até o pior, dentro das limitações técnicas de análise disponíveis. No primeiro cenário, foi criada uma *thread* que simulava uma sala de aula realizando o controle de frequência por meio de requisições ao servidor através da rede. Utilizou-se cinquenta imagens aleatórias, geradas durante a primeira etapa do experimento, para simular a captura de rostos pelo sistema. A *thread* foi executada trinta vezes com as mesmas 50 imagens, gerando, portanto, 1500 linhas em um arquivo *Comma-Separated Values* (CSV) para análise posterior. As mesmas trinta execuções foram realizadas em todas as outras simulações.

Os seis cenários eram compostos por 1, 5, 10, 15, 20 e 25 *threads*, simulando um número de salas correspondente ao número de *threads*. Assim, uma *thread* simulava uma sala, cinco *threads* simulavam cinco salas, e assim sucessivamente. Considerando que cada *thread* processava 50 imagens e era executada 30 vezes de forma simultânea, o último cenário realizou

um total de 37.500 requisições, distribuídas em 25 *threads*. É importante destacar que o conjunto de *threads* foi executado 30 vezes simultaneamente, a fim de obter uma média de tempo mais fiel à realidade, visto que uma rede doméstica apresenta variações consideráveis na qualidade de transferência de dados.

Quando o experimento estava na simulação do cenário com vinte salas de aulas (vinte *threads*), o sistema retornava um erro como se o servidor não estivesse mais na rede ou estivesse com um IP diferente do especificado no cliente. Observou-se então que o IP na versão quatro havia desaparecido na rede do dispositivo de borda mas ainda constava na rede do cliente. Ao desconectar e reconectar, o sistema voltava a funcionar, mas o teste precisava ser reiniciado e novamente parava pelo mesmo erro. A solução para o problema foi forçar a rede a utilizar somente o IP na versão quatro passando o sistema a funcionar íntegro até o final do experimento. Portanto, a partir do conjunto de vinte *threads* de execução a rede em que se encontravam os nós estava com o IP na versão seis desabilitado.

Na Tabela 4 são apresentadas as métricas extraídas do experimento, na qual se pode observar o comportamento sistema escalando à medida em que se tem mais *threads* realizando o reconhecimento ao mesmo tempo. A média e a mediana aumentam à medida que cresce o número de salas ou *threads*, o que sugere uma elevação no valor central dos dados obtidos porém ainda dentro de um valor esperando dado que a rede estava sobrecarregada. Quanto à variância e ao desvio padrão, o aumento expressivo desses indicadores revela que a dispersão dos dados também se intensifica com o acréscimo de *threads*. Essa dispersão nos dados se destaca à medida em que se tem mais requisições, possivelmente devido a problemas como sobrecarga na rede já que a mesma estava em uso e o roteador retarda a transmissão dos dados aos destinatários.

Tabela 4 – Tabela com os resultados obtidos nos testes com cenários de 1, 5, 10, 15, 20 e 25 salas/*threads*

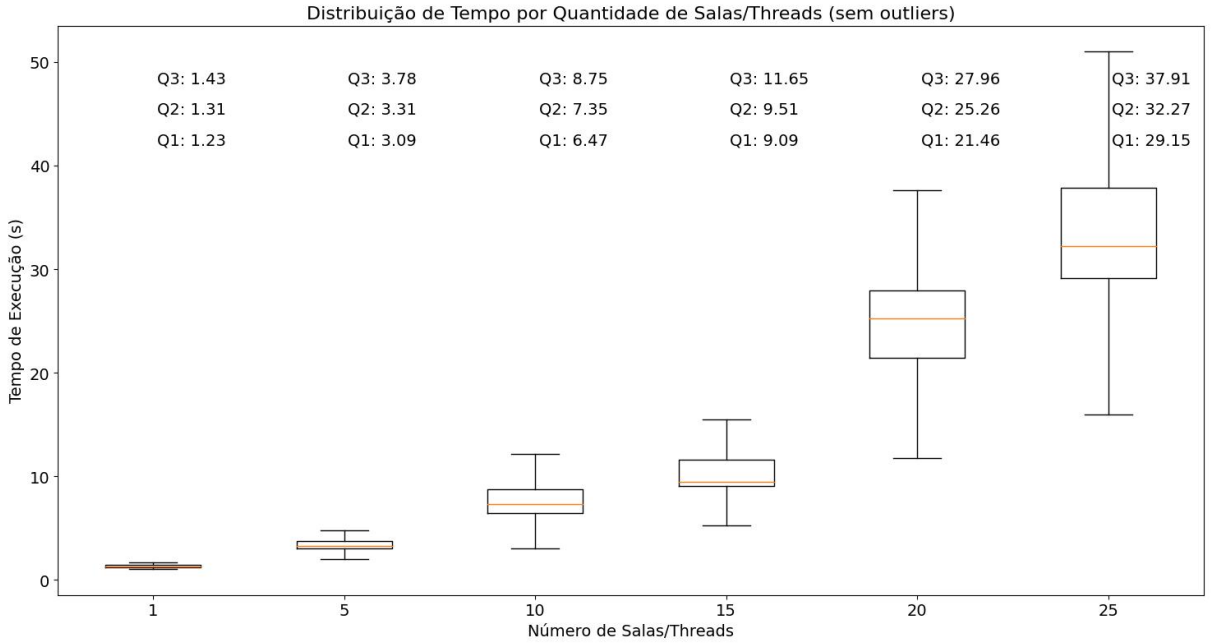
Salas	Média (s)	Mediana (s)	Variância (s)	Q1 (s)	Q3 (s)	Desvio (s)	Acertos	Total
1	1,39	1,31	0,17	1,23	1,43	0,41	84,06%	1500
5	3,69	3,31	1,32	3,09	3,78	1,15	84,00%	7500
10	7,95	7,35	5,24	6,47	8,75	2,29	84,00%	15000
15	10,31	9,51	5,30	9,09	11,65	2,30	84,00%	22500
20	28,32	25,26	146,34	21,46	27,96	12,10	84,00%	30000
25	34,41	32,27	269,67	29,15	37,91	16,42	84,23%	37500

Fonte: O autor.

O sistema demonstrou ter uma boa escalabilidade, com um aumento significativo na capacidade de processamento de imagens à medida que o número de salas/*threads* aumenta. Como as requisições partiram de um único computador com *threads* simulando salas de aula a variabilidade dos resultados aumenta com a crescente no número de requisições. Isso pode acontecer devido até a fatores como a concorrência por recursos do sistema operacional no notebook cliente ou a concorrência por recursos no servidor dado que apenas 5 threads foram dedicadas para atender as requisições.

Para facilitar a análise, a Figura 10 apresenta os resultados em forma gráfica, na qual se percebe que, dado que o sistema se encontra em um cenário de estresse muito grande, os 30 segundos para reconhecimento de um aluno pode ser bom. Isto pois, o próprio *Raspberry Pi* se mostrou capaz de fazer também o reconhecimento podendo, portanto, dividir com o servidor essa tarefa. Outro fator importante é que o servidor gRPC estava configurado para trabalhar

Figura 10 – Evolução do sistema à medida em que se tem mais salas utilizando o mesmo



Fonte: O autor.

com apenas cinco *workers*, o que limita o servidor a fazer cinco tarefas de reconhecimento simultâneas. Dado que o valor presente neste parâmetro pode ser muito maior do que cinco, é possível considerar que o mesmo poderia ter um desempenho ainda melhor.

Outra informação importante, mas que era esperada foi o número de acertos do sistema. A primeira etapa dos experimentos deste trabalho, apontou que o sistema, adotando o par detector/modelo [’ssd’/’ArcFace’] apresenta 85% de acertos no reconhecimento dos rostos dos alunos. Nesta segunda etapa, com 114000 imagens analisadas, o mesmo continua apresentando o mesmo grau de acertos em relação às mesmas imagens.

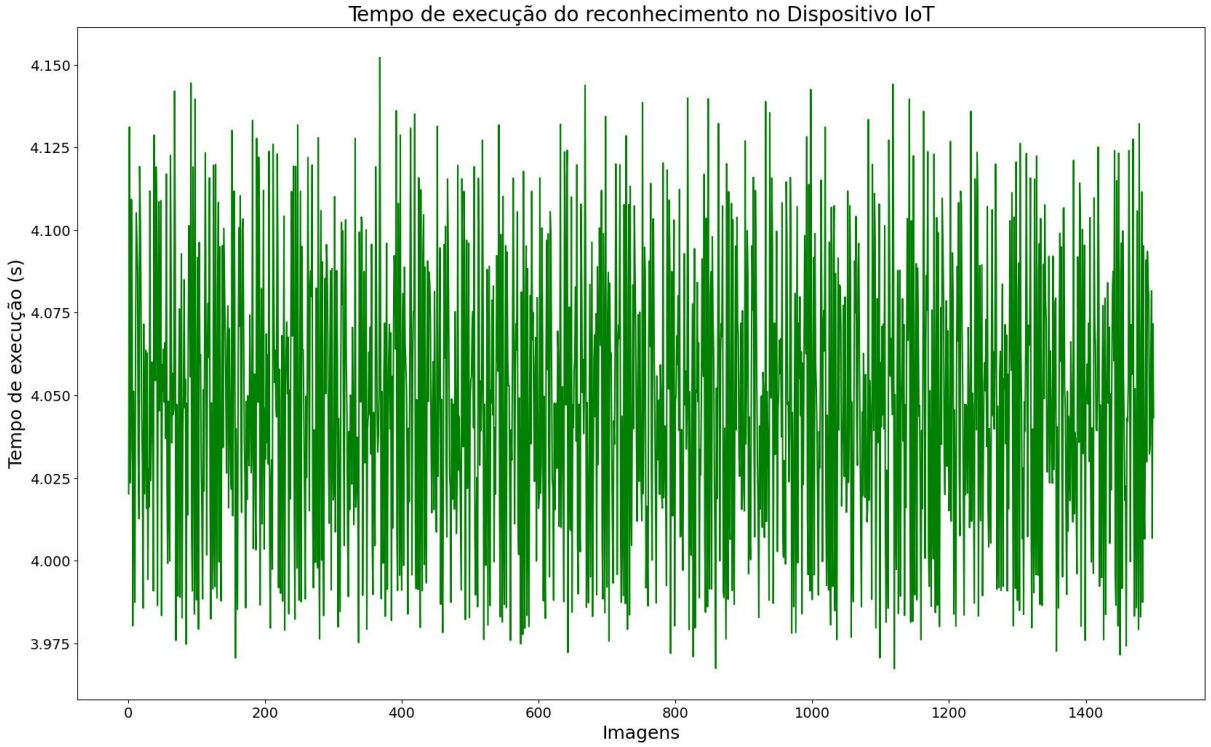
Também foi realizada uma etapa de testes com as mesmas cinquenta imagens rodando no próprio *Raspberry Pi* para extrair as informações sobre o reconhecimento neste dispositivo. Para tal, utilizou-se o mesmo par detector/modelo mencionado anteriormente e o mesmo conjunto de imagens. A partir disso, executou-se também trinta rodadas no próprio dispositivo gerando assim 1500 reconhecimentos no *Raspberry Pi*. Na Figura 11, é mostrado o resultado do experimento realizado no próprio *Raspberry Pi*. No resultado é possível perceber um tempo de mais de dez segundos para a primeira imagem, isso se deve ao fato de o sistema realizar uma varredura no sistema procurando por uma GPU dedicada para acelerar o reconhecimento da imagem. Caso não seja encontrada uma placa disponível, o sistema não fará mais essa verificação tornando os próximos reconhecimentos mais rápidos.

6 CONCLUSÕES E TRABALHOS FUTUROS

A utilização de dispositivos IoT no ambiente de sala de aula é importante para otimizar o dia a dia dos profissionais de educação, mas também para produzir dados úteis à instituição. Um sistema de visão computacional sendo executado em um dispositivo de baixo consumo energético e de alto grau de portabilidade se apresenta como oportunidade para diversas soluções para problemas do cotidiano.

O presente estudo demonstrou a viabilidade de um sistema de frequência escolar baseado em visão computacional, com o uso de um dispositivo *Raspberry Pi*, integrado a

Figura 11 – Execução de teste com 56 imagens a serem reconhecidas no próprio *Raspberry Pi*



Fonte: O autor

tecnologias de IoT. Mesmo com as limitações de performance associadas à linguagem *Python*, o sistema foi capaz de realizar o reconhecimento facial dos alunos com 85% de acerto. O mesmo ainda findou proporcionando tempos de resposta adequados para a tarefa de registrar a presença em sala de aula para cenários de até quinze requisições simultâneas. É importante esclarecer que esse cenário é crítico pois representa um situação em que quinze alunos estão registrando sua frequência no mesmo exato instante. Esse desempenho razoável é relevante, especialmente em ambientes educacionais, onde a otimização do tempo destinado ao aprendizado é crucial.

Além disso, a estratégia de *offloading* para um dispositivo de borda adicional se mostrou eficaz, possibilitando a distribuição do processamento e, consequentemente, contribuindo para a escalabilidade do sistema. Isso reforça a adequação da solução proposta para cenários que demandam maior capacidade de processamento, sem sobrecarregar o dispositivo principal.

Os testes realizados com pessoas reais e em simulações de cenários de alto estresse computacional confirmaram que o sistema é robusto o suficiente para lidar com demandas complexas, mantendo a precisão no reconhecimento facial e a agilidade na transferência de dados.

Para trabalhos futuros, sugere-se o teste utilizando uma rede dedicada ao serviço e com maior capacidade de tráfego. Além de:

1. Melhorar a acessibilidade do sistema, bem como implementar um serviço de áudio descrição para deficientes visuais e adicionar uma segunda câmera para cadeirantes ou facilitar o alcance da câmera para os mesmo;
2. Investigar se é possível adaptar o sistema para identificar pessoas com acessórios como bonés, tatuagem, barba, mascaras, etc;
3. Melhorar as configurações do servidor, por exemplo, aumentado o número de *workers*;
4. Desenvolver estratégias para distribuir melhor o processamento de forma que parte das imagens sejam processadas no *Raspberry Pi* e outra seja enviada ao servidor. Podendo

ainda, utilizar o servidor apenas para extrair as características e realizar a comparação com as imagens do banco;
5. Reproduzir o experimento com as *threads* utilizando conjuntos de *Raspberry Pi* realizando requisições ao servidor;
6. Realizar os mesmos experimentos utilizando outras bibliotecas e linguagens em trechos que se mostrem críticos;
7. Propor uma arquitetura do sistema que busque maior manutenibilidade e otimize o fluxo do mesmo;
8. Melhorar o sistema de captura da imagem podendo, por exemplo, testar a viabilidade de capturar três imagens e decidir no próprio *Raspberry Pi* qual delas possui melhor qualidade.

REFERÊNCIAS

BARBOSA, X. de C. Breve introdução à história da inteligência artificial. **Jamaxi**, v. 4, n. 1, 2020.

BRAHMBHATT, S. **Practical OpenCV**. [*S. l.*]: Apress, 2013.

CHANDRASEKHARAN, K. **Essentials of cloud computing**. [*S. l.*]: CRC Press, 2014.

COPPIN, B. **Artificial intelligence illuminated**. [*S. l.*]: Jones & Bartlett Learning, 2004.

COSTA, P. B. Uma abordagem para offloading em múltiplas plataformas móveis. 2014.

DE, D. **Mobile cloud computing: architectures, algorithms and applications**. [*S. l.*]: CRC Press, 2016.

DIANA, M.; CHIKAMA, J.; AMAGASAKI, M.; IIDA, M. Low-cost image search system on off-line situation. **Electronics**, MDPI, v. 9, n. 1, p. 153, 2020.

EJIMA, T.; HATANO, T. Quantum efficiency of back-illuminated cmos sensor with 1.4$\times$ 1.4 μm2 pixel size. **Microscopy and Microanalysis**, Cambridge University Press, v. 24, n. S2, p. 320–321, 2018.

GAY, W. **Advanced raspberry pi: raspbian linux and GPIO integration**. [*S. l.*]: Apress, 2018.

GOMES, F. A.; REGO, P. A.; ROCHA, L.; SOUZA, J. N. de; TRINTA, F. Caos: A context acquisition and offloading system. In: IEEE. **2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)**. [*S. l.*], 2017. v. 1, p. 957–966.

HERMANS, A.; BEYER, L.; LEIBE, B. In defense of the triplet loss for person re-identification. **arXiv preprint arXiv:1703.07737**, 2017.

HOSAKI, G. Y. G. Y.; RIBEIRO, D. F. Deep learning: ensinando a aprender. 275, 2021.

HUSSAIN, T.; HUSSAIN, D.; HUSSAIN, I.; ALSALMAN, H.; HUSSAIN, S.; ULLAH, S. S.; AL-HADHRAMI, S. Internet of things with deep learning-based face recognition approach for authentication in control medical systems. **Computational and Mathematical Methods in Medicine**, v. 2022, 2022. Cited by: 12; All Open Access, Gold Open Access, Green Open Access. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85125124694&doi=10.1155%2f2022%2f5137513&partnerID=40&md5=1ea8d3e876d36bda9425d768247c3cb2.

IBM. **What is Machine Learning?** United States: [*S. n.*], 2023. Disponível em: https://www.ibm.com/topics/machine-learning. Acesso em: 28 Nov. 2023.

INDRASIRI, K.; KURUPPU, D. **gRPC: up and running: building cloud native applications with Go and Java for Docker and Kubernetes**. [*S. l.*]: O'Reilly Media, 2020.

KHAN, M. Z.; HAROUS, S.; HASSAN, S. U.; KHAN, M. U. G.; IQBAL, R.; MUMTAZ, S. Deep unified model for face recognition based on convolution neural network and edge computing. **IEEE Access**, v. 7, p. 72622 – 72633, 2019. Cited by: 94; All Open Access, Gold Open Access. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85067410048&doi=10.1109%2fACCESS.2019.2918275&partnerID=40&md5=2cffe8f2829eb43619a74dd6982d74d1.

KHAN, S.; AKRAM, A.; USMAN, N. Real time automatic attendance system for face recognition using face api and opencv. **Wireless Personal Communications**, v. 113, n. 1, p. 469 – 480, 2020. Cited by: 40. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083157811&doi=10.1007%2fs11277-020-07224-2&partnerID=40&md5=820cab8f2f2b56211c2c0cb48e9cff78.

MAGALHÃES, W.; FARIAS, M.; MARINHO, L.; GOMES, H.; AGUIAR, G.; SILVEIRA, P. Evaluating edge-cloud computing trade-offs for mobile object detection and classification with deep learning. **Journal of Information and Data Management**, v. 11, n. 1, 2020.

MATOS, F. F. S. B. de; REGO, P. A. L.; TRINTA, F. A. M. Secure computational offloading with grpc: A performance evaluation in a mobile cloud computing environment. In: **Proceedings of the 11th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications**. New York, NY, USA: Association for Computing Machinery, 2021. (DIVANet '21), p. 45–52. ISBN 9781450390811. Disponível em: https://doi.org/10.1145/3479243.3487295.

MILANO, D. de; HONORATO, L. B. Visão computacional. **UNICAMP Universidade Estadual de Campinas FT Faculdade de Tecnologia**, 2014.

MUTTAQIN, R.; FUADA, S.; MULYANA, E. *et al.* Attendance system using machine learning-based face detection for meeting room application. **International Journal of Advanced Computer Science and Applications**, Science and Information (SAI) Organization Limited, v. 11, n. 8, 2020.

PAIVA, E. S.; SILVA, O. R. da. O uso do tempo pedagógico numa escola de tempo integral do ensino médio. **VI CONGRESSO NACIONAL DE EDUCAÇÃO**, 2020.

QIAN, L.; LUO, Z.; DU, Y.; GUO, L. Cloud computing: An overview. In: SPRINGER. **Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1**. [*S. l.*], 2009. p. 626–631.

RAUBER, T. W. Redes neurais artificiais. **Universidade Federal do Espírito Santo**, v. 29, 2005.

REGO, P. A.; TRINTA, F. A.; HASAN, M. Z.; SOUZA, J. N. de *et al.* Enhancing offloading systems with smart decisions, adaptive monitoring, and mobility support. **Wireless Communications and Mobile Computing**, Hindawi, v. 2019, 2019.

RUSSELL, S. J. **Artificial intelligence a modern approach**. [*S. l.*]: Pearson Education, Inc., 2010.

SAAVEDRA, M. R. M.; SOUZA, F. R. A. de; ARAUJO, J. de S. O uso de placa gráfica aplicada ao reconhecimento facial: Um estudo de caso com redes neurais artificiais. **Anais da II Escola Regional de Alto Desempenho Norte 2 e II Escola Regional de Aprendizado de Máquina e Inteligência Artificial Norte 2**, SBC, p. 13–16, 2022.

SANTOS, B. P.; SILVA, L. A.; CELES, C.; BORGES, J. B.; NETO, B. S. P.; VIEIRA, M. A. M.; VIEIRA, L. F. M.; GOUSSEVSKAIA, O. N.; LOUREIRO, A. Internet das coisas: da teoria à prática. **Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos**, v. 31, p. 16, 2016.

SANTOS, G. B. d. Uma proposta de solução para offloading de métodos entre dispositivos móveis. 2017.

SATYANARAYANAN, M.; BAHL, P.; CACERES, R.; DAVIES, N. The case for vm-based cloudlets in mobile computing. **IEEE pervasive Computing**, IEEE, v. 8, n. 4, p. 14–23, 2009.

SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S. l.*: *s. n.*], 2015. p. 815–823.

SERENGIL, S. I.; OZPINAR, A. Lightface: A hybrid deep face recognition framework. In: IEEE. **2020 innovations in intelligent systems and applications conference (ASYU)**. [*S. l.*], 2020. p. 1–5.

SOBIN, C. A survey on architecture, protocols and challenges in iot. **Wireless Personal Communications**, Springer, v. 112, n. 3, p. 1383–1429, 2020.

TAIGMAN, Y.; YANG, M.; RANZATO, M.; WOLF, L. Deepface: Closing the gap to human-level performance in face verification. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S. l.*: *s. n.*], 2014. p. 1701–1708.

THORAT, S.; NAYAK, S.; DANDALE, J. P. Facial recognition technology: An analysis with scope in india. **arXiv preprint arXiv:1005.4263**, 2010.

WANG, F.; DIAO, B.; SUN, T.; XU, Y. Data security and privacy challenges of computing offloading in fins. **IEEE Network**, IEEE, v. 34, n. 2, p. 14–20, 2020.

WANG, M.; DENG, W. Deep face recognition: A survey. **Neurocomputing**, Elsevier, v. 429, p. 215–244, 2021.

ZAMONER, M. Gestão do tempo escolar: a questão das interrupções de aulas. In: **V Educere-PUCPR-III Congresso Nacional da Área de Educação. Curitiba**. [*S. l.*: *s. n.*], 2005.