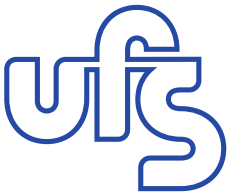


UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

AtalaIA: Compressão de modelos de detecção facial para dispositivos embarcados de baixo custo

Trabalho de Conclusão de Curso

Luan Fabrício de Carvalho Lima Leite

São Cristóvão – Sergipe

2025

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Luan Fabrício de Carvalho Lima Leite

AtalaIA: Compressão de modelos de detecção facial para dispositivos embarcados de baixo custo

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Leonardo Nogueira Matos
Coorientador(a): Rafael Andrade da Silva

São Cristóvão – Sergipe

2025

Dedico este trabalho a minha família, amigos, professores e todos que de alguma forma me ajudaram a chegar até aqui.

Resumo

Redes Neurais Convolucionais estão ficando cada vez mais populares para solução de diversos desafios, sendo um deles o de reconhecimento facial, que é uma das tarefas onde essa abordagem já supera o ser humano. Entretanto, esse método costuma exigir um alto poder de processamento e quantidade de memória, o que acaba limitando o seu uso em casos de computação de borda com dispositivos embarcados. Este trabalho tem como foco tratar esse problema, comprimindo um modelo de reconhecimento facial para que ele seja embarcado e consiga realizar operações na borda, mantendo a acurácia alta e tempo de resposta baixo. Para que esse objetivo seja cumprido, será necessário utilizar um modelo como base, para que ele seja comprimido e avaliado, onde ele será escolhido a partir da comparação de soluções já existentes e utilizadas. O modelo embarcado será avaliado com base em métricas como acurácia, F1 score, latência e ocupação de memória e comparado a outras soluções existentes.

Palavras-chave: CNN, Compressão de modelos, Visão computacional, Sistemas embarcados, Computação em borda, Reconhecimento facial

Abstract

Convolutional Neural Networks are becoming increasingly popular for solving various challenges, one of them being facial recognition, which is one of the tasks that this approach overcomes humans. However, this method usually requires a high computational power and amount of memory, which ends up limiting its use case in edge computing for embedded devices. This work focuses on addressing this issue, compressing a facial recognition model so that it is embedded and can perform operations at the edge, maintaining high accuracy and low response time. For this objective to be achieved, it will be necessary to use a model as a basis, so that it can be compressed and evaluated, where it will be chosen based on the comparison of the existing and used solutions. The embedded model will be evaluated based on metrics such as accuracy, F1 score, latency, and memory footprint, and compared to other existing solutions.

Keywords: CNN. Model compression. Computer Vision. Embedded systems. Edge computing. Facial recognition.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de códigos

Lista de algoritmos

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
abnTeX	ABsurdas Normas para TeX
DCOMP	Departamento de Computação
UFS	Universidade Federal de Sergipe
ANN	Rede Neural Artificial ou *Artificial Neural network*
CNN	Rede Neural Convolucional ou *Convolutional Neural Network*
KB	Kilobytes
MB	Megabytes
API	Interface de Programação de Aplicações ou *Application Program Interface*
IoT	Internet das Coisas ou *Internet of Things*

Lista de símbolos

α	Letra grega alfa
Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
$\in$	Pertence

Sumário

1 Introdução

Redes Neurais Artificiais, ou *Artificial Neural Network* (ANN), são ferramentas poderosas para auxiliar a sociedade. Podendo ser utilizadas em diversas tarefas, como reconhecimento facial, onde a rede é treinada para realizar a classificação da face da pessoa, permitindo que ela seja usada em várias áreas diferentes, indo de entretenimento até segurança.

1.1 Motivação

O uso de Redes Neurais Artificiais vem crescendo bastante no ramo de computação visual, principalmente desde 2012, quando Redes Neurais Convolucionais ou *Convolutional Neural Networks* (CNN) começaram a ser utilizadas para classificação de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Um dos usos desse tipo de rede é na detecção de face, que é muito relevante para a área de segurança e vigilância, onde o modelo pode fazer a detecção do rosto de uma pessoa, abrir uma porta ou enviar uma notificação para algum segurança. Porém, essa abordagem necessita de uma quantidade elevada de poder computacional, tornando inviável que tal tipo de produto seja embarcado e mantenha um baixo tempo de resposta, o que pode atrapalhar a experiência do usuário, ou reduzir a efetividade da ação que será tomada.

Um dos principais problemas das Redes Neurais Profundas, como CNN, é que elas necessitam de um alto processamento e uso de memória, o que acaba dificultando a sua execução em dispositivos com poder computacional e memória limitados (como os dispositivos embarcados). Porém, existem técnicas que podem ser aplicadas para reduzir o poder computacional necessário, como o uso de destilação de conhecimento (HINTON; VINYALS; DEAN, 2015), poda e quantização. Possibilitando a implantação do modelo em sistemas embarcados na borda, de forma que a latência do dispositivo seja baixa.

1.2 Objetivos

O objetivo deste trabalho é comprimir uma Rede Neural Convolucional, permitindo que ela realize o reconhecimento facial em microcontroladores, na borda. Nele também serão tratadas formas de comprimir e otimizar o modelo, para que a sua versão final consiga ser embarcada em um dispositivo com hardware limitado, mantendo acurácia alta e baixa latência.

1.2.1 Objetivo específicos

O trabalho estará completo se os seguintes objetivos forem alcançados:

- Reduzir o custo computacional de um modelo, possibilitando que ele seja embarcado enquanto a capacidade de reconhecimento facial é mantida.
- Validar performance do modelo, com a finalidade de validar que a acurácia e F1-Score sejam preservados ou próximos.
- Embarcar o modelo comprimido de forma que ele consiga realizar operações na borda, mantendo acurácia alta e baixa latência.

1.3 Metodologia

Para atingir o objetivo do estudo, foi necessário dividir o processo em algumas etapas, cada uma sendo essencial para que o objetivo do trabalho fosse atingido. Sendo elas:

1. **Levantamento do estado da arte:**

   Nessa etapa, são selecionados artigos que possuem o objetivo similar ao deste artigo, com base nesses artigos serão testadas novas técnicas para compressão de modelos.

2. **Reprodução do estado da arte:**

   a) **Seleção de base e treino de modelos:**

   Nesta etapa, uma base de dados é selecionada e a partir dela serão desenvolvidos modelos, com o objetivo de atingir uma alta acurácia, sem sofrer *overfitting*.

   Nesta etapa, uma base de dados é selecionada e a partir dela modelos foram validados e desenvolvidos, com o objetivo de atingir uma alta acurácia, sem sofrer *overfitting*.

   b) **Aplicação de técnicas de compressão para Modelos:**

   Após definir e treinar os modelos, serão aplicadas técnicas de compressão, tendo como objetivo ter uma acurácia parecida com a do modelo original. Onde as técnicas aplicadas foram: poda, quantização e destilação de conhecimento.

c) **Avaliação do desempenho:**

Depois de treinar e aplicar técnicas de compressão, os dados dos modelos serão coletados e avaliados. Para realizar essa avaliação, será necessário utilizar um conjunto de testes. Nesta avaliação, foram medidos gastos computacionais do modelo quanto o seu desempenho para realizar a tarefa de reconhecimento facial.

Onde pegada de memória e tempo de inferência são métricas referente ao custo computacional, e acurácia e F1-Score foram métricas do desempenho na tarefa de reconhecimento facial.

d) **Análise e comparação dos resultados:**

Para finalizar, os dados dos modelos serão comparados e analisados. Com o objetivo de identificar o melhor modelo e descobrir quais foram os motivos para que esse modelo tenha se saído melhor, mesmo após a aplicação de compressão. Nesta etapa as métricas de acurácia e tamanho do modelo são avaliadas.

3. **Escolha do modelo para reconhecimento facial:**

   Nesta etapa serão avaliados os modelos com base em métricas como acurácia, F1 score, latência e ocupação de memória. Após a avaliação será escolhido um modelo que servirá como base nas próximas etapas.

4. **Implantação do modelo em hardware limitado:**

   Com o modelo final pronto e avaliado, ele será adaptado para ser implantado em um dispositivo embarcado.

5. **Desenvolvimento do estudo de caso:**

   Com o modelo final comprimido ao ponto de ser implantado em um sistema embarcado, será desenvolvida uma aplicação que servirá como experimento para avaliar a eficácia do modelo dentro de dispositivos com hardware limitado.

1.4 Estrutura do documento

Este documento foi dividido em capítulos, onde cada um apresenta uma proposta diferente:

- Capítulo 2 - **Conceitos Básicos**: Apresenta os tópicos principais para o entendimento do trabalho.
- Capítulo 3 - **Trabalhos Relacionados**: Apresenta uma revisão dos trabalhos relacionados ao tema do trabalho.
- Capítulo 4 - **Resultados Preliminares**: Apresenta os resultados preliminares dos experimentos realizados durante o trabalho.

- Capítulo 5 - **Planos de continuidade**: Contém o planejamento da continuidade do Trabalho de Conclusão 2.

2

Conceitos básicos

O foco deste capítulo é introduzir os tópicos mais relevantes para o trabalho, de forma que o leitor consiga entender o conteúdo independente de conhecimento prévio. Neste capítulo serão abordados os tópicos relacionados a ANN (seção 2.1), CNN (seção 2.2), *Data augmentation* (seção 2.3), transferência de conhecimento (seção 2.4), técnicas de compressão para redes neurais (seção 2.5) e otimização Bayesiana (seção 2.6).

2.1 Redes Neurais Artificiais

Redes Neurais Artificiais é composta por neurônios interconectados, onde cada um é responsável por fazer um processamento simples. Dentro dessa estrutura cada neurônio reforça ou enfraquece a conexão com um dos neurônios da camada anterior, assim replicando o processo de aprendizagem do cérebro humano (FACELI et al., 2011). A Figura 1 ilustra uma ANN simples.

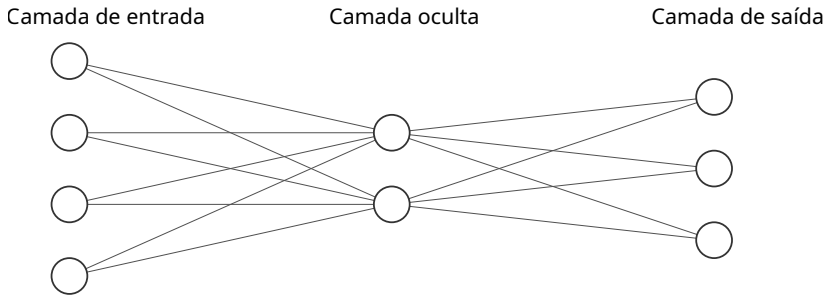


Figura 1 – Exemplo de uma ANN

Fonte: Autor

O neurônio é uma parte fundamental de uma ANN, nele que o aprendizado é armazenado através do reforço de conexões com outros neurônios. Esse reforço é o peso da conexão, ele é multiplicado pela entrada e somado com os outros valores, como é demonstrado na equação

2.1, onde x é um vetor com os valores de entrada do neurônio, w é um vetor com os pesos de cada entrada e b é o viés (*bias*) do modelo. Depois disso, os valores passam por uma função de ativação $g(x)$ (2.2), que é responsável por transformar estes dados antes que sejam passados para a próxima etapa, por esta razão, a função de ativação também é chamada função de transferência.

$$u = \sum x_i w_i \tag{2.1}$$

$$y = g(u + b) \tag{2.2}$$

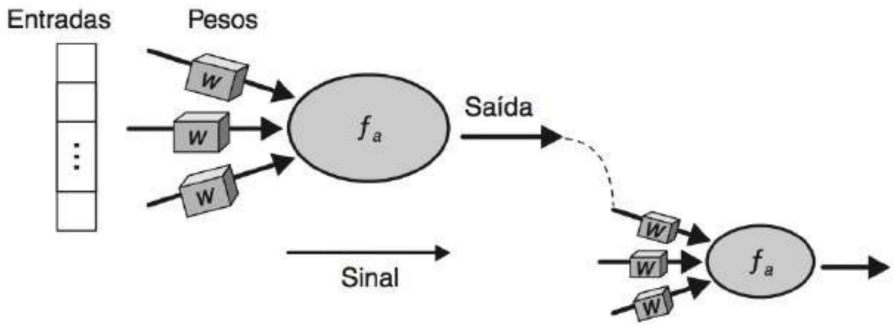


Figura 2 – Exemplo de um neurônio artificial

Fonte: (FACELI et al., 2011)

2.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais são Redes Neurais Artificiais que utilizam a operação de convolução para o processamento e análise de dados no formato de *grid* (grade). Por exemplo, uma imagem, que pode ser representada no formato 2-D (GOODFELLOW; BENGIO; COURVILLE, 2016).

A arquitetura de uma CNN tem como componentes principais as camadas convolucionais (subseção 2.2.1), que tem como o objetivo extrair as características dos dados de entrada; *pooling* (subseção 2.2.2), que realça certos pontos da sua entrada, reduzindo o tamanho final da sua saída; a camada totalmente conectada (subseção 2.2.3), que aprende a interpretar esses dados, para que a rede consiga realizar o processo de classificação. Na Figura 3 é mostrada a ilustração da arquitetura de uma CNN.

2.2.1 Camada de Convolução

Nessa camada são aplicados filtros (matriz de pesos) nos dados de entrada, onde esses filtros deslizam ao longo da grade de entrada executando operações de multiplicação e soma em cada elemento da matriz de entrada, com o objetivo de gerar um mapa de características (*feature*

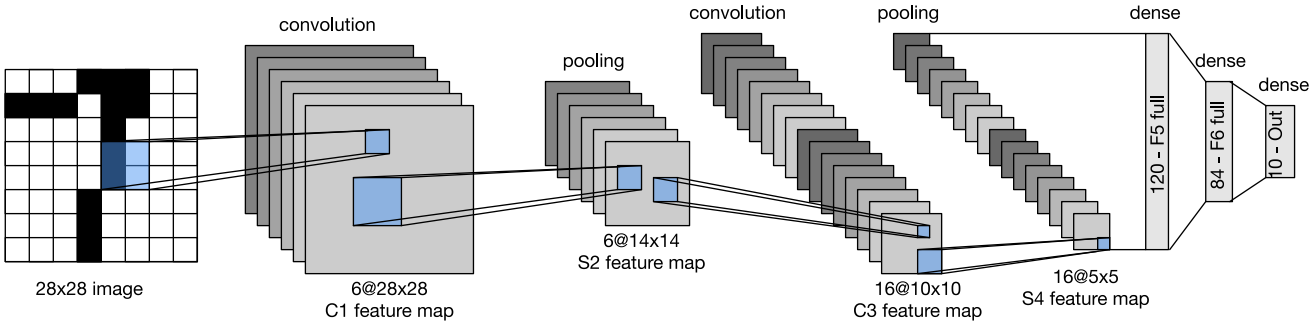
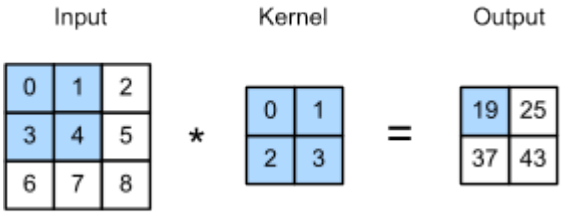


Figura 3 – Arquitetura da LeNet

Fonte: Zhang et al. (2023)

map). O objetivo desses filtros é realçar as características dos dados de entrada. Alguns padrões como curvas e linhas podem ser reconhecidos por estas operações de filtragem. A Figura 4, é um exemplo da operação de convolução sendo aplicada em uma matriz.

Figura 4 – Exemplo de convolução



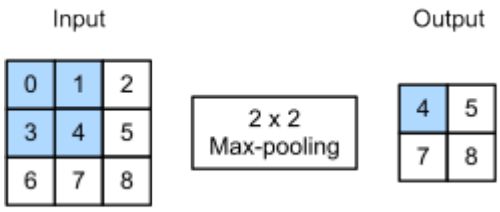
Fonte: Zhang et al. (2023)

2.2.2 Camada de *pooling*

A abordagem da camada de *pooling* é relativamente parecida com a camada de convolução, uma matriz desliza pelas células da imagem, salvando apenas um dos valores dessa área na matriz de saída. Esta operação reduz o tamanho da matriz de entrada, fazendo com que o poder computacional necessário seja reduzido, junto com o uso de memória.

Existem diversos tipos de *pooling*, min, *average* e max, onde cada um foca em extrair um valor dos dados de entrada na matriz. O tipo mais comum de *pooling* é o max, que salva apenas o maior valor da área, além de reduzir o tamanho da matriz de entrada ele consegue realçar algumas características mais expressivas da matriz. A Figura 5 demonstra uma operação de max *pooling* em uma matriz.

Figura 5 – Exemplo de max *pooling*



Fonte: Zhang et al. (2023)

2.2.3 Camada totalmente conectada

A camada totalmente conectada (*fully connected layer*) é uma das últimas camadas de uma CNN. Depois das camadas anteriores extraírem as características da imagem, ela é responsável por aprender a interpretar essas características e inferir um resultado a partir do seu treinamento. Essa camada é uma ANN (seção 2.1) que geralmente é focada em realizar a classificação dos dados de entrada.

2.3 *Data augmentation*

CNN tem um bom desempenho em tarefas de visão computacional. Entretanto, esse tipo de rede neural precisa de uma grande quantidade de dados no treinamento para não sofrer de superajuste (*overfitting*) (SHORTEN; KHOSHGOFTAAR, 2019). O objetivo do *data augmentation* (aumento de dados) é gerar mais dados a partir de um conjunto de dados que já existe, aplicando algumas transformações geométricas ou espaciais, ou realizando injeção de ruído nas imagens originais.

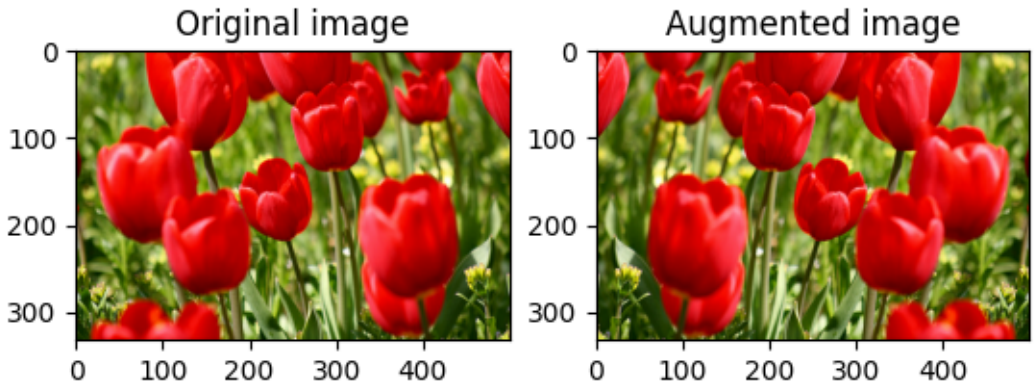
Na Figura 6, é apresentado um exemplo de uma imagem que sofreu uma transformação para efetuar um *data augmentation*. Nesse caso, uma que foi revertida (*flip*), gerando um novo dado para o dataset

2.4 Transferência de conhecimento

Transferência de conhecimento consiste em usar um modelo pré-treinado em uma base de dados específica e aproveitar o conhecimento adquirido durante esse treinamento para um novo conjunto de dados.

Para realizar a transferência de conhecimento é necessário adaptar a camada de entrada e de saída (totalmente conectada) do modelo base, para que ocorra um pré-processamento dos dados de entrada (antes deles serem passados para o modelo base), além disso é necessário definir e treinar a camada totalmente conectada com o *dataset* do problema.

Figura 6 – Exemplo de uma transformação utilizando *data augmentation*



Fonte: TensorFlow (2024)

2.5 Métodos de compressão para Redes Neurais

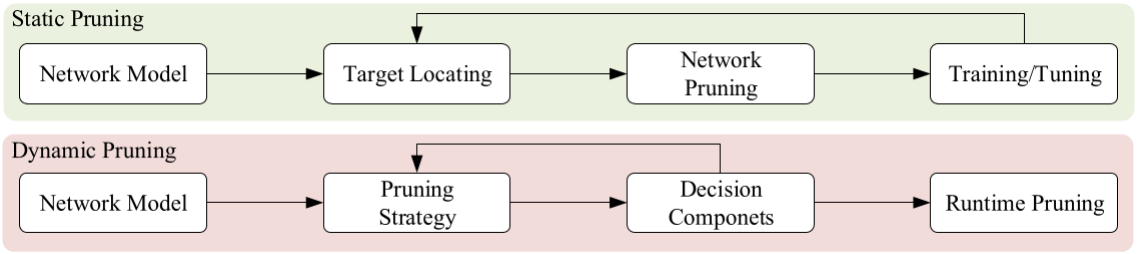
ANN são utilizadas em várias aplicações, demonstrando habilidades no campo de visão computacional. No entanto, redes com arquiteturas complexas são um desafio para a implantação em tempo real e necessitam de uma grande quantidade de energia e poder computacional (LIANG et al., 2021). Por causa disso foram desenvolvidos métodos para reduzir o tamanho dessas redes, as tornando mais eficientes. Nesse trabalho os métodos de poda (subseção 2.5.1), quantização (subseção 2.5.2) e destilação do conhecimento (subseção 2.5.3) serão usados.

2.5.1 *Pruning*(Poda)

A poda de redes neurais tem como foco eliminar conexões ou neurônios que não apresentam uma grade contribuição para a rede. Essa operação, é muito vantajosa para diminuir a pegada de memória (*memory footprint*) da rede, pois ela reduz a quantidade de parâmetros redundantes ou que não contribuem muito para a precisão dos resultados.

A operação de poda procura pesos com valores abaixo de um determinado limiar e os muda para a zero, assim deixando a rede neural mais esparsa, o que facilita o processo de compressão. O processo de poda pode reduzir o *overfitting* da rede, uma vez que remove pesos pouco importantes ou redundantes da rede. Esse processo geralmente é divido em dois, poda estática (*static pruning*), que tem todas as etapas de poda executadas off-line (antes da inferência), e poda dinâmica (*dynamic pruning*), que é realizada junto com o processo de execução do modelo, permitindo que os nós relevantes sejam identificados. A Figura 7 ilustra o fluxo dos dois tipos de poda.

Figura 7 – Fluxo dos tipos de poda



Fonte: (LIANG et al., 2021)

2.5.2 Quantização

Quantização reduz a computação diminuindo a precisão dos tipos de dados. Pesos, *bias* (vieses) e ativações geralmente devem ser quantizadas para inteiros de 8 bits, embora implementações menores que 8 bits sejam discutidas incluindo redes neurais binárias. (LIANG et al., 2021)

2.5.3 Destilação de conhecimento (Professor-Aluno)

Destilação de conhecimento ou *knowledge distillation* (HINTON; VINYALS; DEAN, 2015), é uma técnica que tem como objetivo treinar um modelo Aluno (menor e sem pré-treinamento) com um modelo Professor (maior e com pré-treinamento). Ela é amplamente utilizada para as áreas de visão computacional e linguagem natural, e tem como objetivo reduzir o tamanho do modelo final (Aluno).

Para transferir o conhecimento do modelo Professor para o Aluno, a técnica utiliza os *logits* (entrada da função de ativação final *softmax*) no lugar da classe prevista. Além disso, são utilizado os *soft targets* (probabilidades das classes previstas pelo modelo Professor) junto com os *hard targets* (classe esperada).

2.6 Otimização Bayesiana

A otimização Bayesiana é um método utilizado para a otimização de hiperparâmetros em modelos de aprendizagem de máquina, especialmente do tipo caixa preta (*black box*), como CNN. Esse método de otimização possui dois componentes principais, o modelo estatístico Bayesiano, que serve para modelar a função alvo, e a função de aquisição, que serve para escolher a próxima amostra de dados (FRAZIER, 2018). Durante o teste inicial, o modelo escolhe os hiperparâmetros de forma aleatória, para aprender o comportamento da CNN. Depois disso, no processo iterativo, o modelo começa a convergir para uma combinação de hiperparâmetros otimizados, aumentando a acurácia da rede.

3

Trabalhos relacionados

Neste capítulo serão discutidos os trabalhos relacionados à compressão de CNN com foco em dispositivos embarcados.

3.1 Trabalhos acadêmicos

Os seguintes critérios de busca foram utilizados para filtrar os trabalhos acadêmicos:

- Os artigos devem ser relacionado ao tema de CNN com compressão para sistemas embarcados.
- Trabalhos publicados entre 2020 e 2023.
- Trabalhos escritos em inglês.

  As bases utilizadas foram: IEE Eletronic Library, ACM Digital Library e Science Citation Index Expanded. Utilizando a seguinte string de busca:

- "CNN"AND "EMBEDDED"AND "Edge devices"AND ("Pruning"OR "Knowledge distillation"OR "Quantization")

3.1.1 *A Resource Constrained Pipeline Approach to Embed Convolutional Neural Models (CNN)*

O objetivo deste trabalho de dissertação (SILVA, 2022) é elaborar um modelo de detecção de placas de trânsito que seja computacionalmente e energeticamente barato. Para atingir esse objetivo, foi elaborada uma pipeline de compressão, começando pela destilação de conhecimento, e partindo para poda e quantização.

O resultado alcançado foi uma CNN capaz de detectar placas de trânsito, consumindo 59KB de espaço, com 85, 91% de acurácia e F1-Score igual a 85, 80%, atingindo um tempo de inferência de 80 ms no ESP32 e 83 ms no ESP32-2.

3.1.2 *Intelligent security system based on face recognition and IoT*

Este artigo (BAGCHI et al., 2022) tem como objetivo desenvolver um protótipo de sistema de segurança baseado em reconhecimento facial, utilizando *Internet of Things* (IOT). Para atingir esse objetivo, os autores utilizam um ESP32-CAM como componente principal para realizar a coleta e reconhecimento da face.

O resultado alcançado foi um protótipo que é capaz de verificar se a face capturada é conhecida, caso seja, ela acende um LED verde, caso contrário um LED vermelho é ligado. Além disso, foi levantada uma lista de pontos fortes, fracos, ameaças e oportunidade do protótipo.

Tabela 1 – Pontos fortes, fracos, oportunidades e ameaças

Pontos fortes	Eficiência energética Custo x benefício Design robusto Pode armazenar até 200 faces
Pontos fracos	Precisa de conexão com WiFi constante Alcance de detecção facial limitado
Oportunidades	Sistema de segurança com baixo custo Usado em sistema de atendimento Usado em sistema de pagamento
Ameaça	Não aplicável para usos ao ar livre

Fonte: Bagchi et al. (2022)

3.1.3 *A face recognition application for Alzheimer's patients using ESP32-CAM and Raspberry Pi*

Este trabalho (KADHIM et al., 2023) tem como objetivo, desenvolver um aparelho para auxiliar pessoas com Alzheimer, utilizando reconhecimento facial. Para atingir esse objetivo, foi utilizada uma ESPCAM, para fazer a detecção facial e um Raspberry Pi, para fazer o reconhecimento facial. De forma que, a câmera detecta a face, realiza um recorte (*crop*) e a envia para o servidor realizar o reconhecimento facial, onde, caso seja uma face conhecida, o nome da pessoa será falado por um alto-falante (*speaker*), como é possível notar na Figura 8.

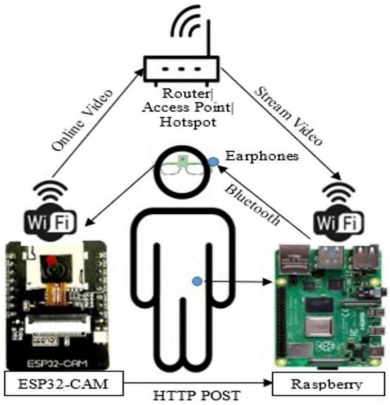


Figura 8 – Pipeline do reconhecimento facial

Fonte: Kadhim et al. (2023)

4

Experimentos e resultados

Neste capítulo serão apresentados os experimentos feitos durante o TCC. Eles tiveram como finalidade avaliar o uso de métodos e técnica de compressão de modelos, focado em dispositivos embarcados. Nele serão apresentados o dispositivo utilizado (seção 4.1), modelos usados (seção 4.2), métodos de treinamento (seção 4.3), o método de avaliação do modelo (seção 4.4), *datasets* utilizados (seção 4.5) e resultados (seção 4.6).

4.1 Dispositivos utilizados

O ESP32-S3 N16R8 foi escolhido pelo seu custo-benefício, sendo um aparelho poderoso considerando o seu baixo custo e consumo energético. Além disso, esse dispositivo possui 16 MB de memória flash e 8 MB de PSRAM, permitindo que modelos maiores sejam portados para esse dispositivo sem redução de parâmetros, podendo afetar o tempo de inferência do modelo.

Para executar os modelos no ESP foi utilizado o projeto person-detection do repositório esp-tflite-micro da Espressif [1]. A partir desse projeto, os modelos foram portados, sendo necessário realizar algumas alterações no projeto, para suportar o tamanho dos modelos e suas camadas, e adicionar alguns utilitários, como um *log* com o uso da memória.

4.2 Modelos utilizados

Para realizar os experimentos, os modelos **MobileFaceNet**, **Rafael-2** e **MobileNetV3Small** foram utilizados como teste, tanto na etapa de treinamento (seção 4.3) quanto de avaliação (seção 4.4).

Para treinar os modelos, foi utilizada a plataforma kaggle[2], o *dataset* utilizado foi *Labeled*

[1] Disponível em: <https://github.com/espressif/esp-tflite-micro/tree/master> Acessado em Março de 2025
[2] Disponível em: <https://www.kaggle.com/>. Acessado em Março de 2025

Faces in the Wild (LFW) [3], onde os métodos de treinamento *Triplet Loss* e *Triplet Distillation* (FENG et al., 2020) foram utilizados, pois são eficazes para melhorar a precisão do modelo, no contexto de reconhecimento facial. Além disso os modelos foram treinados considerando com imagens R8G8B8, cada imagem tem seus pixels representados por triplas de 8 bits, contendo o valor das cores vermelho, verde e azul, respectivamente.

Para validar os resultados de cada modelo, foi criado o repositório model-eval[4], para ser utilizado junto com o *dataset* Faces - UFS (SANTANA, 2024). Onde o método de validação consiste em comparar uma imagem com *flip* com todo o *dataset*, e definindo a classe prevista pelo modelo aquela que possui menor distância de cosseno.

MobileFaceNet é um modelo focado para fazer reconhecimento de faces em tempo real em dispositivos móveis, assim possuindo uma baixa pegada de memória, abaixo de 5MB e mantendo uma acurácia alta, acima de 90% (SANTANA, 2024).

Rafael-2 é uma variação do modelo Rafael (SILVA, 2022), adaptada para realizar a tarefa de reconhecimento facial. Como o modelo base é simples e focado para dispositivos embarcados, ele possuí uma baixa quantidade de parâmetros, o que melhora a performance do modelo em dispositivos embarcados.

MobileNetV3Small é uma variação do MobileNetV3, com uma quantidade reduzida de parâmetros, reduzindo a sua pegada de memória e aumentando a performance.

4.3 Método de treinamento do modelo

Para fazer o treinamento dos modelos, foi necessário utilizar a técnica de *Triplet Distillation* (FENG et al., 2020), para que seja possível medir o quão próximo as *embeddings* de uma imagem são parecidas com as de outra. Essa técnica usa três imagens: **âncora**, que é serve como imagem base; **positiva**, que é uma variação da mesma categoria da **âncora**; e a **negativa**, que pertence a outra categoria.

No contexto de reconhecimento facial, os *embeddings* das imagens **âncora** e **positivas** devem possuir uma distância de cosseno pequena, enquanto o vetor de característica das imagens **âncoras** e **negativas** devem possuir uma distância de cosseno alta.

4.3.1 *Triplet Loss*

Para realizar o treinamento do modelo MobileFaceNet, foi utilizada a técnica *triplet loss* (FENG et al., 2020). Ela tem como objetivo comparar os *embeddings* de três imagens, utilizando a distância de cosseno (D) de um embedding (x_i^j) comparado com outro (x_i^k).

[3] Disponível em: <https://www.kaggle.com/datasets/luhtookyaw/lfwpreparedtriplets>. Acessado em Março de 2025.

[4] Disponível em: <https://github.com/LuanFabricio/model-eval>. Acessado em Março 2025.

$$Loss = \frac{1}{N}\sum_{i}^{N} max(D(x_i^a, x_i^p) - D(x_i^a, x_i^n) + m, 0)) \tag{4.1}$$

Na equação 4.1, é a função *Loss* utilizada para treinar o modelo. Onde, x_i^a é a imagem âncora, x_i^p é a imagem positiva e x_i^n é a imagem negativa, todas referentes a i-ésima tripla e m é um hiperparâmetro que define a margem entre o par positivo e par negativo. Então, quanto mais parecidos forem os *embeddings* da imagem âncora com a imagem positiva, menor será a perda, sendo o inverso para a âncora com a imagem negativa.

4.3.2 *Triplet Distillation*

Para realizar o treinamento dos modelos MobileNetV3 e Rafael-2, foi utilizada a técnica de *Knowledge Distillation* (HINTON; VINYALS; DEAN, 2015) com *Triplet Loss*, conhecida como *triplet loss* (FENG et al., 2020). Ela utiliza cálculo da *Loss Function* (4.1) como base, adicionando a distância entre os *embeddings* do modelo estudante e do modelo professor, como pode ser visto na fórmula 4.2.

$$Loss = \frac{1}{N}\sum_{i}^{N} max(D(x_i^a, x_i^p) - D(x_i^a, x_i^n) + d, 0)) \tag{4.2}$$

$$d = max(T(x_i^a, x_i^n) - T(x_i^a, x_i^p), 0) \tag{4.3}$$

Onde T (4.3) é a distância entre os *embeddings* do modelo estudante e professor, e d (4.2) é utilizado como uma margem dinâmica entre par positivo e negativo.

4.4 Método de avaliação

Para avaliar o modelo, primeiro, as características de cada imagem e sua versão espelhada (*flip* horizontal) são extraídas pelo modelo. Depois é realizada a verificação da face, calculando a distância de cosseno entre os vetores de características (FENG et al., 2020), onde a imagem com menor distância é considerada a previsão do modelo. A Figura 9 é um exemplo da imagem original e sua versão com *flip*.

4.5 *Datasets*

Nessa seção serão apresentados os dois *datasets* utilizados, *Labeled Faces in the Wild* (LFW), que foi utilizado para o treinamento dos modelos, e Faces - UFS, que foi utilizado para a validação dos modelos.

Figura 9 – Exemplo de uma imagem e sua versão com *flip*

Fonte: LFW

4.5.1 *Labeled Faces in the Wild*

Este *dataset* possui imagens da face de várias celebridades, classificadas com o nome da pessoa. Para esse trabalho, foi utilizada uma variação do LFW, que agrupa as imagens em triplas, **âncora**, **positiva** e **negativa**, para ser utilizado como dataset de treinamento, seguindo a técnica de *triplet distillation* (FENG et al., 2020).

Para realizar o treinamento do modelo, foi utilizada uma variação do *dataset Labeled faces in the Wild* (LFW), que possui várias amostras contendo triplas com as imagens das faces, assim facilitando o uso da técnica de *triplet distillation* para o treinamento.

4.5.2 Faces - UFS

Este *dataset* possui faces coletadas de alunos da Universidade Federal de Sergipe (UFS) (SANTANA, 2024). Para coletar essas imagens, foi disponibilizado um estande com uma câmera, monitor, teclado e mouse, permitindo que qualquer pessoa pudesse salvar uma imagem com o seu nome.

Esse *dataset* foi utilizado na etapa de validação do modelo (4.4), por apresentar faces novas e com um padrão diferente do LFW, já que o seu público consiste em celebridades, principalmente dos Estados Unidos. Enquanto o *dataset* coletado na UFS possui um público diferente, estudantes ou funcionários da UFS.

4.6 Resultados

Para obter os resultados, primeiro foram definidas as métricas que serão usadas para avaliar os modelos. Em seguida, cada modelo foi submetido ao método de avaliação escolhido (seção 4.4), com o objetivo de validar e comparar o desempenho na tarefa de reconhecimento facial de cada um, junto com o seu tempo de execução.

Para realizar a avaliação na tarefa de reconhecimento facial, os modelos foram executados em um computador com um AMD Ryzen 5 4600G e em um Raspberry Model B. Na etapa de avaliação de performance em microcontroladores, foi utilizado o ESP32-S3 N16R8, descrito na seção 4.1.

4.6.1 Métricas usadas

Para avaliar o desempenho, foram utilizadas as métricas de precisão (4.6.1) e acurácia (4.6.1). Como a avaliação é binária, a imagem prevista é a imagem verdadeira, também foi utilizada a matriz de confusão, para facilitar o entendimento dos acertos e erros, quanto facilitar o cálculo da precisão e acurácia.

A precisão pode ser calculada pela Equação 4.4 e a acurácia pela Equação 4.5. Onde a variável TP indica o valor que foi previsto corretamente como verdadeiro, enquanto a variável FP indica o valor que foi previsto falsamente como verdadeiro. Considerando o método de avaliação utilizada, a variável TP indica os casos onde o modelo escolheu corretamente a imagem, e a variável FP indica quando o modelo escolheu a imagem errada.

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{4.4}$$

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.5}$$

4.6.2 Avaliação do modelo

Para a avaliação do modelo, o método descrito na seção 4.4 foi executado 10 vezes, utilizando o *dataset* Faces - UFS (4.5.2), medindo o tempo médio de inferência, desvio padrão e acurácia de cada execução. Com isso, esses resultados, foi gerada a Tabela 2, onde as métricas são dos modelos descritos na seção 4.2.

Tabela 2 – Acurácia e tempo de inferência com o dataset Faces - UFS (Desktop)

Modelo	**Quantização**	**Acurácia (%)**	**Tempo médio de inferência (ms)**	**Método de treinamento**
MobileFaceNet	-	100.00	129 ± 2.002	Pré-treinado + *Triplet Loss*
MobileFaceNet	uint8	100.00	443 ± 2.734	Pré-treinado + *Triplet Loss*
Rafael-2	-	3.25	115 ± 0.553	*Triplet Distillation*
Rafael-2	uint8	24.03	103 ± 0.209	*Triplet Distillation*
MobileNetV3Small	-	7.14	37 ± 0.701	*Triplet Distillation*

Fonte: Autor

A Tabela 2 contém os resultados dos testes feitos com os modelos MobileFaceNet, Rafael-2 e MobileNetV3Small, mostrando o tipo de quantização que foi feito, acurácia, tempo médio de inferência e o método de treinamento.

Com isso, é possível notar que o modelo que o melhor modelo foi o MobileFaceNet, por conta da sua estrutura mais robusta que tem o foco na tarefa de reconhecimento facial. Além disso, fica evidente a diferença entre o tempo de inferência do MobileFaceNet quantizado e não quantizado, isso se deve pelo ambiente de execução do TensorFlowLite, que em tempo de execução converte os pesos quantizados de inteiro de 8 bits para float.

Tabela 3 – Acurácia e tempo de inferência com o dataset Faces - UFS (Raspberry Pi Model B)

Modelo	**Quantização**	**Acurácia (%)**	**Tempo médio de inferência (ms)**	**Método de treinamento**
MobileFaceNet	-	100.00	3677 ± 13.218	Pré-treinado + *Triplet Loss*
MobileFaceNet	uint8	99.35	10607 ± 28.274	Pré-treinado + *Triplet Loss*
Rafael-2	-	3.25	1779 ± 10.346	*Triplet Distillation*
Rafael-2	uint8	24.03	2515 ± 10.8334	*Triplet Distillation*

Fonte: Autor

A Tabela 3 contém os resultados dos experimentos feitos no Raspberry Pi Model B, utilizando o *dataset* Faces - UFS e o interpretador do tflite-runtime [5] no lugar do interpretador do tensorflow, por ser a versão indicada para executar modelos de aprendizagem de máquina em dispositivos embarcados, móveis ou com baixo poder computacional. Por conta de uma limitação do tflite-runtime, o modelo MobileNetV3Small não foi incluído nos testes.

Com isso, é possível observar que os resultados são parecidos com o experimento no Desktop, o MobileFaceNet permanece com acurácia alta, com e sem quantização. A diferença no tempo de execução entre os tipos de quantização também é parecida, por conta da forma como o *runtime* lida com a quantização do modelo.

Tabela 4 – Acurácia e tempo de inferência Faces - UFS (ESP)

Modelo	**Quantização**	**Acurácia (%)**	**Tempo médio de inferência (ms)**	***Runs***
MobileFaceNet	uint8	93.33	3703 ± 0.6010	5
MobileFaceNet	uint8	93.33	3703 ± 0.6067	10
Rafael-2	uint8	93.33	737 ± 0.061	5
Rafael-2	uint8	93.33	737 ± 0.056	10

Fonte: Autor

A Tabela 4 mostra o resultado dos testes feitos no ESP32-S3, utilizando a ferramenta idf.py[6] com a biblioteca Tensorflow Lite Micro[7], e com uma versão com uma versão limitada do *dataset* Faces - UFS, contendo apenas três imagens. O foco desse experimento foi o medir o tempo de inferência dos modelos e avaliar a sua acurácia no ESP32-S3, onde foi possível perceber os limites do do microcontrolador.

[5] Disponível em: <https://pypi.org/project/tflite-runtime/>. Acessado em Maio 2025.
[6] Disponível em: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/tools/idf-py.html>. Acessado em Maio 2025.
[7] Disponível em: <https://github.com/espressif/esp-tflite-micro>. Acessado em Maio 2025.

Com isso, é possível observar que acurácia permanece igual, isso é causado pela limitação na quantidade de imagens utilizada, visto que a memória do ESP32-S3 é limitada, então não possível portar todo *dataset* utilizado no experimento da tabela Tabela 2.

5
Conclusão

Este trabalho teve como objetivo principal a compressão de uma rede neural artificial para viabilizar o reconhecimento facial em dispositivos embarcados, buscando manter uma alta acurácia e baixa latência no dispositivo, assim o tornado viável o seu uso em tarefas que executam em tempo real.

Para alcançar esse objetivo, foi realizada a compressão, treinamento e avaliação de modelos de reconhecimento facial, onde foi desenvolvidos *benchmarks* para medir a acurácia e latência dos modelos, para dispositivos de propósito geral e embarcados. Os dispositivos utilizados foram *Desktop*, Raspberry Model B e ESP32-S3, junto com os interpretadores TensorFlow, TFLite Runtime e ESP TFLite Micro.

Como foi possível observar nos experimentos, o melhor modelo para realizar o trabalho de reconhecimento facial é o MobileFaceNet com quantização para o tipo uint8, pois ele apresenta um acurácia alta, mesmo possuindo um tempo de resposta elevado. Considerando isso, esse modelo não é o mais indicado para realizar reconhecimento facial em tempo real, pois ele demora 3.7 segundos gerando os *embeddings* de uma imagem, ou seja, ele é muito lento para ser executado em tempo real, visto que o ideal seria alcançar uma média de 33.33 milissegundos, para alcançar 30 *frames* por segundo.

Inicialmente, esperava-se que fosse possível realizar compressão ao ponto que o modelo MobileFaceNet executasse a tarefa de reconhecimento facial com baixa latência, ou que seria possível aplicar *Triplet Distillation* no modelo Rafael-2 para melhorar a sua acurácia consideravelmente, a tornando próxima do MobileFaceNet. Porém, como é possível observar, esse resultado não foi alcançado, uma vez que o único modelo com acurácia alta foi o MobileFaceNet, que possui um tempo de inferência elevado no ESP32-S3, o tornando inviável para a executá-lo em tempo real.

Considerando isso, sugere-se que, como trabalhos futuros, seja feita um estudo utilizando diferentes hardwares, com o objetivo de aumentar a amostra de dispositivos testados e encontrar

um dispositivo com baixo custo que seja capaz de realizar a tarefa de reconhecimento facial em tempo real. Outra sugestão seria utilizar *benchmark* construído para o ESP32-S3 como ponto de partida para desenvolver um produto ou demonstração funcional que integre a detecção e reconhecimento facial em aplicações práticas.

Referências

BAGCHI, T. et al. Intelligent security system based on face recognition and iot. *Materials Today: Proceedings*, v. 62, p. 2133–2137, 2022. ISSN 2214-7853. International Conference on Design, Manufacturing and Materials Engineering. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214785322016984>. Citado na página 25.

FACELI, K. et al. *Inteligência artificial: uma abordagem de aprendizado de máquina*. [S.l.]: LTC, 2011. Citado 2 vezes nas páginas 18 e 19.

FENG, Y. et al. 2020. Disponível em: <https://arxiv.org/abs/1905.04457>. Citado 3 vezes nas páginas 28, 29 e 30.

FRAZIER, P. I. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. Citado na página 23.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 19.

HINTON, G.; VINYALS, O.; DEAN, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. Citado 3 vezes nas páginas 14, 23 e 29.

KADHIM, T. A. et al. A face recognition application for alzheimer's patients using esp32-cam and raspberry pi. 2023. Disponível em: <https://rdcu.be/d3MWS>. Citado 2 vezes nas páginas 25 e 26.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. v. 25. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>. Citado na página 14.

LIANG, T. et al. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, v. 461, p. 370–403, 2021. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231221010894>. Citado 2 vezes nas páginas 22 e 23.

SANTANA, L. de J. D. *Edge FR - Desenvolvimento de um modelo de reconhecimento facial para dispositivos embarcados de baixo custo*. 2024. Citado 2 vezes nas páginas 28 e 30.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. *Journal of big data*, Springer, v. 6, n. 1, p. 1–48, 2019. Citado na página 21.

SILVA, R. A. da. A resource constrained pipeline approach to embed convolutional neural models (cnns). 2022. Disponível em: <https://drive.google.com/file/d/1yl0EMe8q0iasmqoZpCXEV9L5UO6bgfF4/view>. Citado 2 vezes nas páginas 24 e 28.

TENSORFLOW. *Data augmentation | TensorFlow Core*. 2024. Disponível em: <https://www.tensorflow.org/tutorials/images/data_augmentation>. Citado na página 22.

ZHANG, A. et al. *Dive into Deep Learning*. [S.l.]: Cambridge University Press, 2023. <https://D2L.ai>. Citado 2 vezes nas páginas 20 e 21.