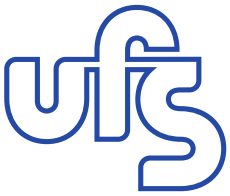


UNIVERSIDADE FEDERAL DE SERGIPE

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

2024

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

2024

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*

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.

Redes Neurais Artificiais, ou *Artificial Neural Network* (ANN), são ferramentas poderosas para auxiliar a sociedade, principalmente em tarefas de classificação ou reconhecimento facial.

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.

As etapas 3, 4 e 5 serão realizadas no Trabalho de Conclusão de Curso 2.

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

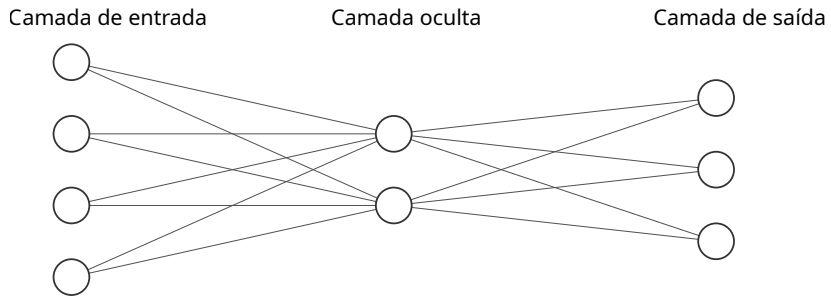
Conceitos básicos

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

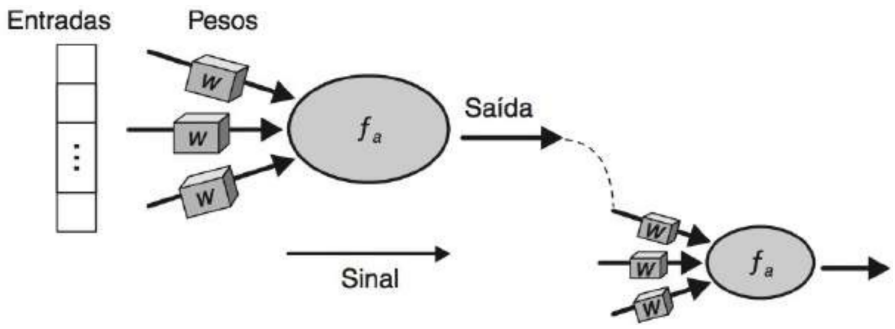
O neurônio é uma parte fundamental de uma ANN, nele que o aprendizado é armazenado através do reforço de conexões com outros neurônios. Esse reforço é o peso da conexão, ele é multiplicado pela entrada e somado com os outros valores, como é demonstrado na equação

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

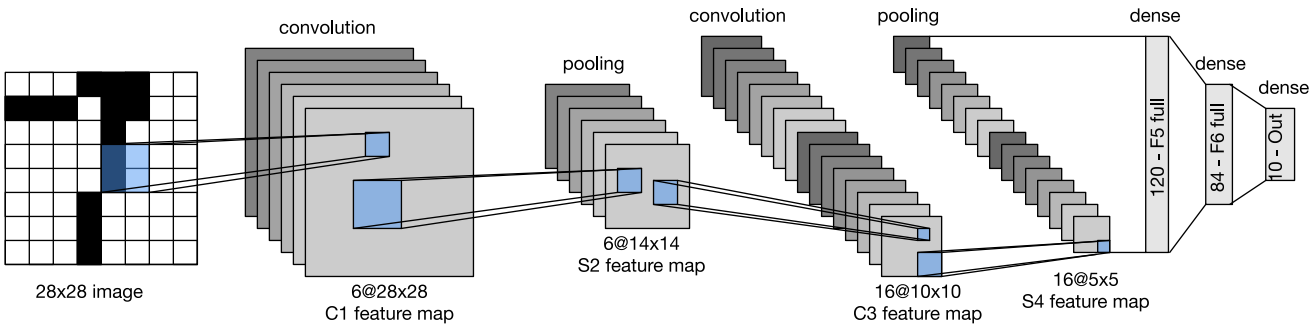
Redes Neurais Convolucionais são Redes Neurais Artificiais que utilizam a operação de convolução para o processamento e análise de dados no formato de *grid* (grade). Por exemplo, uma imagem, que pode ser representada no formato 2-D (GOODFELLOW; BENGIO; COURVILLE, 2016).

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*

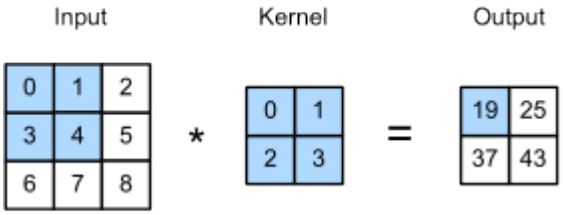
Figura 3 – Arquitetura da LeNet



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



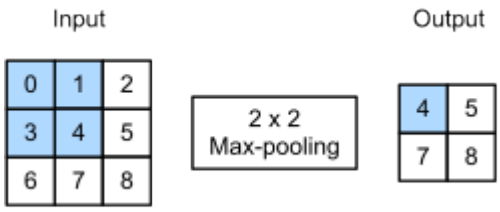
Fonte: Zhang et al. (2023)

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.

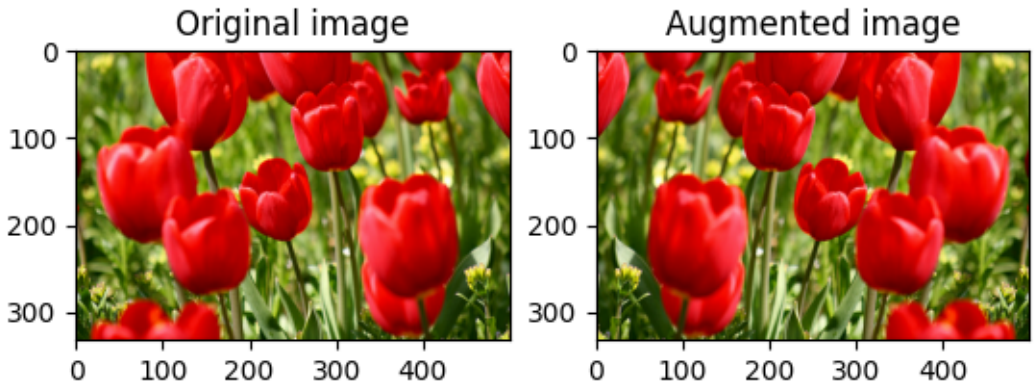
Na Figura 6, é apresentado um exemplo de uma imagem que sofreu uma transformação para efetuar um *data augmentation*. Nesse caso, uma que foi revertida (*flip*), gerando um novo dado para o dataset

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

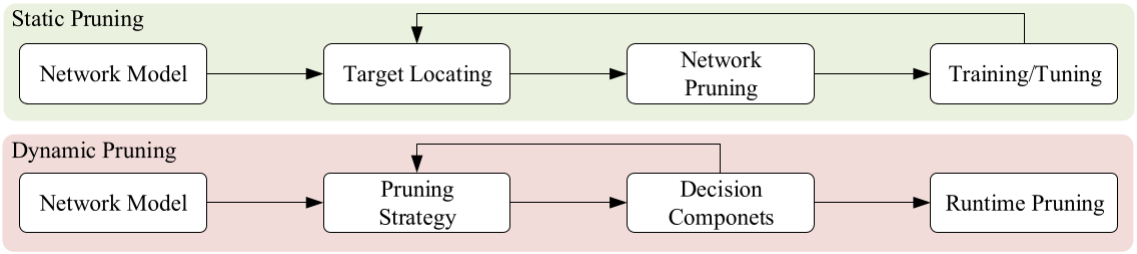
ANN são utilizadas em várias aplicações, demonstrando habilidades no campo de visão computacional. No entanto, redes com arquiteturas complexas são um desafio para a implantação em tempo real e necessitam de uma grande quantidade de energia e poder computacional (LIANG et al., 2021). Por causa disso 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:

- Portal de periódicos utilizada foi a CAPES CAFE.
- A string de busca foi a seguinte: "face recognition"and "esp32".
- O seguintes filtros foram utilizados:
  - Idioma: Inglês.
  - Revisado por pares: Sim.

—

- 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 *IMPROVED FACE DETECTION ACCURACY USING HAAR CASCADE CLASSIFIER METHOD AND ESP32-CAM FOR IOT-BASED HOME DOOR SECURITY*

4

Resultados preliminares

Neste capítulo serão apresentados testes feitos durante o período inicial do TCC, eles tiveram a finalidade de aplicar ténicas para ter modelos menores. O objetivo principal é aplicar as técnicas de compressão para criar modelos menores e computacionalmente eficientes.

4.1 Destilação de conhecimento (modelo Professor-Aluno)

Para fazer o experimento com destilação de conhecimento foi utilizada a base STL-10 (COATES HONGLAK LEE, 2011), que possui 500 imagens para treinamento e 800 para teste, com resolução de 96×96 e 3 canais de cor (RGB). Como o conjunto de dados não possui muitas imagens, foi aplicada a técnica de *data augmentation* (aumento de dados) para reduzir o *overfitting*.

Como já descrito na subseção 2.5.3, o objetivo dessa etapa é utilizar o conhecimento do modelo Professor (mais robusto e pré-treinado) para treinar o modelo Aluno (mais simples e sem pré-treinamento). O modelo professor (Código 1) é a ResNet-50 (HE et al., 2016) e o modelo estudante é gerado pelo Código 2. Além disso, o modelo Rafael (SILVA, 2022) foi adaptado e utilizado (com algumas variações).

Para aumentar a precisão do modelo Aluno com o destilação de conhecimento, foi utilizada a otimização Bayesiana, para procurar os valores dos hiperparâmetros α e *Temperature*. Os possíveis valores de α foram 0.1, 0.5, 0.01 e 0.25. E os possíveis valores de *Temperature* foram 2, 5, 7, 10, 12, 15, 17 e 20. Os resultados do experimento estão na Tabela 1.

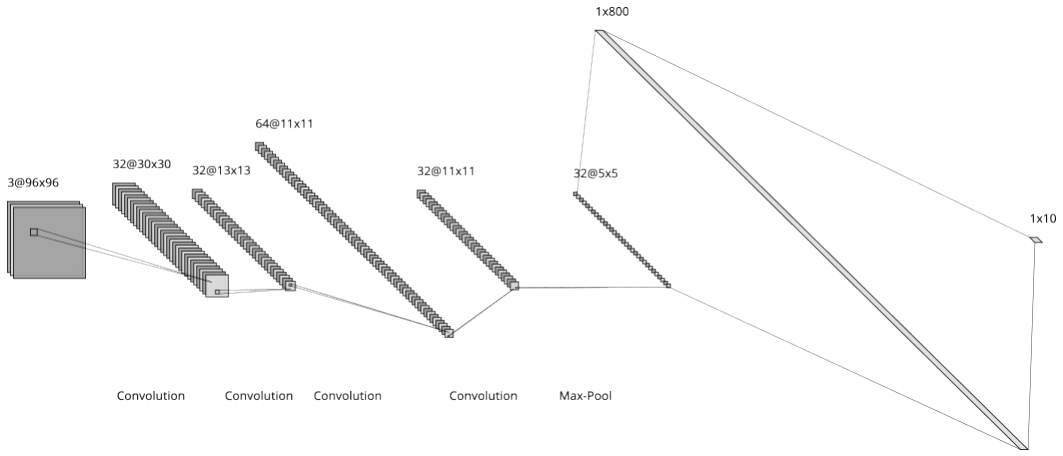
A Tabela 1 apresenta a acurácia dos modelos Rafael-Base, Rafael-1, Rafael-2 e Rafael-3 na tarefa de classificação na base STL-10. Onde a coluna **Com destilação de conhecimento?**

Tabela 1 – Acurácia dos modelos.

Modelo	**Com destilação de conhecimento?**	**Acurácia (validação)**	α	***Temperature***
Rafael-base	Não	71.38%	-	-
Rafael-1	Não	67.08%	-	-
Rafael-2	Não	74.57%	-	-
Rafael-3	Não	71.01%	-	-
Rafael-base	Sim	74.21%	0.1	10
Rafael-1	Sim	69.70%	0.01	20
Rafael-2	Sim	79.12%	0.01	7
Rafael-3	Sim	76.55%	0.01	5

Fonte: Autor

Figura 8 – Arquitetura do modelo Rafael-1



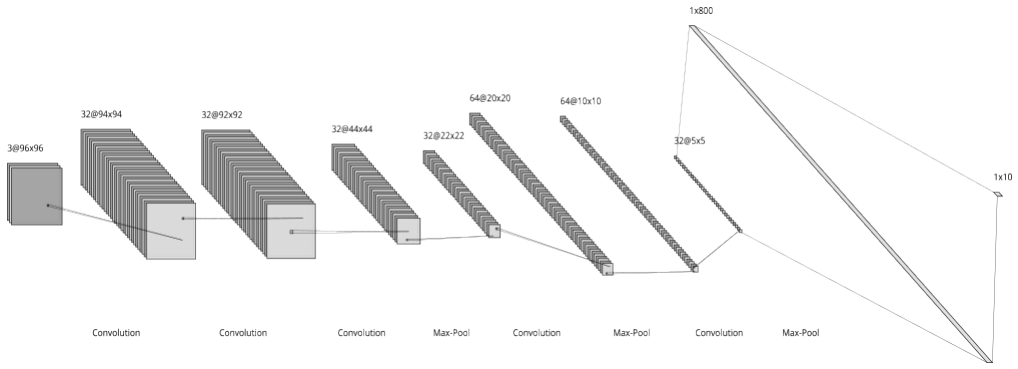
Fonte: Autor

indica se a técnica de destilação foi aplicada ou não, e as colunas α e *Temperature* indicam os valores dos hiperparâmetros utilizados. Percebe-se que, em todos os casos, os modelos que foram treinados utilizando a técnica de destilação de conhecimento alcançaram uma acurácia maior do que os modelos treinados apenas com os atributos alvo, chegando a um aumento maior que 5% (no modelo Rafael-3). Nela, as variações indicam a quantidade de camadas convolucionais adicionadas, Rafael-1 (Figura 8) indica que foi adicionada uma camada convolucional $9x9$ no modelo, Rafael-2 (Figura 9) indica que foi adicionada duas camadas $3x3$ convolucionais e Rafael-3 (Figura 10)indica que foi adicionada três camadas $3x3$.

4.2 *Pruning* e Quantização

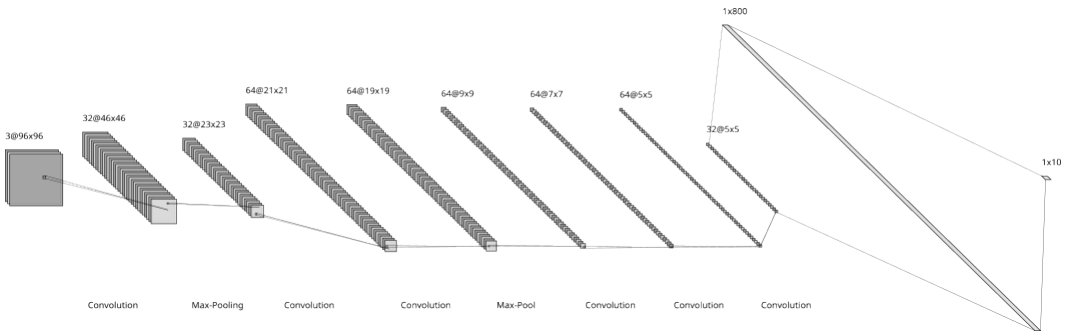
Para esse experimento foi utilizada a base CIFAR-10, pois ela exige menos do modelo, facilitando o treinamento de uma CNN que performe bem nessa tarefa de classificação, o que permite que o foco da atividade seja mantido. Essa base possui 10 classes, com 6.000 imagens para cada classe, sendo que essas imagens têm resolução igual a $32x32$ e possui 3 canais de cores (RGB). Nesse conjunto de dados também foi necessário fazer *data augmentation* para melhorar a

Figura 9 – Arquitetura do modelo Rafael-2



Fonte: Autor

Figura 10 – Arquitetura do modelo Rafael-3



Fonte: Autor

acurácia do modelo final.

O modelo utilizado no experimento foi gerado pelo Código 3, inicialmente ele possuía 2.397.226 parâmetros (28 MB). Inicialmente ele foi treinado durante 25 *epochs* (épocas), atingindo uma acurácia de 90.18% nos dados de treinamento e 85.91% nos dados de validação. Onde somente os dados de validação serão utilizados para comparar os modelos, pois eles podem sofrer *overfitting* e ter um alto desempenho nos dados de treinamento e baixo desempenho nos dados de validação.

Depois de treinado, o modelo foi podado utilizando a estratégia de *prune low magnitude* (podar baixa magnitude), que tem como foco zerar valores abaixo de um certo limiar. Depois de definir os parâmetros da poda, o modelo foi retreinado por 2 *epochs*, para que o algoritmo de poda consiga identificar as conexões importantes durante o uso do modelo. Após o retreinamento, o modelo final teve acurácia de 83.83% nos dados de treinamento e 84.40% nos dados de validação, consumindo 9.3 MB após remover todos os valores iguais a zero.

Depois de podado, o modelo foi convertido para TensorFlow Lite, consumindo 9.2 MB de armazenamento e ficando com 84.91% de acurácia. Depois disso, a quantização é aplicada

utilizando a API do TensorFlow Lite, deixando o modelo final com 2.4 MB e 84.36% de acurácia.

Tabela 2 – Acurácia e peso dos modelos.

Modelo	**Acurácia (validação)**	***Memory footprint* (MB)**
Modelo base	85.91%	28
Modelo base podado	84.40%	9.3
Modelo base podado (TFLite)	84.91%	9.2
Modelo base podado e quantizado (TFLite)	84.36%	2.4

Fonte: Autor

A Tabela 2 mostra a acurácia dos modelos (nos dados de validação) durante as etapas de poda e quantização, onde a coluna ***Memory footprint*** indica o consumo de armazenamento do modelo. A poda e quantização do modelo base resultou em uma redução de pegada de memória de 28 MB para 2.4 MB, apresentando uma perda de 1.5% da sua acurácia, tornando o modelo mais eficiente sem a perda de desempenho.

5 Planos de continuidade

No Trabalho de Conclusão de Curso 1, foram realizadas as etapas de aprendizado e resultados de experimentos relacionados ao tema.

Tabela 3 – Cronograma das atividades.

Atividade	Maio	Jun.	Jul.	Ago.	Set.
Busca de soluções existentes	X				
Escolha de um modelo base	X				
Aplicação de técnicas de compressão e execução de experimentos		X	X		
Proposição de uma solução e teste				X	X

Fonte: Autor

Na primeira etapa, será realizada a busca de modelos de reconhecimento facial, com o foco em sistemas embarcados, onde o ponto de partida será o trabalho de Leandro (SANTANA, 2024). Depois disso, esses modelos serão comparados, tendo como principais métricas acurácia e pegada de memória. Com o modelo base escolhido, serão utilizadas técnicas de compressão, com o objetivo de produzir um modelo que seja computacionalmente barato e consuma pouca memória. Para finalizar, o modelo final será testado em diversos dispositivos embarcados, onde o objetivo principal é que ele seja executado em dispositivos baratos, como pouco poder computacional e memória disponível.

Vale ressaltar que, o cronograma é uma expectativa das atividades que serão realizadas durante o Trabalho de Conclusão de Curso 2, algumas atividades podem ser prolongadas, reduzidas, adicionadas ou removidas.

Referências

COATES HONGLAK LEE, A. Y. N. A. *An Analysis of Single Layer Networks in Unsupervised Feature Learning*. 2011. Disponível em: <http://cs.stanford.edu/~acoates/stl10>. Citado na página 26.

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

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.

HE, K. et al. Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 770–778. Citado 2 vezes nas páginas 26 e 33.

HINTON, G.; VINYALS, O.; DEAN, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. Citado 2 vezes nas páginas 14 e 23.

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 na página 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 25 e 26.

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.

Apêndices

APÊNDICE A – Modelo Professor

Código utilizado para criar a o modelo Professor usando a ResNet-50 (HE et al., 2016) com TensorFlow 2.0.

Código 1 – Criação do modelo Professor

```
preprocess_input = tf.keras.applications.resnet50.preprocess_input
base_model =
   tf.keras.applications.resnet.ResNet50(input_shape=IMG_SHAPE,
           include_top=False,
           pooling='avg',
           weights='imagenet')
base_model.trainable = False
input = tf.keras.Input(shape=(96, 96, 3))
x = input
x = preprocess_input(x)
x = base_model(x, training=False)
x = tf.keras.layers.Dropout(0.2)(x)
output = tf.keras.layers.Dense(n)(x)
teacher = tf.keras.Model(input, output)
```

Fonte: Autor

APÊNDICE B – Modelo Aluno

Código utilizado para criar a o modelo Aluno com TensorFlow 2.0.

Código 2 – Criação do modelo Aluno

```
def create_student_model():
  i = tf.keras.layers.Input(shape=IMG_SHAPE)
  x = add_cnorm_layer(32, i)
  x = add_cnorm_layer(64, x)
  x = add_cnorm_layer(128, x)
  x = tf.keras.layers.Flatten()(x)
  x = tf.keras.layers.Dropout(0.2)(x)
  x = tf.keras.layers.Dense(1024, activation='relu')(x)
  x = tf.keras.layers.Dropout(0.2)(x)
  x = tf.keras.layers.Dense(n)(x)
  return tf.keras.Model(i, x)

def add_cnorm_layer(size, x):
  x = tf.keras.layers.Conv2D(size, (3, 3), padding='same',
      activation='relu')(x)
  x = tf.keras.layers.BatchNormalization()(x)
  x = tf.keras.layers.Conv2D(size, (3, 3), padding='same',
      activation='relu')(x)
  x = tf.keras.layers.BatchNormalization()(x)
  x = tf.keras.layers.MaxPooling2D((2, 2))(x)
  return x
```

Fonte: Autor

APÊNDICE C – Modelo utilizado para fazer poda e quantização

Código 3 – Criação do modelo utilizado na etapa de poda e quantização

```
def create_model():
  i = Input(shape=x_train[0].shape)

  x = add_cnorm_layer(32, i)
  x = add_cnorm_layer(64, x)
  x = add_cnorm_layer(128, x)

  x = Flatten()(x)
  x = Dropout(0.2)(x)
  x = Dense(1024, activation='relu')(x)
  x = Dropout(0.2)(x)
  x = Dense(K, activation='softmax')(x)

  return Model(i, x)

def add_cnorm_layer(size, x):
  x = Conv2D(size, (3, 3), padding='same', activation='relu')(x)
  x = BatchNormalization()(x)

  x = Conv2D(size, (3, 3), padding='same', activation='relu')(x)
  x = BatchNormalization()(x)

  x = MaxPooling2D((2, 2))(x)

  return x
```

Fonte: Autor

Anexos

ANEXO A – Tabelas com os preços e MCUs listados

Tabela 4 – Acurácia dos modelos.

CPU (Clock Speed)	RAM	Preço	Link
Quad-core Cortex-A53 (2.30GHz)	1GB	$129.99	https://www.seeedstudio.com/Coral-Dev-Board-p-2900.
Quad-core ARM A57 (1.43GHz)	4GB	$89.00	https://www.seeedstudio.com/NVIDIAr-Jetson-Nanotm-Developer-
RISC-V Dual Core (400Mhz)	8MB	$40.90	https://www.seeedstudio.com/Sipeed-MAix-GO-Suit-for-RISC-V-AI
Cortex-A72 - ARM v8 (1.5GHz)	1GB	$35.90	https://www.seeedstudio.com/Raspberry-Pi-4-Computer-Model-B-1

Fonte: Autor