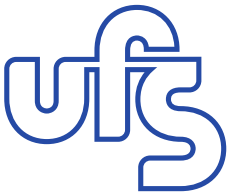


UNIVERSIDADE FEDERAL DE SERGIPE

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. Para finalizar, o modelo será embarcado para realizar operações na borda, tanto de inferência quanto de processamento do dispositivo, a partir disso a sua eficácia será medida e comparada com outras soluções já criadas.

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. To finish, this model will be embedded to perform operations on edge, both inference and processing on the device, from there, its effectiveness will be measured and compared with other solutions already created.

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 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, de acordo com o seu conjunto de dados. Permitindo que ela seja usada em várias áreas diferentes, indo de entreterimento 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 começaram a ser utilizadas para classificação de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Um dos principais 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 tenha um baixo tempo de resposta, o que pode atrapalhar a experiência do usuário, ou reduz a efetividade da ação que será tomada.

Um dos principais problemas das Redes Neurais 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 destilamento de conhecimento (HINTON; VINYALS; DEAN, 2015), poda e quantização. Possibilitando a implantação do modelo em sistemas embarcados na borda, de forma que o tempo de resposta do dispositivo seja baixo.

1.2 Objetivo principal

O objetivo deste trabalho é desenvolver uma Rede Neural Convolucional que consiga realizar a tarefa de reconhecimento facial em microcontroladores, na borda. De forma que, o reconhecimento facial seja realizado dentro do dispositivo embarcado, permitindo que ele realize essa tarefa e execute uma rotina, com a menor latência possível, por exemplo, desbloqueando uma porta assim que o rosto for reconhecido pelo sistema. Para que ele seja alcançado, será necessário comprimir um modelo base, para que ele exija pouco poder computacional e uso de memória.

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 seja atingido. Sendo elas:

1. **Seleção de artigos:**

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

2. **Seleção de base e treino de modelos:**

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

3. **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 destilamento de conhecimento.

4. **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.

5. **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 tão bem, mesmo após a aplicação de compressão. Nesta etapa as métricas de acurácia e tamanho do modelo são avaliadas.

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

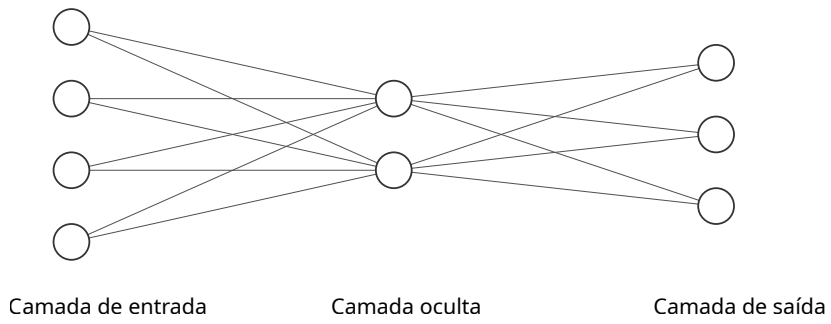
Conceitos básicos

O foco deste capítulo é introduzir alguns tópicos 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 ANNs (seção 2.1), CNNs (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 (ANNs), são neurônios interconectados que realizam um processamento simples. Dentro dessa estrutura cada neurônio reforça ou enfraquece a conexão com um dos neurônio da coluna anterior, assim replicando o processo de aprendizagem do cérebro humano. A Figura 1 exemplifica uma ANN bem simples.

Figura 1 – Exemplo de uma ANN



Fonte: Autor

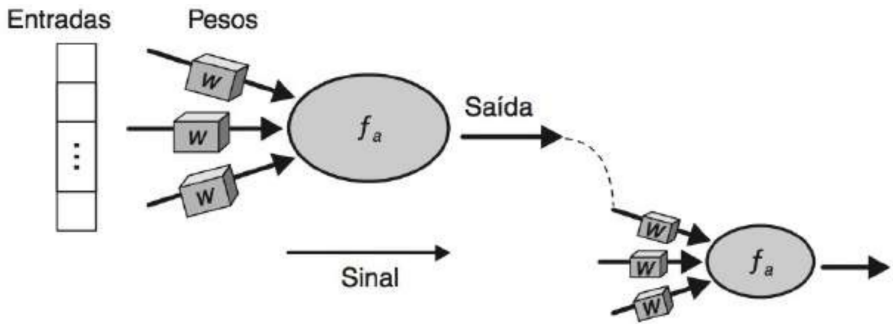
O neurônio é uma parte fundamental de uma ANN, nele que o aprendizado é armazenado através do reforço de conexões com outros neurônios. Esse reforço é o peso da conexão, ele é multiplicado pela entrada e somado com os outros valores, como é demonstrado na equação

2.1 (onde x é um vetor com os valores de entrada do neurônio e w é um vetor com os pesos de cada entrada). Depois disso os valores passam por uma função de ativação $g(x)$ (2.2), que é responsável por "tratar"esses dados de saída antes que eles sejam passados para próxima etapa.

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

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

Figura 2 – Exemplo de um neurônio artificial



Fonte: (FACELI et al., 2011)

2.2 Redes Neurais Convolucionais

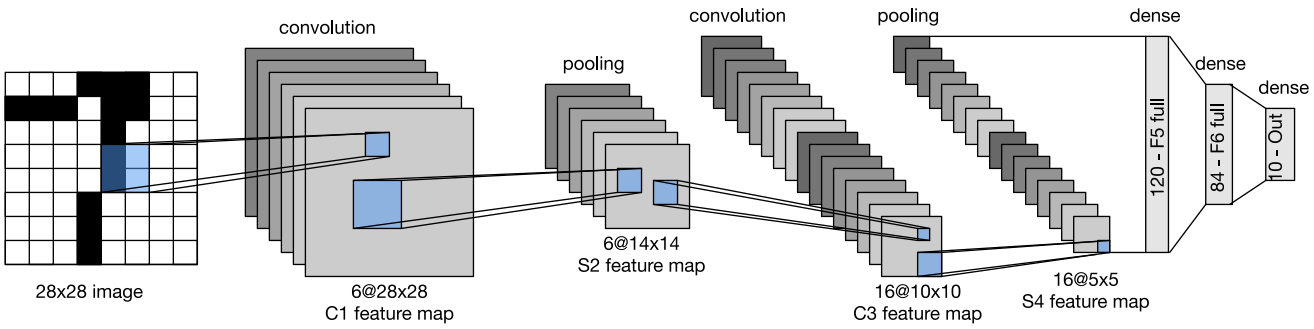
Redes Neurais Convolucionais (CNNs) são Redes Neurais Artificiais (ANN) que utilizam a operação de convolução para o processamento e análise de dados no formato de *grid*(grade). Por exemplo, uma série temporal que pode ser representada no formato de *grid* 1-D, ou uma imagem, que pode ser representada no formato 2-D. (GOODFELLOW; BENGIO; COURVILLE, 2016) Onde, LeNet (LECUN et al., 1995), Residual Network (ResNet) (HE et al., 2016) e AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) são alguns exemplos de CNNs famosas.

A arquitetura de uma CNN é composta por camadas convolucionais (2.2.1), *pooling* (2.2.2) e totalmente conectadas (2.2.3), como podemos ver na Figura 3.

2.2.1 Camada de Convolução

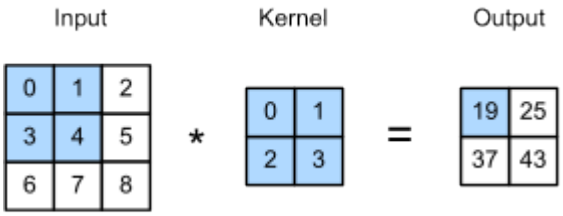
Nessa camada são aplicados filtros (matriz de pesos) nos dados de entrada, onde esses filtros deslizam cada célula da imagem executando operações de multiplicação e soma em cada elemento da matriz de entrada, com o objetivo de gerar um mapa de características (*feature map*). O objetivo desses filtros é realçar as características dos dados de entrada, como curvas, linhas e outros padrões. A Figura 4, é um exemplo da operação de convolução sendo aplicada em uma matriz.

Figura 3 – Arquitetura da LeNet



Fonte: (ZHANG et al., 2023)

Figura 4 – Exemplo de convolução



Fonte: (ZHANG et al., 2023)

2.2.2 Camada de *pooling*

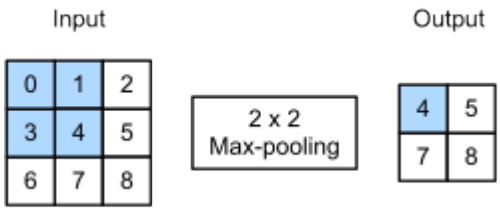
A abordagem da camada de *pooling* é um pouco parecida com a camada de convolução, uma matriz desliza pelas células da imagem, salvando apenas um dos valores dessa área na matriz de saída. A partir disso, a camada consegue reduzir o tamanho da matriz de entrada, fazendo com que o poder computacional 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.

2.2.3 Camada totalmente conectada

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

Figura 5 – Exemplo de max *pooling*



Fonte: (ZHANG et al., 2023)

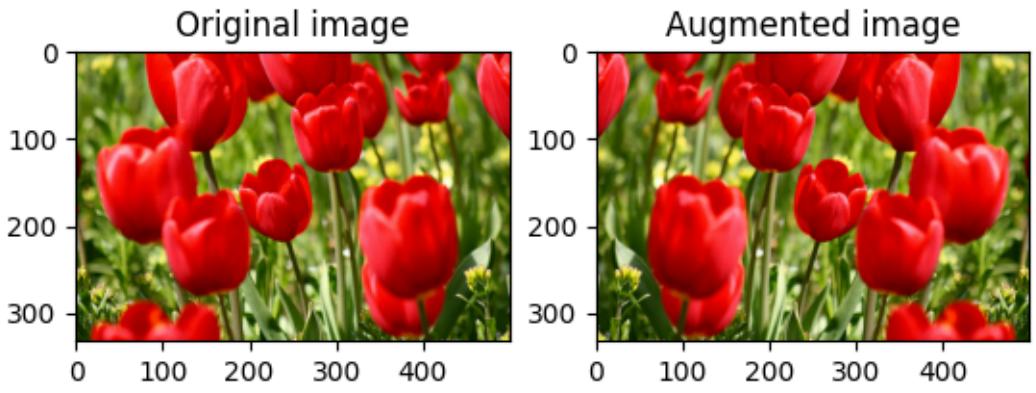
dos dados de entrada.

2.3 *Data augmentation*

CNNs tem um ótimo desempenho em tarefas de visão computacional. Entretanto esse tipo de rede neural precisa de uma grande quantidade dados para não sofrer de *overfitting* (superajuste). (SHORTEN; KHOSHGOFTAAR, 2019) Esse é o objetivo do *data augmentation* (aumento de dados), 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, temos um exemplo de uma imagem que sofreu uma transformação para efetuar um *data augmentation*. Nesse caso, temos uma que foi flipada, gerando um novo dado para o dataset

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



Fonte: (TENSORFLOW, 2024)

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. É necessário que o problema do *dataset* (conjunto de dados) atual seja um subconjunto do *dataset* que foi usado para treinar o modelo base.

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.

2.5 Métodos de compressão para Redes Neurais

ANNs são utilizadas em várias aplicações, demonstrando habilidades extraordinárias 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 eficiente. Nesse trabalho os métodos de poda (2.5.1), quantização (2.5.2) e destilamento do conhecimento (2.5.3) serão usados.

2.5.1 *Pruning*(Poda)

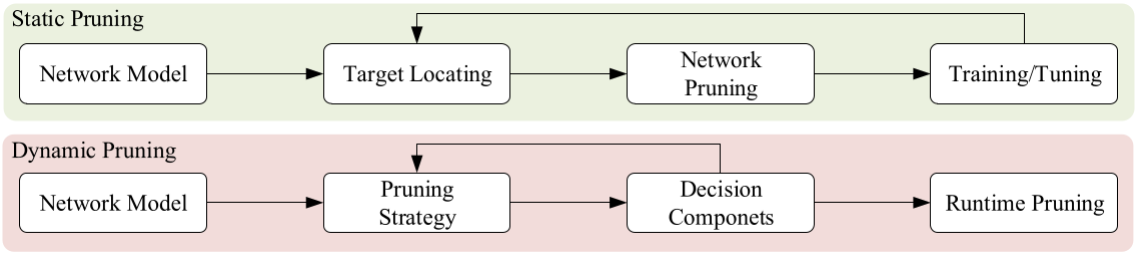
A poda de redes neurais tem como foco eliminar conexões ou neurônios que não apresentam uma grade contribuição para a rede. Essa operação, é muito vantajosa para diminuir a pegada de memória da rede, pois ela reduz a quantidade de parâmetros redundantes ou que não contribuem muito para a precisão dos 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 superajuste da rede, uma vez que remove pesos pouco importantes ou redundantes da rede. Podemos dividir esse processo em dois, poda estática (*static pruning*), que é realiza todas as etapas de poda o modelo sem executar o processo de inferência, e poda dinâmica (*dynamic pruning*), que é realizada junto com o processo de execução do modelo, permitindo que os nós relevantes sejam identificados.

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 bit, embora implementações menores que 8 bit sejam discutidas incluindo redes neurais binárias. (LIANG et al., 2021)

Figura 7 – Fluxo dos tipos de poda



Fonte: (LIANG et al., 2021)

2.5.3 Destilamento de conhecimento (Professor-Aluno)

Destilamento 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). Então, o Aluno é treinado com uma porcentagem α do erro com o *hard target* e $1 - \alpha$ do erro com *soft target*, assim é calculado o erro do aluno.

2.6 Otimização Bayesiana

Testar diferentes valores para os hiperparâmetros é uma tarefa essencial para otimizar o desempenho de ANNs. A otimização Bayesiana é um dos métodos utilizados para fazer esse teste, ela possui dois componentes principais, o modelo estatístico Bayesiano, que modelar a função objetiva, e a função de aquisição, que decide a próxima amostra de parâmetros. (FRAZIER, 2018)

A otimização Bayesiana procura encontrar um valor ótimo (maximizando ou minimizando alguma métrica do modelo). Inicialmente os hiperparâmetros são escolhidos aleatoriamente e testados, após algumas iterações o modelo começa a convergir para um resultado ótimo, em algum ponto essa função de otimização só irá escolher o melhor conjunto de parâmetros testado.

3

Trabalhos relacionados

Neste capítulo serão listados e descritos os trabalhos relacionados à compressão de CNNs 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 CNNs 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 (CNNs)*

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 pelo destilamento 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.

4

Resultados preliminares

Neste capítulo serão apresentados testes feitos durante o trabalho, eles tiveram a finalidade de exercitar o conteúdo estudado. O objetivo principal é usar as técnicas de compressão para criar modelos menores e mais eficientes.

4.1 Destilamento de conhecimento (modelo Professor-Aluno)

Para fazer o experimento com destilamento de conhecimento foi utilizada da base STL-10, 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 foi 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). Onde 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 destilamento 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 3.

Na Tabela 3, a variação Rafael-1 indica que foi adicionada uma camada $9x9$ no modelo, Rafael-2 indica que foi adicionada duas camadas $3x3$ e Rafael-3 indica que foi adicionada três camadas $3x3$.

Tabela 1 – Acurácia dos modelos.

Modelo	**Com destilamento de conhecimento?**	**Acurácia (validação)**	α	***Temperature***
ResNet-50	Não	90,65%	-	-
Aluno	Não	76,80%	-	-
Rafael-1	Não	67,08%	-	-
Rafael-base	Não	71,38%	-	-
Rafael-2	Não	74,57%	-	-
Rafael-3	Não	71,01%	-	-
Aluno	Sim	83,79%	0,1	5
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

4.2 *Pruning* e Quantização

Para esse experimento foi utilizada a base CIFAR-10, que possui 6.000 imagens para cada classe, sendo que essas imagens tem resolução igual $32x32$ e possui 3 canais de cores (RGB). Nesse conjunto de dados também foi necessário fazer *data augmentation* para aumentar a precisão do modelo final.

O modelo utilizado no experimento foi gerado pelo Código 3, inicialmente ele possui 2.397.226 parâmetros (pesando 28 MB). Inicialmente ele é treinado durante 25 *epochs* (épocas), atingindo uma acurácia de $90,18\%$ nos dados de treinamento e $85,91\%$ nos dados de validação.

Depois de treinado, o modelo é 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 é 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 tem acurácia de $83,83\%$ nos dados de treinamento e $84,40\%$ nos dados de validação, pesando 9,3 MB após remover todos os valores iguais a zero.

Depois de podado, modelo foi convertido para TensorFlow Lite, consumindo 9,2 MB de armazenamento e ficando com $84,91\%$ de precisão. Depois disso, a quantização é aplicada utilizando a API do TensorFlow Lite, deixando o modelo final com 2,4 MB e $84,36\%$ de precisão.

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)	84,40%	9,2
Modelo base podado e quantizado (TFLite)	84,36%	2,4

Fonte: Autor

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. 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

FACELI, K. et al. *Inteligência artificial: uma abordagem de aprendizado de máquina*. [S.l.]: LTC, 2011. Citado na página 18.

FRAZIER, P. I. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. Citado na página 22.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 18.

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 3 vezes nas páginas 18, 25 e 30.

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 22.

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 2 vezes nas páginas 14 e 18.

LECUN, Y. et al. Ua m uller, e. s ackinger, p. simard, and v. vapnik. comparison of learning algorithms for handwritten digit recognition. In: *Proceedings ICANN*. [S.l.: s.n.], 1995. v. 95, p. 53–60. Citado na página 18.

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 21 e 22.

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 20.

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 23 e 25.

TENSORFLOW. *Data augmentation | TensorFlow Core*. 2024. Disponível em: <https://www.tensorflow.org/tutorials/images/data_augmentation>. Citado na página 20.

ZHANG, A. et al. *Dive into Deep Learning*. [S.l.]: Cambridge University Press, 2023. <https://D2L.ai>. Citado 2 vezes nas páginas 19 e 20.

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