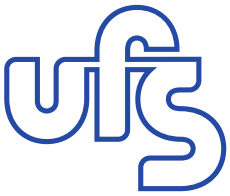


UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

AtalaIA: Compressão de modelos de detecção de velocidade de veículos

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 de velocidade de veículos

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

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

Agradecimentos

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Este trabalho, além de cultural, filosófico e pedagógico
É também medicinal, preventivo e curativo
Servindo entre outras coisas para pano branco e pano preto
Curuba e ferida braba
Piolho, chulé e caspa
Cravo, espinha e berruga
Panarismo e água na pleura
Só não cura o velho chifre
Por que não mata a raiz
Pois fica ela encravada
No fundo do coração
(Falcão)

Resumo

Segundo a ABNT (2003, 3.1-3.2), o resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Palavras-chave: latex. abntex. editoração de texto.

Abstract

This is the english abstract.

Keywords: latex. abntex. text editoration.

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

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). Entretanto

1.2 Objetivo principal

O objetivo principal do projeto é comprimir uma Rede Neural que calcula a velocidade de veículos. Para fazer isso será necessário utilizar uma série de métodos e técnicas de compressão, como destilamento de conhecimento, poda e quantização de Redes Neurais.

1.3 Metodologia

1.4 Estrutura do documento

2

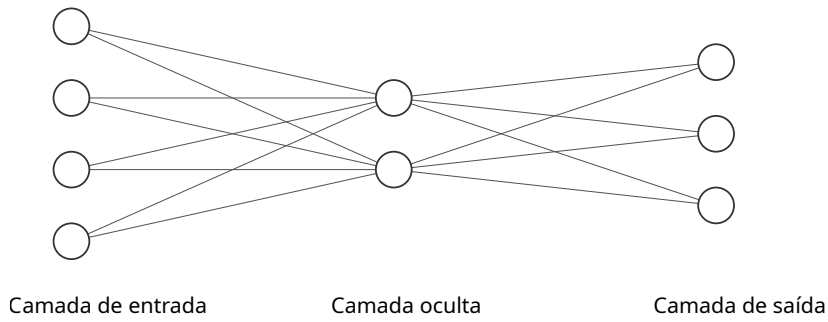
Conceitos básicos

Isto é uma sinopse de capítulo. A ABNT não traz nenhuma normatização a respeito desse tipo de resumo, que é mais comum em romances e livros técnicos.

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



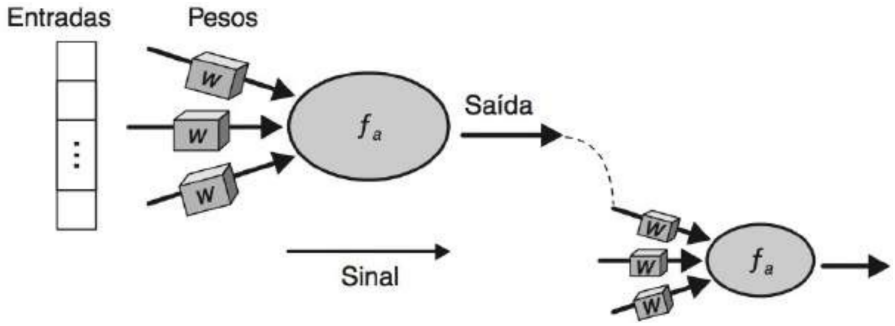
Fonte: Autor

O neurônio é uma parte fundamental de uma ANN, nele que o aprendizado é armazenado através do reforço de conexões com outros neurônio. 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



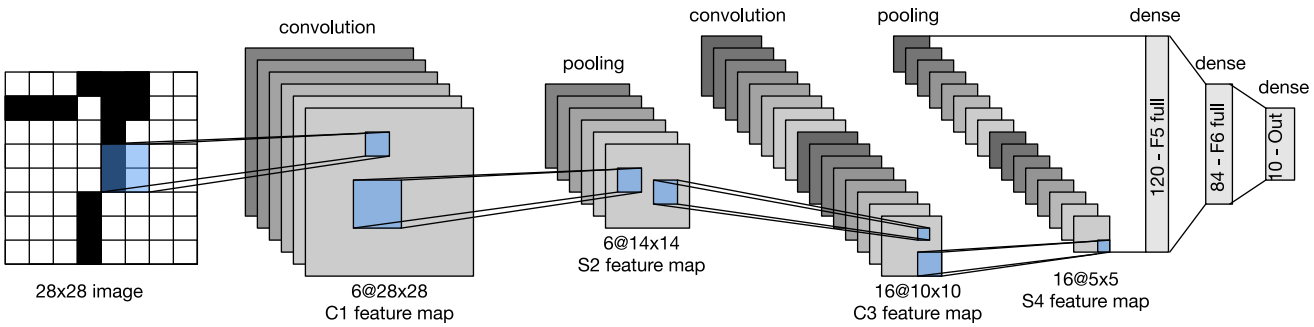
Fonte: (FACELI et al., 2011)

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.

Figura 3 – Arquitetura da LeNet



Fonte: (ZHANG et al., 2023)

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.

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 o maior valor 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. (TODO: Adicionar figuras, referências e detalhar mais)

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, a camada totalmente conectada as interpreta e gerar uma resposta. (TODO: Revisar texto)

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 realizam injeção de ruído nas imagens originais.

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 objetivo principal remover pesos ou neurônios que são redundantes ou irrelevantes para o problema, além disso reduz o *overfitting*(superajuste). (TODO: escrever sobre poda de redes neurais)

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)

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 $\alpha - 1$ 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)

3

Trabalhos relacionados

3.1 Trabalhos acadêmicos

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 detecta 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 4) é a ResNet-50 (HE et al., 2016) e o modelo estudante é gerado pelo Código 5. 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 1.

Na Tabela 1, 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 6, 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 Customização DCOMP

5.1 Lista de códigos

Usado para criar a lista de códigos, adicionar sintaxe highlight, enumerar as linhas e colorir o fundo, para dar destaque a implementação.

Sintaxe básica:

```
\begin{codigo}[!htb]
    \caption{Espaço para o título do código}
    \label{Espaço para o label do código, para ser usado na referência}
    \begin{lstlisting}[language = Linguagem de programação a ser usada]
        <CÓDIGO>
    \end{lstlisting}
\end{codigo}
```

Código 1 – Código PHP

```
1 <?php
2
3 echo 'Olá mundo!';
4 print 'Olá mundo!';
```

5.2 Lista de Algoritmos

Usado para criar a lista de algoritmos ou pseudocodigos.

Sintaxe básica:

Código 2 – Código python

```python
import numpy as np

def incmatrix(genl1, genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

Código 3 – Codigo Java

```java
public class Factorial{
    public static void main(String[] args){
        final int NUM_FACTS = 100;
        for(int i = 0; i < NUM_FACTS; i++)
            System.out.println( i + "! is " + factorial(i) +
               factorial(i) factorial(i) factorial(i));
    }

    public static int factorial(int n){
        int result = 1;
        for(int i = 2; i <= n; i++)
            result *= i;
        return result;
    }
}
```

```
\begin{algoritmo}[!htb]
    \caption{Espaço para o título do algoritmo ou pseudocodigo}
    \label{label do do algoritmo ou pseudocodigo, para ser usado na referência}
    <ESPAÇO RESERVADO PARA USAR SEU PACOTE FAVORITO DE CÓDIGOS>
\end{algoritmo}
```

Algoritmo 1 – Algoritmo exemplo

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
    read current;
    if *understand* **then**
        go to next section;
        current section becomes this one;
    else
        go back to the beginning of current section;
    end
end

6
Conclusão

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetuer nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.

Sed eleifend, eros sit amet faucibus elementum, urna sapien consectetuer mauris, quis egestas leo justo non risus. Morbi non felis ac libero vulputate fringilla. Mauris libero eros, lacinia non, sodales quis, dapibus porttitor, pede. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi dapibus mauris condimentum nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Etiam sit amet erat. Nulla varius. Etiam tincidunt dui vitae turpis. Donec leo. Morbi vulputate convallis est. Integer aliquet. Pellentesque aliquet sodales urna.

Referências

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 6028*: Resumo - apresentação. Rio de Janeiro, 2003. 2 p. Citado na página 5.

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 21.

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, 23 e 31.

HINTON, G.; VINYALS, O.; DEAN, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. Citado na página 20.

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 16 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 na página 20.

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 19.

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 22 e 23.

ZHANG, A. et al. *Dive into Deep Learning*. [S.l.]: Cambridge University Press, 2023. <https://D2L.ai>. Citado na página 18.

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 4 – 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 5 – 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 6 – 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 – Morbi ultrices rutrum lorem.

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

ANEXO B – Cras non urna sed feugiat cum sociis natoque penatibus et magnis dis parturient montes nascetur ridiculus mus

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetuer nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

ANEXO C – Fusce facilisis lacinia dui

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.