DISSERTAÇÃO DE MESTRADO

Análise de Quantização para Codificação de Redes Neurais sem Retreino

Marcos V. P. Tonin

Brasília, Setembro de 2021

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO

Análise de Quantização para Codificação de Redes Neurais sem Retreino

Marcos V. P. Tonin

Dissertação de Mestrado submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. Ricardo Lopes de Queiroz, Ph.D., PPGEE / UnB
Orientador

Prof. Eduardo Peixoto Fernandes da Silva, Ph.D., PGEA / UnB
Examinador Interno

Prof. Eduardo Antônio Barros da Silva, Ph.D., UFRJ
Examinador Externo

FICHA CATALOGRÁFICA

TONIN, MARCOS
Análise de Quantização para Codificação de Redes Neurais sem Retreino [Distrito Federal] 2021.
xvi, 56 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2021).
Dissertação de Mestrado - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica
1. Compressão de redes neurais
2. Quantização
3. *Open Neural Network Exchange* (ONNX)
4. Recursos limitados
I. ENE/FT/UnB
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

TONIN, M. V. P. (2021). *Análise de Quantização para Codificação de Redes Neurais sem Retreino*. Dissertação de Mestrado, Publicação: PPGENE.DM-XXX/AA, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 56 p.

CESSÃO DE DIREITOS

AUTOR: Marcos V. P. Tonin
TÍTULO: Análise de Quantização para Codificação de Redes Neurais sem Retreino.
GRAU: Mestre em Engenharia Elétrica ANO: 2021

Marcos V. P. Tonin
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Tradidi quod et accepi
I Cor. 11,23

AGRADECIMENTOS

Durante a produção desse trabalho eu recebi diversos tipos de auxílio.

Ao meu orientador, Professor Ricardo Queiroz, o qual com seu maciço conhecimento sempre fez progredir dentro da pesquisa.

Aos meus professores, por todo conhecimento passado.

À minha família, por todo suporte e por todos conselhos acadêmicos recebidos desde de minha tenra idade.

Marcos V. P. Tonin

RESUMO

O aprendizado de máquinas e o aprendizado profundo são utilizados para a resolução de diversos problemas, em diferentes áreas de atuação. Esse fato impulsiona o desenvolvimento de redes neurais, além de estimular o crescimento do tamanho destas. Este estudo propõe um método para reduzir o tamanho de redes neurais sem retreiná-las, relacionando a entropia dos pesos dos modelos e a acurácia dos modelos. Parte deste estudo foi dedicado à distribuição dos pesos, procurando semelhanças entre elas e as distribuições conhecidas. Com intuito de reduzir o tamanho da rede foi realizada a compressão do modelo por meio de vários tipos de quantização. Ao final deste estudo, indica-se que é possível diminuir o tamanho da rede em 8 vezes, com um prejuízo não maior que 0,8% para as métricas de acurácia, além de mostrar que quantização com *deadzone* possui um bom resultado para as redes testadas. E assim, a quantização e a codificação recomendadas podem ser incorporadas a um formato de distribuição de redes neurais.

ABSTRACT

Machine learning and deep learning are used to solve different problems in different areas of expertise. This fact drives the development of neural networks, in addition to stimulating the growth of their size. This study proposes a method to reduce the size of neural networks without retraining them, relating the entropy of the weights of the models and the accuracy of the models. Part of this study is about the distribution of weights, their similarities, and specific distributions. We studied various types of quantization in the compression of neural networks. This study indicates that it is possible to reduce the size of the network by 8 times, with a maximum loss of 0.8% for the accuracy metrics. The recommended quantization and encoding may be incorporated into a format for the deployment of neural networks.

CONTEÚDOS

LISTA DE FIGURAS

LISTA DE TABELAS

Símbolos

acc	Acurácia
acc_1	Acurácia Top 1
acc_5	Acurácia Top 5
err_1	Erro Top 1
err_5	Erro Top 5
H	Entropia
$f1s$	F1-*Score*
rec	*Recall*
pre	Precisão
X_q	Valor quantizado
$\widehat{X}$	Valor reconstruído
Δ	Passo de quantização
$\eth$	Tamanho relativo do *deadzone*
σ	Desvio padrão

Siglas

2D	Duas dimensões
3D	Três dimensões
ACF	*Autocorrelation Function*
CNN	Redes Neurais Convolucionais
DKL	Divergência de Kullback-Leibler
DL	*Deep Learning*
GC	Ganho Cumulativo
GCD	Ganho Cumulativo Descontado
GCDI	Ganho Cumulativo Descontado Ideal
GCDN	Ganho Cumulativo Descontado Normalizado
IoT	*Internet of Things*
IA	Inteligência Artificial
ILSVRC	*ImageNet Large Scale Visual Recognition Challenge*
ML	*Machine Learning*
MSE	*Mean Square Error*
MAE	*Mean Absolute Error*
MRR	*Mean reciprocal rank*
PDF	*Função densidade de probabilidade*

PSNR	*Peak Signal-to-Noise Ratio*
RNN	Redes Neurais Recorrente
RN	Redes Neurais
RMSE	*Root Mean Square Error*
SSE	*Sum Square Error*
SAE	*Sum Absolute Error*
ONNX	*Open Neural Network eXchange*

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O uso de *Machine Learning* (aprendizado de máquinas, ML) para resolução de problemas nas mais variadas áreas é cada vez mais frequente e aplicável em diferentes contextos, sendo capaz de responder a desafios reais [1]. Entre suas aplicações podemos destacar desde as mais simples como verificar se um *e-mail* é ou não *spam* [2], o reconhecimento de letras em cartas [1], além de questões mais personalizadas, como quais situações e elementos podem afetar a retenção de alunos em uma determinada universidade [3]. Além disso, ML pode ser usado em problemas mais complexos como: detecção de derramamento de óleo pelas imagens de radar [4], direção segura em carros autônomos [5] e detecção de fraudes financeiras [6].

Assim, do mesmo modo que modelos de ML, também as redes neurais (RN's) que são modelos computacionais inspirados (que buscam imitar) o funcionamento neural humano. Atualmente as RN's ganham espaço em diversas áreas mais robustas. Nesse contexto, surge a necessidade de desenvolver modelos maiores que demandam de mais espaço de armazenamento e maior processamento computacional.

Outro aspecto que ganha notoriedade nos dias atuais é a *Internet of Things* (internet das coisas, IoT) que pode ser entendido como a interconexão de objetos do dia a dia pela *internet*. A área de internet das coisas tem promovido desenvolvimento rápido de várias dispositivos e aplicações, como o uso de aprendizado de máquina em dispositivos de sistema de compartilhamento de bicicletas [7] e o uso de equipamentos IoT para cuidados de saúde [8]. Atrelada a essa definição, existe o campo *edge computing*, que pode ser entendido como equipamento IoT que tem a capacidade de processar dados perto de sua origem [9]. Um bom exemplo é IoT industrial que realiza a incorporação da inteligência artificial (IA) com a finalidade de encontrar novos *insights* industriais [10]. Essa vinculação se estende para as demais áreas de IoT. Com isso, o uso de IA com *edge computing* também tem se desenvolvido [10].

Os equipamentos de IoT possuem recursos e energia limitados [11], [12]. Normalmente, os modelos de IA precisam ser compactados e reduzidos para aplicativos IoT e *edge computing*. Assim, as tarefas de IA podem sobrecarregar os dispositivos de IoT e *edge computing*, pois estes têm pouco poder computacional e baixa capacidade de armazenamento de dados [12]. Outro tipo de equipamento que entra nessa lista são os celulares, uma vez que possuem recursos e energia restritos tornando importante a compressão dos modelos de ML [11].

Em consonância com o tema de ML surge um relevante questionamento sobre a interoperabilidade dos modelos de ML, devido à existência de vários arcabouços de ML e, em especial, de *deep learning* (*aprendizado profundo*, DL), como *Pytorch* [13], *Tensorflow* [14], sendo que diversos arcabouços não possuem integração entre si. Diante disso, surge a preocupação em não se prender a

um *framework* específico e suas ferramentas. Com a finalidade de preencher essa lacuna, o formato ONNX (*Open Neural Network eXchange*) [15] foi desenvolvido com o objetivo de proporcionar a interoperabilidade de modelos de ML e de DL.

1.2 MOTIVAÇÃO

O aumento do desenvolvimento de modelos de RN's decorre, principalmente, da ampliação do seu uso para os mais diversos problemas. Com o desenvolvimento da área em rápido progresso, os tamanhos desses modelos também costumam ficar maiores, uma vez que a tendência é que os modelos sejam cada vez mais especializados e sofisticados.

Em contraposição ao cenário do aumento desses modelos, temos o uso de ML nos dispositivos citados no parágrafo anterior (celulares, equipamentos IoT), em que pese esses dispositivos terem seus recursos e espaço limitados. Assim, há uma preocupação com o tamanho que os modelos podem ocupar nos equipamentos e do poder computacional exigido.

1.3 DEFINIÇÃO DO PROBLEMA

O padrão aberto ONNX busca sanar as dificuldades na distribuição de RN's, para permitir que vários arcabouços possam trabalhar em conjunto. Os meios para se reduzir o tamanho das RN's sem perder muita acurácia ganham cada vez mais espaço na literatura. A maioria das pesquisas envolvem técnicas de compressão com retreino de RN ou a mudança da estrutura da RN [16]–[21]. Contudo, esse tipo de compressão de modelos de ML necessita de um grande poder computacional. Por outro lado, a compressão sem envolver retreino é um campo ainda pouco explorado.

1.4 OBJETIVOS

Os objetivos desse trabalho são:

1. Analisar a distribuição de amplitude dos pesos;
2. Analisar se existe correlação de algum modo entre os pesos das RN's;
3. Determinar a melhor forma de quantização para os pesos e vieses de uma RN;
4. Aumentar a facilidade de distribuição de redes neurais;
5. Propor um método de comprimir os pesos e vieses no formato ONNX, sem o uso de retreino, de uma RN de forma a reduzir seu tamanho sem perda significativa de acurácia (de seu funcionamento).

1.5 PUBLICAÇÕES

A pesquisa desenvolvida neste trabalho teve como base a publicação:

- "Codificação de Redes Neurais sem Retreino", Marcos Vinícius Tonin e Ricardo L. de Queiroz. Simpósio Brasileiro de Telecomunicações e Processamento de Sinais, 2021. [22]

1.6 APRESENTAÇÃO DO MANUSCRITO

O Capítulo 2 apresenta alguns trabalhos correlatos a esta pesquisa, além dos conceitos necessários para explicar o método proposto. No Capítulo 3, as comparações entre as quantizações de vários tipos são expostas, bem como a comparação das distribuições dos pesos dos modelos com distribuições de referências. Por fim, conclusões, trabalhos futuros estão presentes no Capítulo 4.

2 REVISÃO BIBLIOGRÁFICA

2.1 INTELIGÊNCIA ARTIFICIAL

Inteligência Artificial (IA) é uma ciência que tentar fazer coisas que requerem uma inteligência humana [23]. De certa forma, é a tentativa das máquinas reproduzirem comportamentos que são essencialmente humanos em suas atividades. Podemos dizer que a IA, como um todo, engloba outras duas principais subcategorias, como ilustrado na Figura 2.1: *Machine Learning* (ML) e *Deep Learning* (DL) [24].

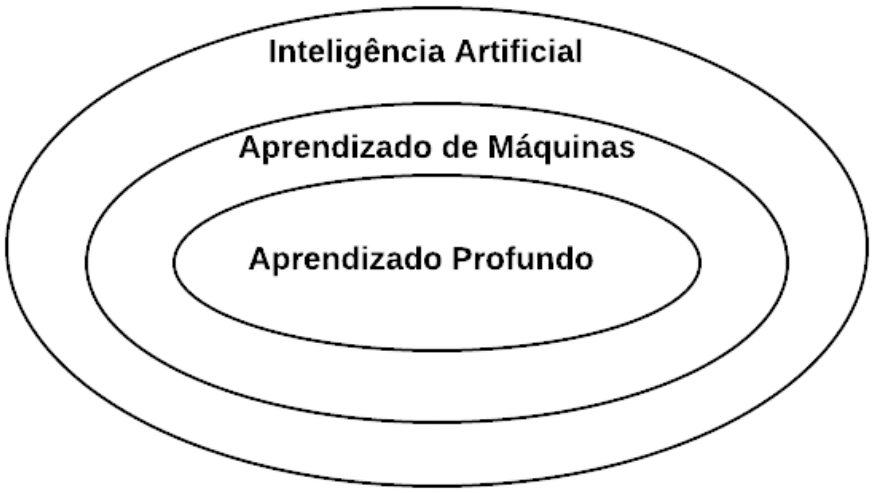


Figura 2.1: Estrutura da ciência IA.

- ***Machine Learning*** é um subgrupo de IA que busca automatizar a análise e tomada de decisão sem supervisão ou interferência humana [23]. A maioria dos algoritmos de ML tendem a melhorar seu funcionamento a partir da experiência, podendo mudar seu comportamento e suas operações [23]. Por outro lado, pode ser entendido como a capacidade do computador aprender sem estar explicitamente programado para isso [24].
- ***Deep Learning*** é um ramo de ML que tem a finalidade de imitar o funcionamento neural humano. *Deep Learning* também é conhecido por permitir processar um nível elevado de dados para encontrar padrões e relações que para os humanos não são facilmente encontrados [24]. Podemos abranger os modelos DL em redes neurais que possuam três ou mais camadas onde os computadores buscam imitar a forma como os neurônios funcionam [23].

2.2 REDES NEURAIS

A arquitetura de um modelo DL pode ser considerada uma rede neural (RN), a depender se a RN tem mais de três camadas, essa diferença será explicada posteriormente, quando definirmos o que é camada e seus tipos. Uma rede neural artificial é uma técnica que procura simular o mecanismo de conhecimento em organismos biológicos [25]. Podemos descrever o mecanismo biológico contendo neurônios que se conectam através de áxions e dendritos, os quais se comunicam através de sinapses (local onde agem neurotransmissores) [25]. Portanto, as RN's procuram simular esse mecanismo do sistema nervoso humano, fazendo um paralelo em que os neurônios seriam as unidades computacionais, conectados por pesos e cada entrada de neurônio é multiplicada por esse peso. A Figura 2.2 mostra a representação da RN. Assim, a rede neural artificial computa uma função das entradas propagando as entradas do neurônio para sua saída, utilizando os pesos como parâmetros intermediários [25]. A partir da Equação 2.1, temos a definição matemática do funcionamento de um neurônio de uma RN, as entradas (e_i) têm seus valores multiplicados pelo peso que as conecta ao neurônio (p_i). No neurônio ocorre a agregação das entradas multiplicadas pelos pesos ($\sum$) e a aplicação da função daquele neurônio ($f()$), produz a saída y.

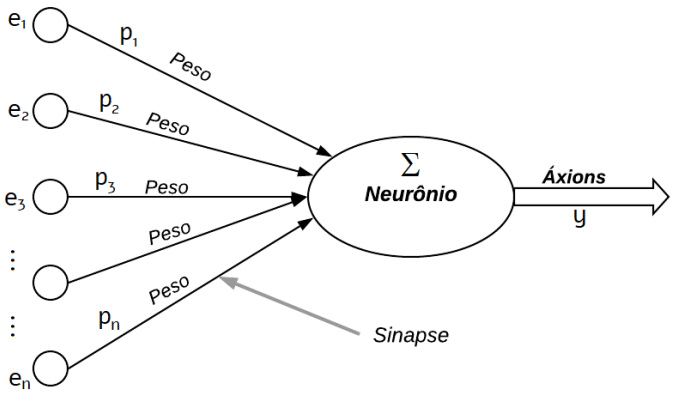


Figura 2.2: Componentes básicos de uma Rede Neural Artificial.

$$y = f\left(\sum_{i}^{n} e_i p_i\right) \tag{2.1}$$

As redes neurais podem ser definidas como sistemas massivos e paralelos, compostos por unidades de processamento simples que computam determinadas funções matemáticas [26]. A partir de um conjunto de exemplos apresentados, esses sistemas conseguem generalizar o conhecimento adquirido para um conjunto de dados desconhecidos [27].

Outro fator importante sobre o funcionamento das RN's é que o 'conhecimento' acontece quando um peso tem seu valor atualizado [25] em decorrência de atividades que ocorreram. Muitas vezes, essas atividades são conhecidas como treinamento em que a RN é alimentada com entradas sabendo a saída correta. A partir dessas entradas e saídas, a RN é capaz de determinar se a saída

produzida (predição) por ela é a correta ou não e, com essa informação, os pesos são ajustados e refinados várias vezes para prover uma predição acurada [25].

2.2.1 *Deep Learning* e *Machine Learning*

Os modelos de DL podem ser comparados com modelos clássicos que são comumente usados em ML com alto nível de abstração [23], uma vez que a maioria dos elementos básicos de DL são inspirados em algoritmos tradicionais de ML [23]. A vantagem da DL acontece porque podemos colocar várias unidades básicas juntas (separadas em pelo menos 3 camadas), além da aquisição do conhecimento por cada peso para diminuir o erro de predição [25].

O ganho de DL em comparação a um modelo de ML é desencadeado quando estas unidades computacionais básicas são combinadas e, os pesos dos modelos são treinados e alterados dinamicamente [25]. Uma diferença notável entre ML e DL consiste que para os modelos de DL (e, consequentemente, para RN's) existe a tendência de ter uma maior acurácia quanto maior for a quantidade de dados usados, como ilustrado na Figura 2.3 [25].

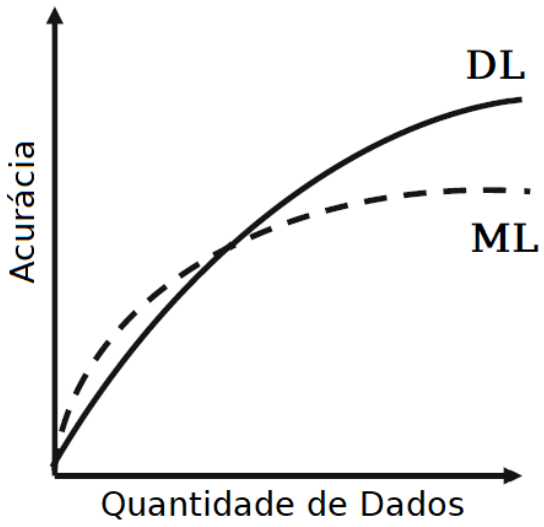


Figura 2.3: Comparação entre DL e ML, adaptado de [25].

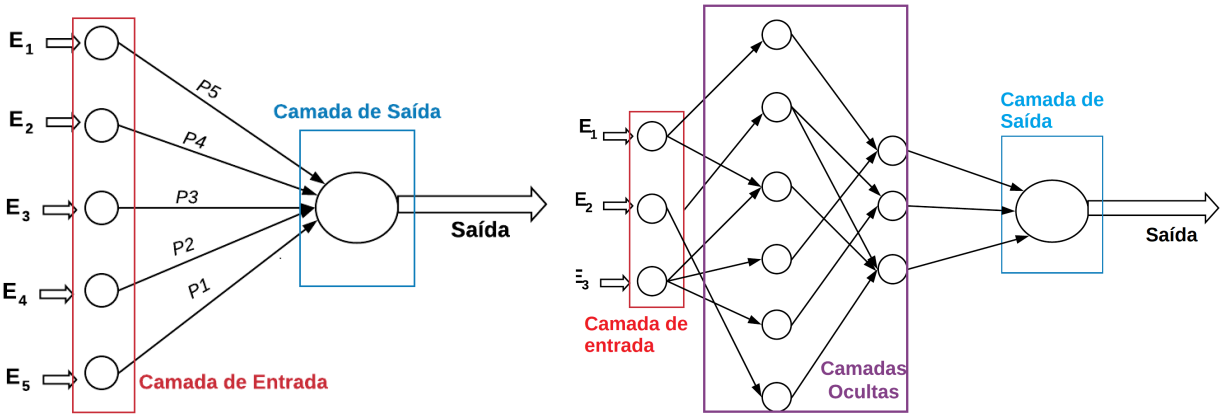
2.2.2 Estrutura Básica

Uma RN pode ser organizada em camadas e nós; e, classificada em camadas simples e redes de camadas múltiplas.

- **Camada simples:** composta pela camada de entrada e nó de saída (Figura 2.4(a)).
- **Camadas múltiplas:** composta por uma única camada de entrada, várias camadas ocultas e uma de saída (Figura 2.4(b)).

Nas Figuras 2.4(a) e 2.4(b), E faz referência às entradas da camada de entrada e p são os valores

dos pesos.



(a) Camada simples. (b) Camadas múltiplas.

Figura 2.4: Camadas simples e múltiplas.

Com a definição de camadas, podemos definir que as redes neurais serão consideradas modelos *deep learning* se houver mais de três camadas. Assim, deve ter em sua composição, além da camada de entrada e de saída, ao menos uma camada oculta. Portanto, podemos concluir que uma RN de camada simples não é um modelo DL e que uma RN de camada múltipla será um modelo DL.

Vimos, então, que existem 3 elementos básicos em uma RN: os nós que são as unidades computacionais de uma RN; os pesos, as conexões entre cada nó que recebe um valor e; as camadas, um conjunto de nós que tem a mesma distância da entrada.

Há mais um elemento usado nas RN's: o viés (*bias*, em inglês), esse tem o efeito de aumentar ou diminuir a entrada de um nó (neurônio da rede neural), dependendo de seu valor [28]. O viés se liga diretamente ao neurônio como na Figura 2.5. Pela Equação 2.2 que é derivada da equação 2.1, vemos que o viés (v) se liga diretamente ao nó.

$$y = f\left(v + \sum_{i}^{n} e_i p_i\right) \tag{2.2}$$

2.3 OPEN NEURAL NETWORK EXCHANGE

Os modelos de IA muitas vezes têm seu desenvolvimento e inferência presos a um arcabouço/-ferramenta específica, uma vez que a interoperabilidade com outras ferramentas não existe. E, com a intenção de sanar essa barreira, o *Open Neural Network eXchange* foi desenvolvido.

O ONNX [15] é um formato para modelos de IA, para DL (tanto CNN quanto RNN) e ML. Esse formato provê um arquivo de fonte aberto (*open source*) com o objetivo de ser comum às RN's e independentes de arcabouços, por exemplo: Pytorch [13], Keras [29], Tensorflow [14], MxNet [30].

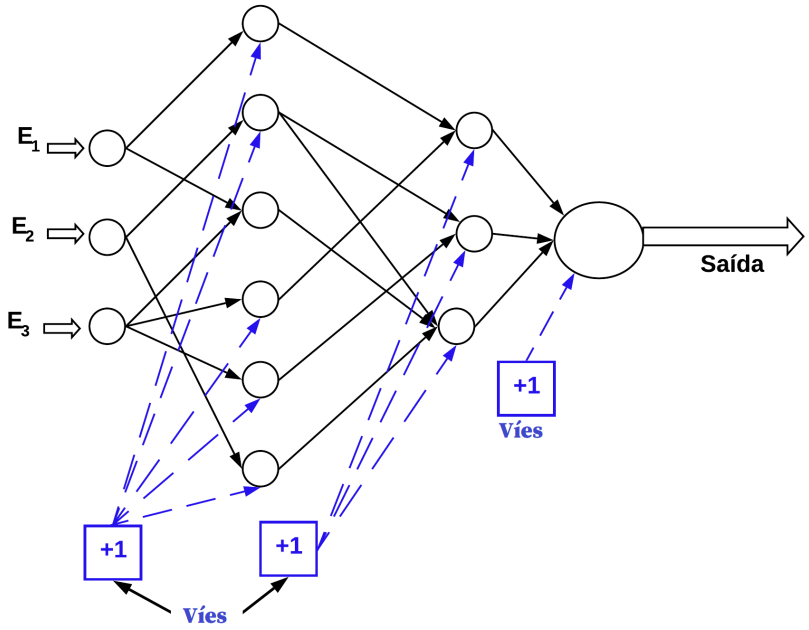


Figura 2.5: Viés em uma RN.

Para isso, o ONNX funciona definindo um conjunto comum de operadores internos (que podem ser considerados como os blocos que constroem um modelo de IA), os tipos de dados padrões, além de definir um modelo extensível de grafos de computação. Um ponto importante para esse trabalho é que esse formato representa os pesos e os vieses, majoritariamente, como ponto flutuante de 32 bits, no padrão IEEE 754 [31] e, por vezes, como ponto flutuante de 64 bits [15].

A atribuição mais importante do ONNX é ser uma representação intermediária de um modelo que permite a interoperabilidade de um ambiente para outro, não se restringindo a um único *framework*, conforme mostra o fluxo da Figura 2.6. Portanto, sustenta a ideia de que embora tenha sido usado, por exemplo, o *Pytorch* para criar e treinar uma rede, podemos avaliá-la e utilizá-la com, por exemplo, *Tensorflow*.

2.3.1 Modelos fornecidos

Além de possibilitar a exportação de modelos de IA próprios, o ONNX provê uma série de modelos conhecidos de exemplos [15], que são divididos em:

- Classificação de imagem;
- Detecção e segmentação de imagem;
- Análise facial, corporal e gestual;
- Manipulação de imagem e;
- Compreensão de máquinas.

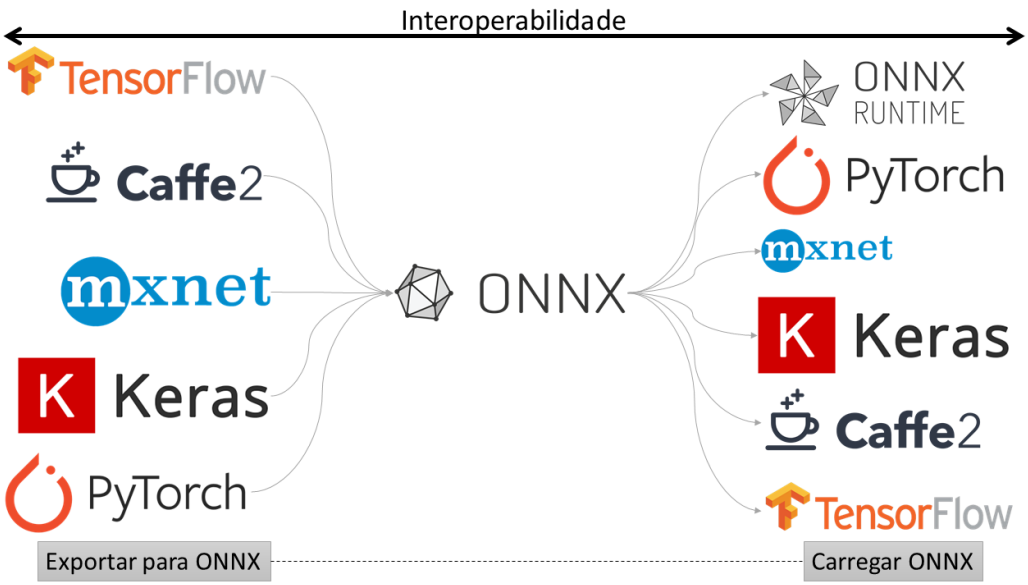


Figura 2.6: Ecossistema ONNX, adaptado de [32].

Para este trabalho, os modelos de classificação de imagens são os mais utilizados para teste e validação. Nessa classificação temos as redes: *MobileNet* [33], *AlexNet* [34], *GoogLenet* [35], *VGG* [36], *SqueezeNet* [37], *Shufflenet* [38] e *EfficientNet* [39]. Essas redes citadas e outras serão utilizadas para o desenvolvimento deste trabalho.

2.4 MÉTRICAS PARA *MACHINE LEARNING* E *DEEP LEARNING*

Os modelos de IA (em especial de ML e DL) têm por essência trabalhar de forma automatizada, sem intervenção humana, por isso para esses modelos é importante verificarmos e validarmos o seu funcionamento. Outro ponto importante se deve ao fato de que esses modelos fazem atividades diferentes, seja no seu funcionamento ou na sua utilidade. Os principais tipos de atividades dos modelos de IA são:

- **Classificação:** esses modelos tem como função tentar descobrir padrões entre os dados e fazer alguma conclusão sobre eles. Por exemplo, um modelo pode ler um *email* e classificá-lo em *spam* ou não.
- **Regressão:** através do estudo e análise dos dados de entrada, o modelo tenta prever outra informação. Por exemplo, retornar o valor de um imóvel, a partir de suas características iniciais.
- **Ordenamento:** esse tipo de modelo buscar ordenar uma lista baseado nos dados de entrada.

Como há diferentes tipos de modelos que fazem variadas atividades não podemos usar uma métrica universal para tudo, costuma-se utilizar métricas específicas a depender do tipo de atividade do modelo, por exemplo, para [40]:

- **classificação:** acurácia, precisão, *recall*, *F1-score*;
- **regressão:** soma dos erros quadrados (SSE), soma dos erros absolutos (SAE), média dos erros quadrados (MSE), média dos erros absolutos (MAE);
- **geração de imagem:** PSNR (relação sinal ruído de pico);
- **Ordenamento:** *Mean reciprocal rank* (MRR).

Neste trabalho, as métricas que mais nos interessam são as de classificação e ordenamento.

2.4.1 Métricas de Classificação

A matriz de confusão é importante para entender as métricas relacionadas à classificação. Ela detalha a classificação de cada classe feita pelo modelo e a classificação que seria correta [40], portanto em suas colunas estão dispostos os resultados esperados e nas linhas estão os resultados preditos pelo modelo. A Tabela 2.1 considera um exemplo de 100 elementos, divididos em 50 para cada classe. A Tabela 2.2 traz outro exemplo com 3 classes e 160 elementos.

Tabela 2.1: Tabela confusão para 2 classes.

Esperada / Predito	**Classe 1**	**Classe 2**
Classe 1	40	10
Classe 2	5	45

Tabela 2.2: Tabela confusão para 3 classes.

Esperada / Predito	**Classe 1**	**Classe 2**	**Classe 3**
Classe 1	40	8	2
Classe 2	5	35	10
Classe 3	5	7	48

Nas Tabelas 2.1 e 2.2, as colunas representam a saída esperada do modelo e as linhas, por sua vez, representam o que foi predito pelo modelo.

2.4.1.1 Acurácia

Acurácia é uma medida simples, a razão entre o número de predições corretas e o número total de predições [40]:

$$acc = \frac{\text{Predições Corretas}}{\text{Total Predições}}. \tag{2.3}$$

Para a Tabela 2.1, a acurácia é igual 0,85; para a Tabela 2.2, acurácia é 0,7687.

2.4.1.2 Acurácia por classe

A acurácia por classe é uma variação da acurácia, mas calculando a média da acurácia de cada classe separada.

$$acc_{classe} = \frac{\sum \frac{\text{Predições Corretas por Classe}}{\text{Total de Predições por Classe}}}{\text{Total de Classes}}. \tag{2.4}$$

Podemos dizer que a acurácia simples é uma micromédia (Eq. 2.3) e acurácia por classe, uma macromédia (Eq. 2.4) [40]. Na Tabela 2.1, temos o valor igual ao da acurácia simples, mas para a Tabela 2.2, um resultado de 0,7666 seria obtido:

2.4.1.3 Precisão

Pela Equação 2.5, a precisão é utilizada para indicar a relação entre predições corretas para a classe X e todas as predições para classe X [40]:

$$\text{precisão} = \frac{\text{Predições Corretas da Classe X}}{\text{Total de Predições para Classe X}}. \tag{2.5}$$

Para a Tabela 2.1, a precisão da classe A é 0,888; enquanto que para a classe B a precisão é 0,8181. Na Tabela 2.2, as precisões são de 0,8; 0,7 e 0,8 para as classes A, B e C, respectivamente.

2.4.1.4 *Recall*

O *Recall* da classe X é a métrica usada para indicar a relação entre as predições corretas da classe X e total de predições corretas que foram feitas (Eq. 2.6) [40].

$$recall = \frac{\text{Predições Corretas da Classe X}}{\text{Total de Predições Corretas para todas as Classes}}. \tag{2.6}$$

Na Tabela 2.1, o *recall* é 0,8 para a classe A, enquanto que para a classe B o *recall* é de 0,9. Na Tabela 2.2, as classes A, B e C possuem um *recall* de 0,8; 0,7 e 0,8; respectivamente.

2.4.1.5 *F1-score*

É uma forma de combinar as métricas *recall* e precisão, para associá-las usa-se a média harmônica, conforme Equação 2.7.

$$F1_{score} = 2 \times \frac{\text{precisão} \times recall}{\text{precisão} + recall}. \tag{2.7}$$

Os resultados F1 são:

- **Tabela 2.1:** 0,842 e 0,857 para as classes A e B, respectivamente.
- **Tabela 2.2:** 0,8; 0,7 e 0,8 para as classes A, B e C, respectivamente.

2.4.2 Métricas de Regressão

Nas tarefas de regressão, o modelo prediz um valor. Por exemplo, o preço de um produto pode ser definido com o uso de um modelo de regressão [40]. Logo, as métricas de classificação não são uma boa escolha, uma vez que essas trabalham com classes fixas.

2.4.2.1 Soma de Erros

Uma das métricas mais comuns é a *root mean square error* (RMSE, Eq. 2.8), ou ainda, raiz quadrada do erro médio [40], porém aqui podemos também descrever as métricas SSE (Eq. 2.9), MSE (Eq. 2.10), SAE (Eq. 2.11) e MAE (Eq. 2.12).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{n}}, \tag{2.8}$$

$$SSE = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2, \tag{2.9}$$

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{n}, \tag{2.10}$$

$$SAE = \sum_{i=1}^{n}(|y_i - \widehat{y}_i|), \tag{2.11}$$

$$MAE = \frac{\sum_{i=1}^{n}(|y_i - \widehat{y}_i|)}{n}. \tag{2.12}$$

Para as Equações 2.8, 2.9, 2.10, 2.11 e 2.12, temos que n é a quantidade total de pontos, y_i é referente ao i-ésimo elemento da lista y, já $\widehat{y}_i$ é o elemento na posição i da segunda lista, $\widehat{y}$.

2.4.3 Ordenamento

O tipo de tarefa de ordenamento é similar à tarefa de classificação, desse modo podemos compartilhar algumas métricas como: precisão, *recall* e F1 score [40]. Um modelo que pontua uma série de elementos e os ordena é considerado um tipo de modelo de ordenamento.

2.4.3.1 Ganho Cumulativo Descontado Normalizado

O ganho cumulativo descontado normalizado é, comumente, utilizado para aferir a eficiência dos algoritmos de pesquisa na internet, além de ser usado em outras aplicações similares. Para entender o ganho cumulativo descontado normalizado (GCDN) temos que primeiro entender o que é ganho cumulativo (GC). Segundo a Equação 2.13, temos que GC (de uma lista de resultados de tamanho p) é definido como a soma da relevância de cada item até a posição p [41]. Cada item dessa

lista de tamanho p tem uma pontuação de relevância associada a ela. A pontuação de relevância pode ser dada, por exemplo, através de uma pesquisa feita entre uma população.

$$GC_p = \sum_{i=1}^{p} rel_i, \tag{2.13}$$

onde rel_i é a pontuação de relevância do resultado da posição i. A GC_p não é afetada pela troca de posição de resultados, assumindo que a troca é de um item com posição menor que p.

Assim, GCDN e sua versão não normalizada GCD (ganho cumulativo descontado) buscam explorar o conceito de que os resultados com maior probabilidade são mais relevantes (os itens que aparecem no início de uma lista ordenada são os de maior classificação) [40]. Portanto, essas métricas (GCDN e GCD) assumem que, em uma lista ordenada, os objetos mais relevantes são, também, os mais valiosos [41], [42]. Outro ponto, quanto menor a posição de um objeto relevante menos valioso se tornará, uma vez que será menos provável de ser examinado [41], [42].

A diferença entre o GCD e GC ocorre quando no ganho cumulativo a troca de posição não altera o resultado (dada a restrição da troca ser para uma posição menor que p), mas o GCD aplica descontos em itens mais abaixo da lista decrescente [40], sendo o GCD definido como [41], [42]:

$$GCD_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i} = \frac{\sum_{i=1}^{p}(2^{rel_i} - 1)}{log_2(1+i)}, \tag{2.14}$$

na Equação 2.14 em que rel_1 é a relevância do primeiro item. A divisão por $\log_2 i$ tem a função de penalizar os itens de acordo com sua posição. Por exemplo, o décimo item da lista teria sua pontuação de relevância (rel_{10}) dividida por $\log_2 10 = 3.322$.

O problema da métrica GCD é que ela pode variar de acordo com o tamanho da lista, pois uma lista maior terá maiores descontos aplicados. Então a versão normalizada da GCD, conhecida como GCDN, normaliza o resultado da GCD de forma que a variação de tamanho da lista não implique em mudanças de resultados. A GCDN divide a GCD pelo *score* perfeito da GCD, conhecido como DCG ideal (GCDI) [41], [42], o DCG ideal é alcançado quando a lista está ordenada perfeitamente pela relevância de cada item, da maior relevância para a menor. Assim, podemos definir a GCDN conforme a Equação 2.15.

$$GCDN_p = \frac{GCD_p}{GCDI}. \tag{2.15}$$

O $GCDN_p$ tem seu valor sempre entre 0 e 1.

2.4.3.2 *Reciprocal Rank*

A medida *reciprocal rank* pode ser utilizada em aplicações que possuem apenas um único item relevante [42]; a MRR, por sua vez, é utilizada para avaliar processos que produzem uma lista de

possíveis respostas, ordenadas pela probabilidade de importância [40], sendo mais indicada para quando há mais de um resultado relevante.

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{\text{posição}_i}, \tag{2.16}$$

sendo Q a quantidade total de consultas feitas e posição_i a colocação em que o elemento da consulta foi encontrado.

2.4.4 Métricas para Classificação de Imagem

Para o *dataset* ILSVRC (*ImageNet Large Scale Visual Recognition Challenge* [43]) foram propostas métricas de acurácia e de erro que são utilizadas para avaliar a classificação de imagem.

2.4.4.1 Erro e Acurácia Top 1

A acurácia Top 1 (acc_1) ocorre quando a saída mais provável do modelo corresponde à resposta esperada. Já, o erro (err_1) é quando essa saída não corresponde.

$$acc_1 = \begin{cases} 1, & SM_1 = GT, \\ 0, & SM_1 \neq GT \end{cases}, \tag{2.17}$$

$$err_1 = \begin{cases} 1, & SM_1 \neq GT, \\ 0, & SM_1 = GT \end{cases}, \tag{2.18}$$

em que GT é a saída verdadeira (*ground truth*), e SM_1 é a saída mais provável do modelo.

2.4.4.2 Erro e Acurácia Top 5

A acurácia Top 5 (acc_5) acontece quando uma das cinco saídas mais prováveis do modelo corresponde à resposta esperada. Já, o erro (err_5), quando as cinco saídas mais prováveis não correspondem a saída esperada.

$$acc_5 = \begin{cases} 1, & SM_1 = GT \text{ ou } SM_2 = GT \text{ ou } SM_3 = GT \text{ ou } SM_4 = GT \text{ ou } SM_5 = GT, \\ 0, & c.c. \end{cases}, \tag{2.19}$$

$$err_5 = \begin{cases} 1, & SM_1 \neq GT \text{ e } SM_2 \neq GT \text{ e } SM_3 \neq GT \text{ e } SM_4 \neq GT \text{ e } SM_5 \neq GT, \\ 0, & c.c. \end{cases}, \tag{2.20}$$

SM_1 até SM_5 correspondem às 5 saídas do modelo com maior probabilidade, sendo SM_1 a mais provável e SM_5, a quinta mais provável.

Para elucidar o funcionamento dessa métrica, utilizamos a Figura 2.7 como entrada do modelo e a Tabela 2.3 para exemplificar uma possível saída de um modelo a partir da Figura 2.7. Com intuito de demonstrar o funcionamento foram consideradas 3 cenários:

- **Cenário 1 - *Tiger cat* (saída esperada):** acurácia Top 1 e Top 5 é igual a 1 e $err_1 = err_5 = 0$, pois a *Tiger cat* é a classe mais provável do modelo fictício.
- **Cenário 2 - saída esperada *Egyptian cat*:** na Tabela 2.3, *Egyptian cat* é a terceira saída mais provável, portanto, temos que $acc_1 = 0$, $acc_5 = 1$, além de, $err_1 = 1$ e $err_5 = 0$.
- **Cenário 3 - *Bengal cat* (saída esperada):** a saída não está entre as 5 classes mais prováveis, logo o resultado será inverso ao do cenário 1. $acc_1 = acc_5 = 0$ e $err_1 = err_5 = 1$.

A Tabela 2.4 sintetiza os resultados para as métricas acurácia e erro (Top 1 e Top 5) dos cenários apresentados acima.

Figura 2.7: Imagem de exemplo para classificação de uma RN.

Tabela 2.3: Exemplo de saída de um modelo.

Rank	Classe	Probabilidade
1	*tiger cat*	0.432
2	*lynx, catamount*	0.281
3	*tabby, tabby cat*	0.183
4	*Egyptian cat*	0.0895
5	*tiger, Panthera tigris*	0.011

Tabela 2.4: Valores das métricas para diferentes resultados.

Cenário	Saída Esperada	Acurácia Top 1	Erro Top 1	Acurácia Top 5	Erro Top 5
1	*tiger cat*	1.0	0.0	1.0	0.0
2	*Egyptian cat*	0.0	1.0	1.0	0.0
3	*Bengal cat*	0.0	1.0	0.0	1.0

2.5 QUANTIZAÇÃO ESCALAR

A quantização escalar é uma etapa importante deste trabalho, após o entendimento da arquitetura das redes neurais, os pesos delas serão extraídos. Esses pesos, então, serão quantizados e reconstruídos a partir do valor quantizado. Para entender a quantização, é preciso analisar a conversão analógico-digital, que pode ser dividida em três processos básicos [44] como ilustrado na Fig. 2.8.

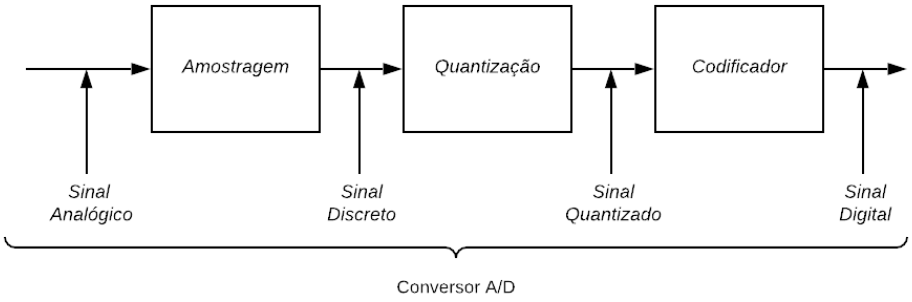


Figura 2.8: Fluxo do conversor analógico digital.

- **Amostragem:** trata da conversão de um sinal contínuo para um sinal discreto, obtendo amostras do sinal contínuo em um determinado instante [44], tornando o sinal discreto no tempo amostrado.
- **Quantização:** a conversão de um valor com amplitudes não estabelecidas em um valor discreto selecionado dentre possíveis finitos valores [44]. O sinal, portanto, é discreto no tempo e na amplitude.
- **Codificador:** cada valor quantizado será representado por uma sequência binária [44].

Em relação à quantização, essa pode ser entendida como a transformação de um sinal de amplitude contínua em um sinal com amplitude discreta [44]. Em resumo, a quantização converte do contínuo para o discreto [45].

Ainda, a quantização pode ser uniforme, contendo espaçamento igual entre cada nível existente, sendo o intervalo constante para cada nível [46], ou não uniforme, que tem diferentes espaçamentos

para cada nível.

2.5.1 Quantização uniforme

Há várias formas de realizar a quantização uniforme, as duas mais conhecidas são as *midrise* e *midtread* [47]. Seja X o valor de um peso a ser quantizado, X_q o valor que será passado ao codificador, $\widehat{X}$ o valor reconstruído pelo codificador, Δ o tamanho do passo de quantização e o operador $round(X)$ que arredonda o valor X para o número inteiro mais próximo. O processo de quantização por um quantizador *midtread* é dado por

$$X_q = round\left(\frac{X}{\Delta}\right), \tag{2.21}$$

enquanto a reconstrução é dada por

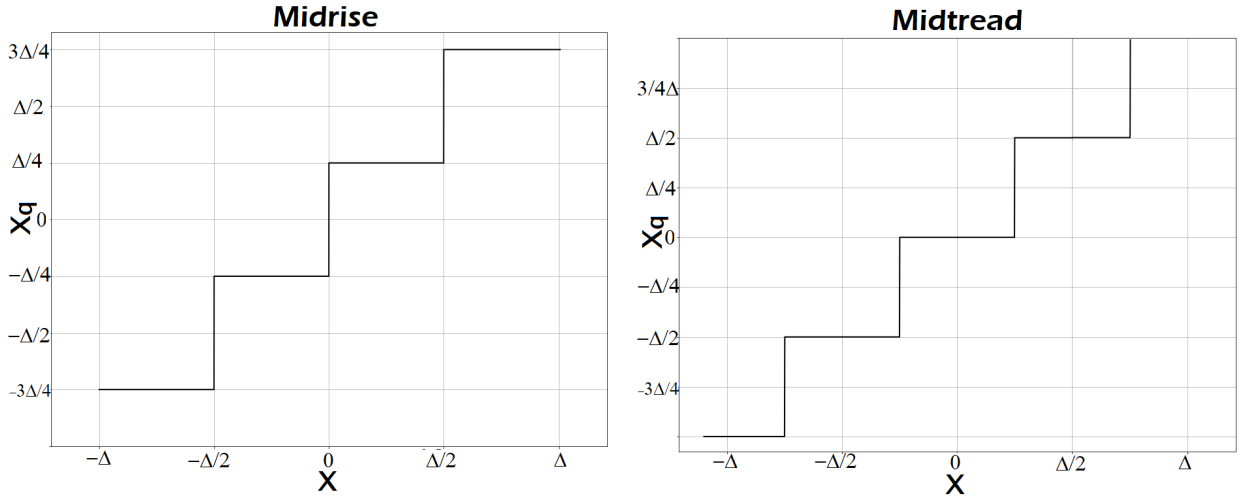
$$\widehat{X} = \Delta X_q. \tag{2.22}$$

Por outro lado, a quantização *midrise* (Eq. 2.23) e sua reconstrução (Eq. 2.24) podem ser definidas como:

$$X_q = s(X)\left\lceil\frac{X}{\Delta}\right\rceil, \tag{2.23}$$

$$\widehat{X} = s(X_q)\Delta\left(X_q - \frac{1}{2}\right), \tag{2.24}$$

onde a função s(X), de sinal, retorna 1 se o valor de X for positivo e -1, caso contrário, enquanto o operador $\lceil X \rceil$ é conhecido como *ceil* e retorna o arredondamento superior do valor X. As quantizações *midrise* e *midtread* são ilustradas na Figura 2.9, onde o passo de quantização usado foi $\frac{\Delta}{2}$.



(a) *Midrise*. (b) *Midtread*.

Figura 2.9: Quantização Uniforme: *Midrise* e *Midtread*.

A vantagem da *midrise* e *midtread* é que ambas são simétricas em relação ao centro e possuem o tamanho dos intervalos iguais. Uma das desvantagens da *midrise* é a de não possuir um nível zero [48].

2.5.2 Quantização com *deadzone*

A quantização com *deadzone* é semi-uniforme, onde o nível em torno do zero tem intervalo maior do que os demais (que terão sempre o mesmo intervalo) [49]. O processo de quantização é dado por

$$X_q = \begin{cases} 0, & \text{se } |X| < \delta\Delta, \\ s(X)\lceil(\frac{|X|}{\Delta} - \delta\rceil, & \text{c.c} \end{cases}, \tag{2.25}$$

enquanto a reconstrução é dada por

$$\widehat{X} = \begin{cases} 0, & \text{se } X_q = 0, \\ s(X_q)\Delta(|X_q| + \delta - 0.5), & \text{c.c.} \end{cases}. \tag{2.26}$$

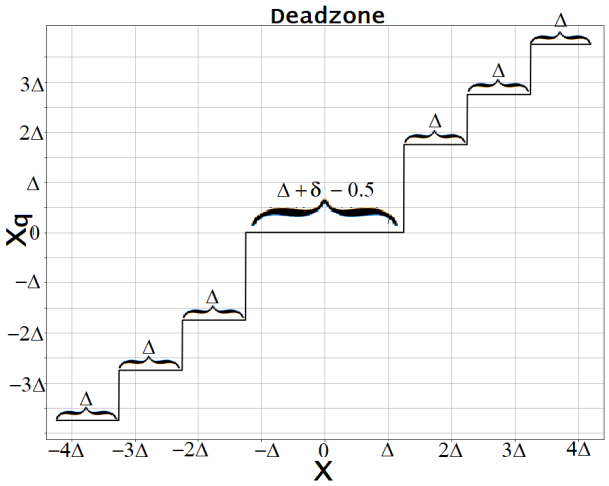


Figura 2.10: Quantização Uniforme com *deadzone*.

Podemos definir δ como o tamanho relativo do *deadzone* em relação ao passo de quantização, a Figura 2.10 ilustra a quantização com zonas mortas. Um fato interessante é que *midrise* e *midtread* são formas de quantização com *deadzone* variando o δ. Para $\delta = 0.5$, temos a quantização *midtread* e, para $\delta = 0.0$, temos a quantização *midrise*.

2.5.3 Quantização não Uniforme

Ao contrário da quantização uniforme, a quantização não uniforme tem intervalos com diferentes tamanhos. Para valores maiores teremos passos de quantizações maiores e, para valores menores, teremos passos de quantizações menores [47]. A vantagem da quantização não uniforme

está em conter intervalos de quantização menores na região que possui maior massa de probabilidade, de forma a diminuir a distorção para essa região [50]. Em contrapartida, haverá um aumento de distorção quando a entrada do quantizador cair nos intervalos mais distantes da origem [50].

As funções logarítmicas podem ser usadas para realizar esse tipo de quantização, pois os passos perto da origem são menores e, ao se distanciarem, serão maiores. A Fig. 2.11 exemplifica como a quantização não uniforme se comporta.

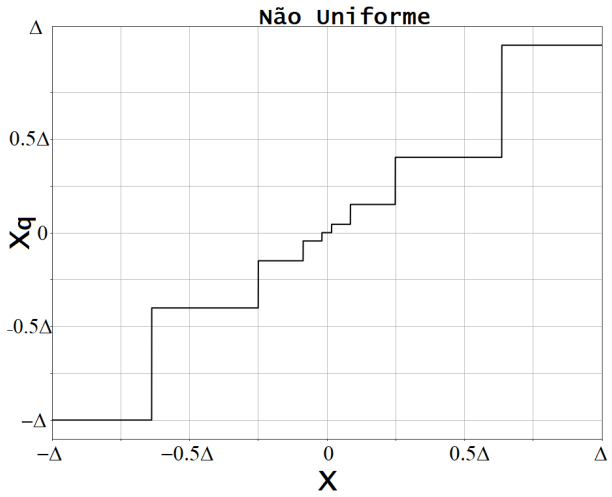


Figura 2.11: Quantização não Uniforme.

2.5.3.1 Quantização Compacta

A quantização compacta (*companded quantization*) tem como base expandir a região em que as entradas possuem alta probabilidade de concentração e essas regiões são próximas da origem [50]. Um quantizador compacto mapeia a entrada através de uma função de compressão, esse novo valor é quantizado e depois reconstruído por uma função de expansão [50]. Mas, se limitarmos a saída de um quantizador não uniforme por um valor X_{max}, esse quantizador também é definido como um quantizador compacto [50].

Duas funções comumente usadas para quantização compactas são a "lei A"e a "lei *mu*", uma vez que essas limitam sua saída para um determinado valor [50]. Neste trabalho, nós utilizamos o tipo de quantização não uniforme denominada “lei μ” [47], [50], [51]. A lei *mu* tem o processo de quantização (ou função de compressão) definido pela fórmula 2.27 e a reconstrução (função de expansão) da lei *mu* é representada pela Equação 2.28. Nas Equações 2.27 e 2.28, temos que X_{max} é o maior valor possível para a entrada, X um valor de peso a ser quantizado, X_q o valor que será passado ao codificador. A variável *mu* é a quantidade de canais existentes, a função $s(X)$ retorna 1 se o X for positivo ou zero e retorna -1 se X for negativo, por fim, $\widehat{X}$ é o valor reconstruído pelo codificador.

$$X_q = X_{max} \frac{\log\left(1 + \mu^{\frac{|X|}{X_{max}}}\right)}{\log(1+\mu)} s(X), \tag{2.27}$$

$$\widehat{X} = \frac{X_{max}}{\mu}\left[(1+\mu)^{\left(\frac{|X|}{X_{max}}-1\right)}\right] s(X). \tag{2.28}$$

2.5.3.2 Ponto Flutuante

A representação de números como ponto flutuante [31] também é considerada uma forma de quantização não uniforme.

O ponto flutuante é a versão (representação) binária da notação científica e é dividida em três partes:

- **Sinal:** um único bit no qual zero representa número positivo e; um, se for negativo.
- **Expoente:** é o valor adicionado ao atual expoente ($expoente - bias$).
- **Mantissa:** é a parte do valor em notação científica.

Os pontos flutuantes podem ser divididos em cinco grupos [31]:

- **Zero:** utilizado para representar o zero, todos os bits da mantissa e expoente são zeros, podendo variar o bit de sinal, existindo a possibilidade de ser +0 ou −0 [31].
- **Infinito:** tem a função de representar valores infinitos, é definido quando os bits do expoente são todos 1's e todos os bits da mantissa são 0's, já para o sinal o bit pode variar formando: $+Inf$ e $-Inf$ [31].
- **Não números:** os bits do expoente são todos 1's e a mantissa terá um valor diferente de 0 [31].
- **Números normalizados:** utilizam um valor de expoente que varia desde de 1 até o maior número possível para o expoente menos 1 [31].
- **Números não normalizados:** possuem um expoente com o valor 0 e a mantissa com um valor diferente de zero [31].

A quantidade total de bits irá influenciar o total de bits usados para expoente e mantissa, além do valor da constante *bias*. O valor de sinal sempre será representado por 1 bit, enquanto a quantidade de mantissa e expoente podem variar, conforme equação abaixo:

$$s_0\ e_0\ e_1 \ldots e_{p-1}\ m_0\ m_1 \ldots m_{q-1}. \tag{2.29}$$

Neste trabalho, o grupo de interesse é o dos números normalizados, os quais possuem a Equação 2.30 para transformar a representação em bits e notação científica.

$$Valor_{float} = (-1)^{s_0} 2^{e-bias}(1+m). \tag{2.30}$$

na Tabela 2.5, apresentamos uma síntese para os principais tipos de ponto flutuante.

Tabela 2.5: Síntese dos pontos flutuantes mais conhecidos.

Nome	Total de bits	Quantidade de bits para expoente (p)	Quantidade de bits para mantissa (q)	Bias
Double	64	11	52	1023
Float	32	8	23	127
Halfprecision	16	5	10	15
Minifloat	8	4	3	3

2.5.4 Entropia

É uma medida do grau de incerteza de uma fonte e é usada aqui para estimar quantos bits um codificador gastaria para codificar cada peso [47]. A entropia H para amostras independentes é:

$$H = -\sum_{i=0}^{n} p_i \log_2(p_i), \tag{2.31}$$

onde n é quantidade total de resultados possíveis e p_i é a probabilidade do i-ésimo resultado possível.

2.6 DISTRIBUIÇÕES

O conhecimento das distribuições estatísticas de amostras é necessário para analisar os efeitos da quantização e para projetar o quantizador [51]. A PDF (*probability density function*, função densidade de probabilidade) de uma variável aleatória X descreve como a distribuição da variável se comporta e mostra como os dados são distribuídos. Além disso, é a derivada da função de densidade cumulativa [52]

2.6.1 Distribuições Populares

2.6.1.1 Gaussiana

A distribuição Gaussiana (ou normal) é uma das distribuições mais comuns usadas e tem seu formato associado a um sino (vide Figura 2.12) [52]. Sua PDF é definida por:

$$f_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{2.32}$$

onde vemos que a PDF depende do desvio padrão (σ) e da média (μ) da variável Gaussiana.

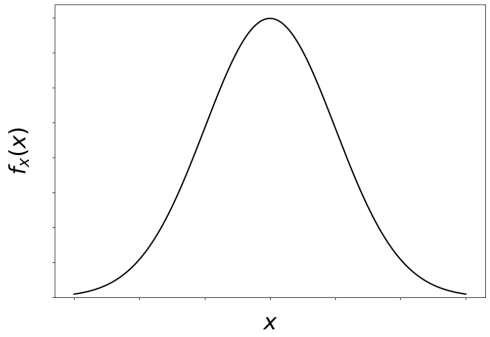


Figura 2.12: PDF Gaussiana.

2.6.1.2 Uniforme

Na distribuição uniforme, a variável será distribuída uniformemente, ou seja: 2.33 [52]:

$$f_x(x) = \frac{1}{B-A}, \text{se } A \leq x \leq B. \tag{2.33}$$

A Figura 2.13 ilustra a PDF de uma variável uniforme, sendo constante no intervalo $[-A, A]$ [52].

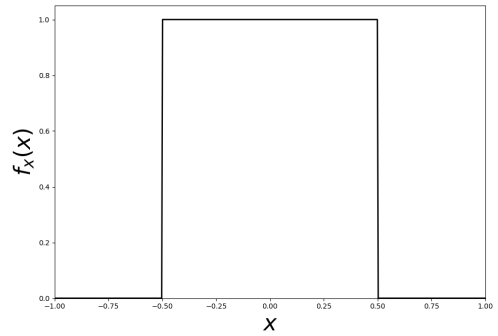


Figura 2.13: PDF uniforme.

2.6.1.3 Cauchy

A distribuição de Cauchy é muitas vezes comparada com a distribuição Gaussiana. Contudo, a distribuição de *Cauchy* possui, em sua PDF, as caudas mais longas e mais planas em relação à

Gaussiana. A Figura 2.14 ilustra a diferença entre essas distribuições [53]. Sua PDF pode ser descrita como:

$$f_x(x) = \frac{\frac{\alpha}{\pi}}{(x-\mu)^2 + \alpha^2}, \tag{2.34}$$

onde μ e α são parâmetros.

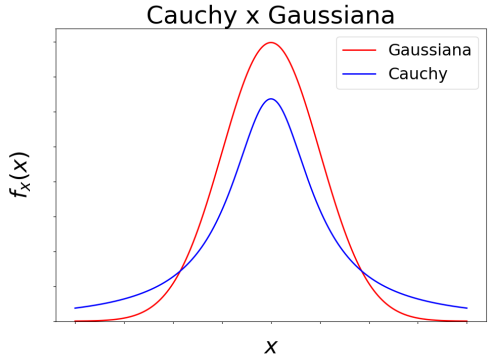


Figura 2.14: Diferença entre a PDF de Cauchy e Gaussiana.

2.6.1.4 Exponencial

A distribuição exponencial é um caso especial da distribuição *chi-square* [51], ilustrada pela Figura 2.15 e descrita por

$$f_x(x) = \lambda e^{-\lambda(x-\mu)}. \tag{2.35}$$

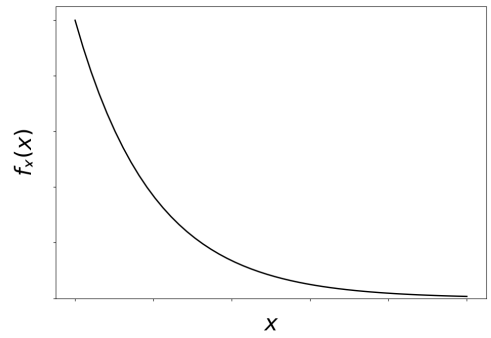


Figura 2.15: PDF Exponencial.

2.6.1.5 Gama

A distribuição gama depende de dois parâmetros α (parâmetro de forma) e β (parâmetro de escala invertida). De acordo com esses dois valores, o comportamento da PDF irá variar (Figura 2.16). A distribuição gama é descrita por [52]:

$$f_x(x) = \frac{(x-\mu)^{\alpha-1}}{\Gamma(\alpha)\beta^{\alpha}} e^{-\frac{x-\mu}{\beta}}, \tag{2.36}$$

$$\Gamma(x) = \int x^{\alpha-1} e^{-x}\, dx = (x-1)!, \tag{2.37}$$

onde $\Gamma(\alpha)$ é a função gama, uma extensão da função fatorial.

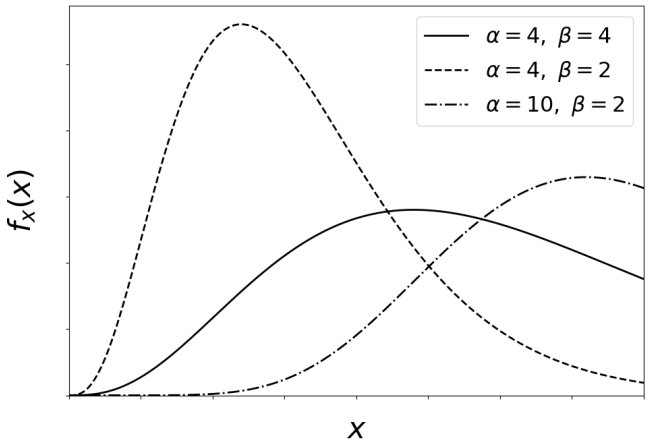


Figura 2.16: PDF gama.

2.6.1.6 Laplaciana

A PDF da distribuição Laplaciana é descrita por

$$f_x(x) = \frac{\alpha}{2} e^{\alpha|x-\mu|}, \tag{2.38}$$

α é parâmetro de escala e μ representa um parâmetro de localização. A distribuição Laplaciana é ilustrada na Fig. 2.17.

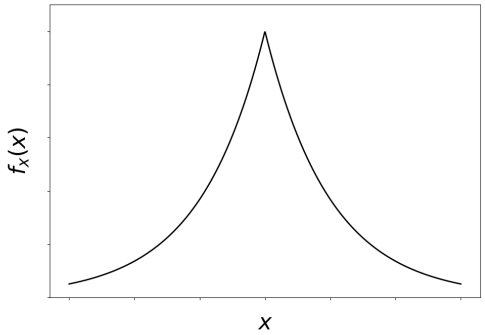


Figura 2.17: PDF Laplaciana.

2.6.1.7 Alfa e Logística

Há duas distribuições menos conhecidas que foram usadas em nosso trabalho [53], [54]. A primeira é a distribuição alfa, definida por

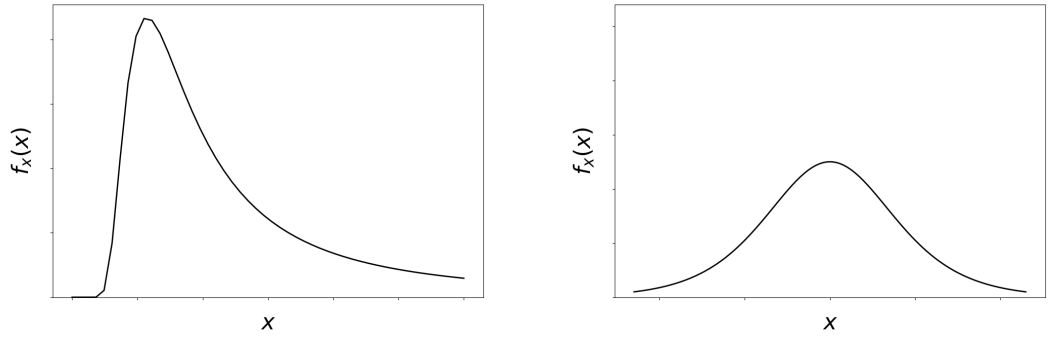
$$f_x(x) = \frac{1}{x^2 \Phi(\alpha)\sqrt{2\pi}} e^{-\frac{1}{2}(\alpha - \frac{\beta}{x})^2}, \tag{2.39}$$

sendo que Φ é a função de distribuição cumulativa normal de α, α e β são os parâmetros de forma e escalabilidade.

A segunda, a distribuição logística, tem sua PDF [53], [55]:

$$f_x(x) = \frac{e^{-x}}{(1 + e^{-x})^2}. \tag{2.40}$$

Pela Figura 2.18(a) vemos a caracterização da distribuição alfa e, na Figura 2.18(b), o formato da PDF da distribuição logística.



(a) PDF alfa. (b) PDF logística.

Figura 2.18: PDF das distribuições alfa e logística.

2.7 QUANTIZAÇÃO ESCALAR ÓTIMA PARA VARIÁVEIS ALEATÓRIAS EXPONENCIAIS E LAPLACIANA

O estudo de Sullivan denominado "*Optimal entropy constrained scalar quantization for exponential and laplacian random variables*" [56] constitui uma das bases teóricas para nosso trabalho. O autor apresenta a solução do quantizador escalar com restrição de entropia para fontes laplacianas e exponenciais [56].

Segundo Sullivan. para a fonte com distribuições exponencial, a quantização ótima é alcançada pela quantização de limiar uniforme (UTQ, *uniform threshold quantizer*) [56]. Já, para as distribuições laplacianas, ele chega à conclusão de que a quantização ótima com restrição de entropia é a quantização uniforme com uma zona morta (*deadzone*) em torno do zero [56].

2.8 SEMELHANÇA ENTRE DISTRIBUIÇÕES

Para analisar a semelhança entre distribuições podemos usar medidas envolvendo a soma de erros [40], como as métricas mostradas na Seção 2.4.2: RMSE (Eq. 2.8), SSE (Eq. 2.9), MSE (Eq. 2.10), SAE (Eq. 2.11) e MAE (Eq. 2.12). Outra métrica ainda não citada é a divergência Kullback-Leibler (D_{KL}). Seja $P(i)$ a distribuição de probabilidade de um primeiro modelo, seja $Q(i)$ a probabilidade de distribuição de um segundo modelo, e D_{KL} a métrica que verifica a perda quando a variável $Q(i)$ é aproximada por $P(i)$ [57], dada por

$$D_{KL}(P||Q) = \sum_{i=1}^{n} P(i) \log \frac{P(i)}{Q(i)}. \tag{2.41}$$

Essa fórmula (Eq. 2.41) mede a semelhança entre duas probabilidades. Um fato importante dessa métrica é o de não ser simétrica, uma vez que não necessariamente $D_{KL}(P||Q)$ é igual $D_{KL}(Q||P)$. Ao analisar seu resultado, percebemos que quanto mais perto D_{KL} é de zero, mais equivalentes serão essas duas distribuições.

2.9 ARTIGOS CORRELATOS

Na literatura, existem diversos trabalhos sobre compressão de RN's, no entanto, a maioria envolve retreino do modelo ou uma mudança da estrutura do mesmo (por exemplo, inserção ou remoção de camadas) para realizar a compressão. Embora menos abundantes, os trabalhos de compressão sem retreino são os que mais nos interessam.

2.9.1 Compressão com Retreino

Entre os trabalhos que envolvem retreino, destacam-se os métodos que utilizam *pruning* [16]–[19], [58], quantização de camadas [59], compartilhamento de pesos [20] e quantização em geral [21].

2.9.1.1 *Pruning*

No trabalho de Han et al. [16], primeiro é realizado um treinamento da rede para descobrir as conexões importantes, cortando as não importantes. A atividade posterior foi realizar a quantização dos pesos, de modo que esses fossem compartilhados com outras camadas. Depois das atividades de *pruning* e quantização, a rede é retreinada para afinação dos pesos. Os autores reduziram os tamanhos das redes VGG e AlexNet em 49× e 35×, respectivamente, com pequena perda de acurácia [16].

Dong et al. [58] buscaram desenvolver um método de realizar o corte(*pruning*) em parâmetros, embasados nas informações da derivada de segunda ordem da função de erro de cada camada. Os

autores relatam ser possível chegar a uma taxa de compressão (definida por eles como a proporção dos números de parâmetros preservados em relação aos parâmetros originais) de 11% para AlexNet sem perdas, após retreinamento. Para VGG, há uma taxa de redução de pesos de 7.5% ocasionando uma perda na acurácia de aproximadamente 0,5% após retreinamento [58].

Li et al. [17], na etapa de treinamento, usam a medida do segundo momento no otimizador de Adam [60] para avaliar a relevância de cada peso na rede. Em seguida, com cada peso com sua importância realizam um corte baseado em limiar calculado automaticamente. Os resultados obtidos são: um corte de 12,5% dos pesos para AlexNet com menos de 1% de prejuízo à acurácia [17].

Zhao et al. [18] desenvolveram duas funções de lógica nebulosa (*fuzzy*), uma para a "importância"e a outra para "não-importância" dos pesos. Os pesos importantes são retreinados, os "não importantes" são cortados, através de um algoritmo denominado α*-cut* com α aumentado progressivamente. Esse método consegue reduzir o tamanho de armazenamento da rede *VGG-16* em 73% e *VGG-19* em 77%, com resultados similares ao original [18]. Por fim, Serra et al. [19] desenvolveram um algoritmo chamado de *Lossless Expressiveness Optimization* que encontra as camadas e unidades da RN que podem ser removidas após reparametrização [19].

2.9.1.2 Quantização em camadas

Na quantização feita separadamente para cada camada, um trabalho relevante é o de Zhu et al. [59], que visa realizar a quantização da rede com diferentes larguras de bits associados para diferentes camadas. Nesse caso, cada camada terá sua quantização uniforme acoplada à estrutura da rede e, por esse motivo, participam do treinamento da rede. Os resultados mostram uma queda maior que 10% da acurácia tanto Top 5 e Top 1 para a rede *AlexNet* com uma taxa de compressão 10,2× [59].

2.9.1.3 Compartilhamento de pesos

Kim et al. [20] propõem dois modos de realizar a compressão: *pruning* e compartilhamento de peso da RN LeNet. As etapas desse trabalho são: remover os pesos "pequenos"; retreinar a rede; agrupar os pesos com valores representativos; retreinar a rede, novamente. Ao final, na rede *LeNet* conseguiram reduzir a quantidade de parâmetros de 430500 para 32, preservando a acurácia original (0,01% de diferença) [20].

2.9.1.4 Quantização geral

Outra forma de comprimir RN's é quantizando os pesos durante o treinamento da rede, aproximando esses pesos para um limitado *codebook* de entrada. Esse é o principal ponto do trabalho de Faraone et al. [21]. Em seus resultados, a RN *AlexNet* obteve detrimento de até 1% no funcionamento da rede, a *VGG* com a acurácia (tanto Top 1 e Top 5) piorando aproximadamente 1% e a *ResNet* [61], com prejuízo de um pouco mais de 1% [21].

2.9.2 Compressão sem Retreino

Os artigos que envolvem retreino se diferenciam do presente trabalho, uma vez que esse procura não usar retreino, o que demandaria mais recursos computacionais e mais tempo. Nesse contexto, o método de compressão sem retreino é mais relevante. Das pesquisas de compressão sem retreino podemos destacar três métodos diferentes: compressão com agrupamento [62], compressão com compartilhamento de pesos [63] e, também, quantização escalar [64].

Seo e Kim usam um método híbrido com compressão uniforme seguido de agrupamento por *K-means* para comprimir a rede *AlexNet*. Tanto para acurácia Top 1 e Top 5 conseguem chegar à quantização com 32 níveis, com uma perda de 0,5% para ambas métricas (acc_1 e acc_5) [62].

Dupuis et al. [63] empregam compressão por compartilhamento de pesos entre as camadas com a necessidade de realizar o agrupamento para cada camada da rede. Vale ressaltar que os autores usam modelos no formato ONNX. No final, conseguem chegar a uma taxa de compressão (definida como a proporção entre o tamanho do modelo original e uma aproximação do que seria o tamanho do modelo desenvolvido) por volta de 5×, com uma perda de menos de 1% na acurácia Top 1 para as redes *ResNet* e *Squeezenet* [63]. A pesquisa de Dupuis et al. não verifica a métrica Top 5 para seus resultados.

Em Haase et al. [64], por sua vez, os parâmetros passam por uma quantização escalar dependente ou quantização *trellis-coded*. Depois da quantização, Haase et al., explicam como é feita a codificação entrópica, fundamentada no codificador *DeepCABAC*. Nos experimentos, foram usadas as redes *VGG*, *ResNet*, *MobileNet* (classificação de imagem), *DCase* (classificação de aúdio) e *UC12B* (*autoenconder* de imagem). Nessas redes, os autores obtiveram uma taxa de compressão $C = 0,118$, sendo $C = \frac{R_C}{R_O}$, onde R_C é um número de bits comprimidos e R_O o número de bits não comprimidos, contudo, as redes tiveram um prejuízo no desempenho, em média, de 0,37% na acurácia [64].

A pesquisa em andamento difere da literatura acima apresentada nos seguintes aspectos: realiza um estudo maior sobre os pesos e sua distribuição, compara diferentes métodos de quantização, utiliza maior número de redes. Por outro lado, assemelha-se a Dupuis [63] ao usar modelos RN no formato ONNX.

2.9.3 MPEG-NNR

Sobre a compressão de redes neurais, encontramos a chamada do MPEG para compressão de redes neurais, tal iniciativa tem o nome de MPEG–NNR [65]. O objetivo do NNR é definir uma representação comprimida, interpretável e interoperável para redes neurais treinadas [65]. Para alcançá-lo, a nova representação deve ser capaz de retratar diferentes tipos de redes neurais (LSTM - memória de curto e longo prazo, CNN – redes neurais convolucionais, RNN – redes neurais recorrentes e outras) [65]. Deve, também, possibilitar incrementar redes neurais e modificá-las, ensejar a escalabilidade dos modelos, ser possível inferir a rede comprimida e, também, possibilitar o uso

com recursos limitados dessas redes [65].

Os formatos de intercâmbio como ONNX (seção 2.3) e NNEF (*Neural Network Exchange Format*) são recomendados pelo MPEG-NNR para realizar uma representação comprimida de redes neurais [65]. Em seguida, o documento do MPEG-NNR mostra vários usos e a visão geral dos requisitos para cada um.

Em março de 2019, houve uma chamada de proposta com os seguintes requisitos [65]: representação eficiente do modelo (O tamanho do modelo comprimido tem que ser pelo menos 30% menor do que o modelo original); suportar diferentes tipos de redes neurais (CNN, RNN e outros); a representação comprimida contendo todos os parâmetros e pesos da rede neural; realizar a inferência do modelo comprimido; o método para comprimir a rede neural independente do conjunto de dados usado para treinar o modelo original; baixo poder computacional e consumo de memória da realizar a decodificação [65].

3 QUANTIZAÇÃO DE PESOS

Esse trabalho busca estabelecer um método, sem utilizar retreinamento, para realizar a compressão dos pesos (a partir desse momento pesos podem referenciar tanto aos pesos como aos vieses) das RN's, de forma que não implique em uma degradação do funcionamento da mesma. Assim, procura-se diminuir as taxas (quantidade de representações) sem perda expressiva de precisão na saída da RN. Os pesos no formato ONNX são representados, geralmente, por *floats* que possuem 32 bits. Essa quantidade de bits permite um total de mais de quatro bilhões de representações. Baseando-se em técnicas de quantização, visa-se diminuir as quantidades de representações que são necessárias para constituir os pesos de um modelo. Um caso comum de compressão é a feita em imagens que utilizam a quantização como forma de diminuir os níveis utilizados.

Ainda serão exploradas as diferentes formas utilizadas para comprimir as RN's, em especial, os pesos e vieses presentes nas RN's. O método da pesquisa consiste em duas grandes etapas, apresentadas na Figura 3.1:

Figura 3.1: Visão geral das principais etapas do trabalho.

- **Comparação entre distribuições:** a distribuição do modelo é contraposta com distribuições conhecidas.
- **Quantização e resultados:** cada peso do modelo passa por diferentes tipos de processo de quantização com menos níveis. Em seguida, é realizada a comparação entre os resultados que foram obtidos.

3.1 MODELOS DE REDES NEURAIS

Antes de apresentar a comparação entre a função densidade probabilidade dos pesos dos modelos com as distribuições de referência, serão apresentados os modelos de RN's utilizados em todas etapas da pesquisa. Os modelos em questão estão todos no formato ONNX [15] e são conhecidos como arquiteturas de RN populares e são do tipo CNN. A Tabela 3.1 apresenta as informações relativas a esses modelos. Nela, podemos ver o tamanho do arquivo ONNX referente a cada modelo,

assim como a porcentagem representando o quanto do arquivo é composto por pesos e vieses, bem como vemos qual é a aplicação daquela RN.

Tabela 3.1: Modelos ONNX e informações básicas.

Modelo	Tamanho do arquivo (Em MB)	Razão percentual de pesos e vieses no arquivo (%)	Tipo
VGG19-bn-7 [36]	548,15	99,997	Classificação de imagem
caffenet [66]	232,57	99,999	Classificação de imagem
bvlcalexnet-9 [34]	232,57	99,999	Classificação de imagem
rcnn-ilsvrc [67]	220,06	99,999	Classificação de imagem
resnet101-v2-7 [61]	170,40	99,930	Classificação de imagem
efficientnet-lite4 [39]	49,54	99,841	Classificação de imagem
Densenet [68]	31,2	99,587	Classificação de imagem
Lenet [35]	26,72	99,907	Classificação de imagem
Age - Lenet [69]	22,85	99,862	Classificação de idade
Gender - Lenet [69]	22,83	99,861	Classificação de gênero
Mobilenet [33]	13,59	99,366	Classificação de imagem
Shufflenet [38]	5,46	99,246	Classificação de imagem
Squezenet [37]	4,73	99,713	Classificação de imagem

3.2 RELAÇÃO E DEPENDÊNCIAS ENTRE OS PESOS

Uma das formas de verificarmos qual a melhor maneira de realizar a quantização de dados é procurarmos se eles possuem algum tipo de padrão e dependência entre si. A alta dependência dos dados nos indica que a quantização vetorial poderia ser a melhor forma de realizar a quantização. Essa verificação pode ser feita por: correlação, análise espectral dos dados e autocorrelação dos dados.

Na presente pesquisa, para verificarmos a dependência dos dados temos que verificar os pesos e vieses do modelo da rede neural. Porém, para verificarmos se existe dependência entre os pesos, há que se estabelecer uma ordem para eles, os pesos são separados em camadas (as quais podemos ordenar desde a camada de entrada até a camada de saída), mas dentro das camadas não podemos encontrar uma ordem entre os pesos, a priori. Assim, sem uma ordenação intrínseca entre os pesos e vieses de uma camada não há razão para procurar correlação entre os pesos. Do mesmo modo, como não podemos determinar uma sequência entre os pesos, não podemos analisar a autocorrelação e a análise espectral.

Desse modo, a única forma de verificarmos qualquer dependência entre os dados de uma rede neural, sem ordenamento definido, é calculando a autocovariância. Porém, para a rede neural *sque-*

ezenet (4,73 MB), o total de coeficientes de autocovariância seria aproximadamente 22,4 MB, tarefa que exige um grande poder computacional e tempo. Por esse motivo, não foram calculados os coeficientes de autocovariância.

Portanto, como não podemos ter evidências que exista algum tipo de dependência (ou relação) entre os pesos, não podemos indicar a quantização vetorial para os pesos ou a transformação dos dados. Por isso, optamos pelo uso da quantização escalar.

3.3 COMPARAÇÃO DA FUNÇÃO DENSIDADE DE PROBABILIDADE

Após analisarmos os modelos usados e percebermos que a maioria do arquivo ONNX é formado por pesos dos modelos de IA, passamos para a etapa em que buscamos caracterizar a distribuição de amplitude. Essa fase é fundamental, uma vez que podemos nos valer das distribuições conhecidas para utilizar métodos de quantização eficientes para determinadas distribuições. Além, podemos usar outras informações que se tornam visíveis ao analisar uma distribuição (se os dados se concentram mais perto do centro, se tem comportamento simétrico, por exemplo). O fluxo desse estágio é apresentado na Figura 3.2.

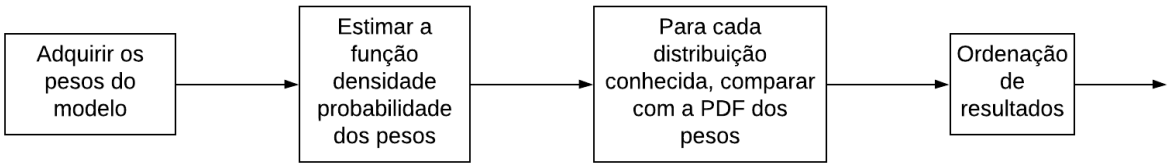


Figura 3.2: Passos para a comparação da PDF com distribuições conhecidas.

3.3.1 PDF dos pesos

Para estimar a PDF dos pesos utilizamos um simples histograma. Comparamos a estimação com um número de funções de distribuições populares como: exponencial (2.6.1.4), uniforme (2.6.1.2), Cauchy (2.6.1.3), Laplace (2.6.1.6), alfa (2.6.1.7), logística (2.6.1.7), gama (2.6.1.5) e Gaussiana (2.6.1.1). Para quantificar e medir qual distribuição mais se aproxima da distribuição dos pesos, recorreu-se a algumas medidas de dispersão, para que possa ser possível medir a semelhança entre cada distribuição e a do modelo de ML. Para tanto, foram utilizadas a D_{KL} e SSE, explicadas nas Seções 2.8 e 2.4.2.1, respectivamente. Com os resultados para cada uma das duas métricas, esses são ordenadas a fim de que seja possível determinar qual distribuição poderia ser usada para modelar os pesos de uma RN.

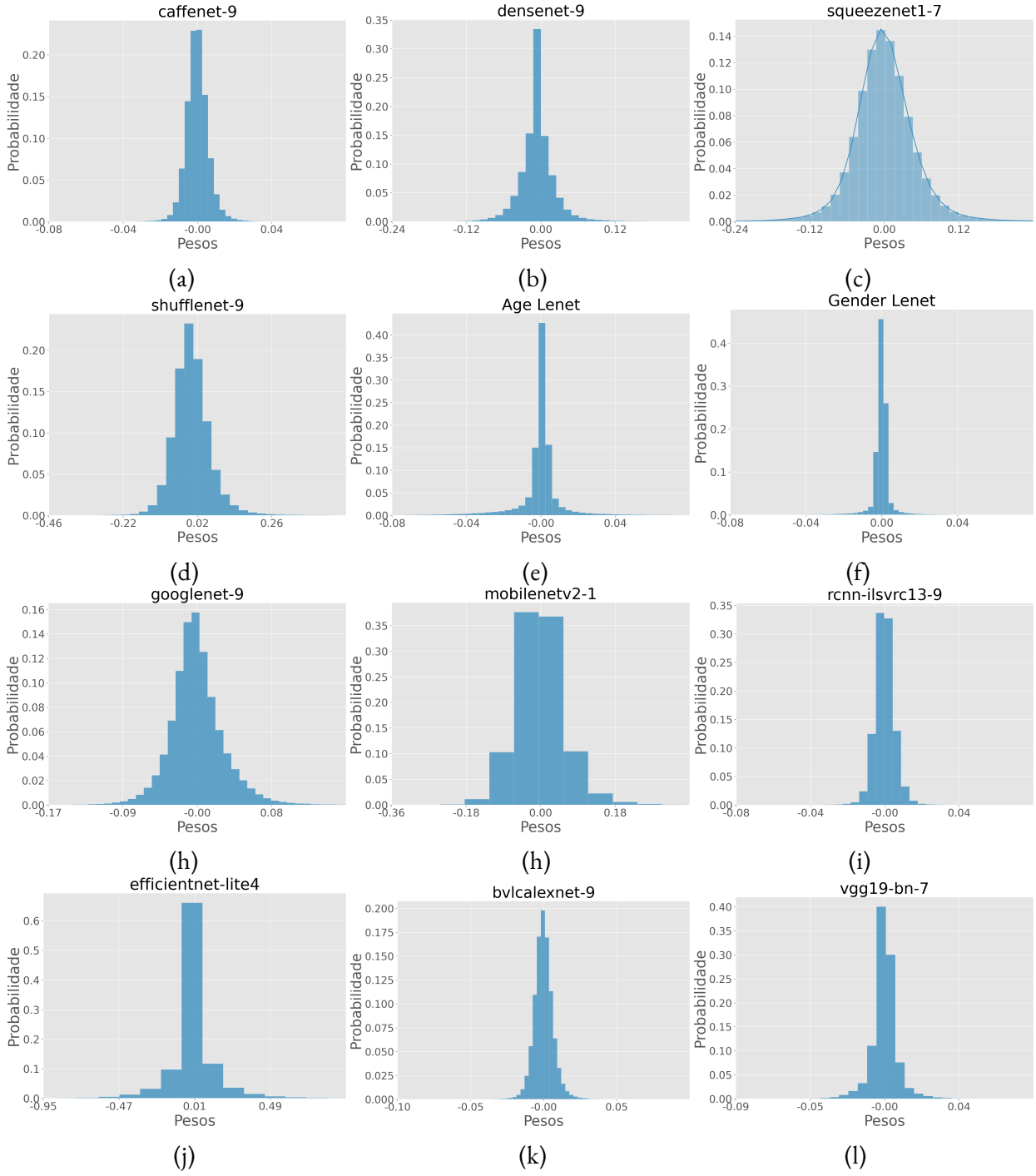


Figura 3.3: Histograma dos modelos.

3.4 ANÁLISE DE SIMILARIDADE DE DISTRIBUIÇÕES

Os histogramas são usados para estimar a PDF de cada modelo e são apresentados na Figura 3.3. A Figura apresenta os dados centralizados na média e o eixo y (Pesos) é limitado onde a maioria dos dados se concentram. Ao observar os histogramas, percebemos que os pesos, em sua maioria, se concentram em torno de 0 (para todos os modelos mostrados na Figura 3.3).

3.4.1 Comparação entre as distribuições conhecidas e PDF's dos modelos

Para cada um desses modelos, houve a comparação da função de densidade de probabilidade original em busca das distribuições conhecidas com maior semelhança. Para averiguar o resultado da similaridade foram usadas a SSE (Seção 2.4.2.1) e a D_{KL} (Seção 2.8).

Tabela 3.2: SSE entre distribuições padrões e a distribuição de cada modelo.

Redes	1^a **dist. semelhante**	2^a **dist. semelhante**	3^a **dist. semelhante**
Caffenet	Laplaciana	Logística	Gaussiana
Alexnet	Alfa	Laplaciana	Logística
Densenet	Laplaciana	Alfa	Gaussiana
efficientnet	Laplaciana	Logística	Gaussiana
GoogLenet	Laplaciana	Logística	Gaussiana
Age - googLenet	Logística	Laplaciana	Cauchy
Gender - googLenet	Laplaciana	Logística	Cauchy
Mobilenet	Laplaciana	Gaussiana	Logística
Rcnn	Gaussiana	Laplaciana	Alfa
Zfnet	Laplaciana	Logística	Gaussiana
Resnet	Laplaciana	Logística	Gaussiana
Shufflenet	Cauchy	Laplaciana	Logística
Squeezenet	Logística	Laplaciana	Gaussiana
VGG	Alfa	Laplaciana	Logística

A Tabela 3.2 sintetiza os resultados das comparações, com o objetivo de mostrar as distribuições que mais se assemelham à PDF de cada modelo. A coluna 1^a *dist. semelhante* contém a distribuição com o melhor resultado. Já, as colunas 2^a *dist. semelhante* e 3^a *dist. semelhante* referem-se à segunda e à terceira distribuição mais similar aos pesos do modelo, respectivamente. A distribuição Laplaciana é a melhor distribuição, de acordo a métrica SSE, para 57,14% e a segunda melhor para 42,86% dos modelos da Tabela 3.2. Em comparação, vemos que a distribuição logística é a melhor distribuição para dois modelos, a segunda melhor em seis oportunidades e a terceira melhor em 4 oportunidades.

A Tabela 3.3 condensa os resultados para a métrica DKL buscando a primeira, a segunda e terceira distribuição mais similar à PDF dos pesos, segundo a métrica D_{KL}. A Laplaciana, novamente, possui bons resultados com 42,86% dos casos, sendo a primeira distribuição mais similar (1^a *dist.*

Tabela 3.3: D_{KL} entre distribuições padrões e a distribuição de cada modelo.

Redes	1^a **dist. semelhante**	2^a **dist. semelhante**	3^a **dist. semelhante**
Caffenet	Gaussiana	Laplaciana	Cauchy
Alexnet	Cauchy	Alfa	Laplaciana
Densenet	Cauchy	Laplaciana	Alfa
Efficientnet	Laplaciana	Logística	Gaussiana
GoogLenet	Laplaciana	Logística	Gaussiana
Age - googLenet	Cauchy	Laplaciana	Logística
Gender - googLenet	Laplaciana	Cauchy	Logística
Mobilenet	Laplaciana	Logística	Alfa
Rcnn	Gaussiana	Laplaciana	Alfa
Zfnet	Alfa	Laplaciana	Logística
Resnet	Laplaciana	Logística	Gaussiana
Shufflenet	Cauchy	Laplaciana	Logística
Squeezenet	Laplaciana	Logística	Gaussiana
VGG	Alfa	Laplaciana	Logística

semelhante). E, em 50% dos modelos é a segunda distribuição mais similar (2^a *dist. semelhante*).

Com base na informação de que a distribuição de Laplace é uma boa aproximação, conforme as métricas SSE e D_{KL}, podemos utilizar a conclusão de Sullivan. O autor relata que para distribuições *Laplacianas*, a quantização ótima é aproximada pela quantização uniforme com zonas mortas (*deadzone*) [56]. Por isso, neste trabalho nos concentramos em quantização uniforme.

3.5 QUANTIZAÇÃO E COMPARAÇÃO DE RESULTADOS

O resultado de Sullivan diz que para distribuições Laplacianas, a quantização ótima é aproximada pela quantização uniforme com *deadzone* [56]. Aliado a essa perspectiva, podemos chegar a última etapa do trabalho: a quantização dos pesos e a avaliação do seu efeito no funcionamento da rede. Para isso, realizou-se a comparação entre o resultado da rede quantizada e o resultado esperado (*ground truth*). No total, são utilizados cinco métodos de quantização, dando um maior destaque às quantizações uniformes. Todos os pesos são quantizados e desquantizados antes da avaliação da rede.

A seguir, apresentam-se os tipos de quantizações usados, a variação dos níveis e do intervalo (máximo – mínimo) e X_{max} para quantização não uniforme lei μ.

- Quantização uniforme *midrise* (Eqs. 2.23 e 2.24), variando os níveis e o intervalo:
  - **intervalo**:
    - **máximo**: 1

        * **mínimo**: -1
    - **nível**: $2^i, 2 \leq i \leq 24$

- Quantização uniforme *midtread* (Eqs. 2.21 e 2.22), variando os níveis e o intervalo:
    - **intervalo**:
        * **máximo**: 1
        * **mínimo**: -1
    - **nível**: $2^i, 2 \leq i \leq 24$

- Quantização não uniforme, baseada na lei μ, com as seguintes especificações:
    - $X_{max} = 1$;
    - **quantidade de níveis**: $2^i, 2 \leq i \leq 24$.

- Quantização *deadzone*, variando o passo de *deadzone* (∂), níveis e o intervalo (máximo – mínimo), conforme Equações 2.25 e 2.26. A variação de ∂ é descrita na Tabela 3.4.

Tabela 3.4: Quantização *deadzone*: ∂ e os níveis utilizados.

∂	Níveis	Máximo	Mínimo
0.1	$2^i, 2 \leq i \leq 24$	1	-1
0.25	$2^i, 2 \leq i \leq 24$	1	-1
0.4	$2^i, 2 \leq i \leq 24$	1	-1
0.7	$2^i, 2 \leq i \leq 24$	1	-1

- Quantização *minifloat*, com o uso da representação IEEE *halffloat* descrita na Tabela 2.5, além das representações derivadas a partir da equação que são apresentadas na Tabela 3.5. As representações que foram derivadas (Tabela 3.5) são pontos flutuantes simplificados composto apenas pela parte normalizada. As partes não normalizada, *Not a Number*, infinito e zeros não foram representadas.

  A Tabela 3.5 mostra os pontos flutuantes que foram definidos para esse trabalho e a quantidade de bits usados para expoente e mantissa.

3.6 *DATASET* E CONDIÇÕES DE TESTES

Na última etapa da pesquisa, as novas RN's comprimidas passam pela validação de seus resultados. Neste trabalho, foram utilizados dois *dataset's* para avaliar o funcionamento das redes, o conjunto de dados escolhidos depende do tipo aplicação do modelo. O conjunto de dados do *ImageNet Large Scale Visual Recognition Challenge* 2012 (ILSVRC 2012) [43] foi utilizado para os modelos que classificam imagens. Entretanto, o conjunto de dados *Adience Benchmark* [70] foi

Tabela 3.5: Representações de pontos flutuantes usadas.

Bits	Bits Sinal	Bits Expoente	Bits Mantissa	Representação
12	1	4	7	$s_0\ e_0\ e_1\ e_2\ e_3\ m_0\ m_1\ m_2\ m_3\ m_4\ m_5\ m_6$
10	1	3	6	$s_0\ e_0\ e_1\ e_2\ m_0\ m_1\ m_2\ m_3\ m_4\ m_5$
8	1	3	4	$s_0\ e_0\ e_1\ e_2\ m_0\ m_1\ m_2\ m_3$
8	1	4	3	$s_0\ e_0\ e_1\ e_2\ e_3\ m_0\ m_1\ m_2$
7	1	3	3	$s_0\ e_0\ e_1\ e_2\ m_0\ m_1\ m_2$
7	1	2	4	$s_0\ e_0\ e_1\ m_0\ m_1\ m_2\ m_3$
6	1	2	3	$s_0\ e_0\ e_1\ m_0\ m_1\ m_2$
6	1	1	4	$s_0\ e_0\ m_0\ m_1\ m_2\ m_3$
5	1	2	2	$s_0\ e_0\ e_1\ m_0\ m_1$
5	1	1	3	$s_0\ e_0\ m_0\ m_1\ m_2$
4	1	1	2	$s_0\ e_0\ m_0\ m_1$
4	1	0	3	$s_0\ m_0\ m_1\ m_2$

usado tanto para classificação de gênero quanto para classificação de faixa etária. A Tabela 3.6 sintetiza as informações dos *dataset's* e ilustra a quantidade de dados usados.

Tabela 3.6: Informações do tipo de RN's e o *dataset* utilizado.

Tipo	Saída do Modelo	*Dataset*	Quantidade
Classificação de Imagem	1000 classes de imagens	ILSVRC 2012 [43]	50000
Classificação de Gênero	Homem ou mulher	*Adience Benchmark* [70]	26580
Classificação de Idade	8 faixas etárias	*Adience Benchmark* [70]	26580

Como exposto na Tabela acima, o *dataset* ILSVRC 2012 é composto por 50000 imagens divididas em 1000 classes, essas variam desde limão até cadeira de balanço, passando por outras 998 classes. A Figura 3.4 mostra alguns exemplos retirados entre as 50000 imagens. Por outro lado, o *Adience Bench* possui 26580 fotos, referentes a 2284 pessoas, com a informação acerca do sexo e faixa etária de cada pessoa. As idades são divididas em:

- idade entre 0 e 2 anos
- idade entre 2 e 4 anos
- idade entre 8 e 13 anos
- idade entre 15 e 20 anos
- idade entre 25 e 32 anos
- idade entre 38 e 43 anos

- idade entre 48 e 53 anos
- Acima de 60 anos.

A Figura 3.5 apresenta três exemplos retirados do dataset *Adience Benchmark*.

Figura 3.4: Exemplos de imagens do *dataset* ILSVRC.

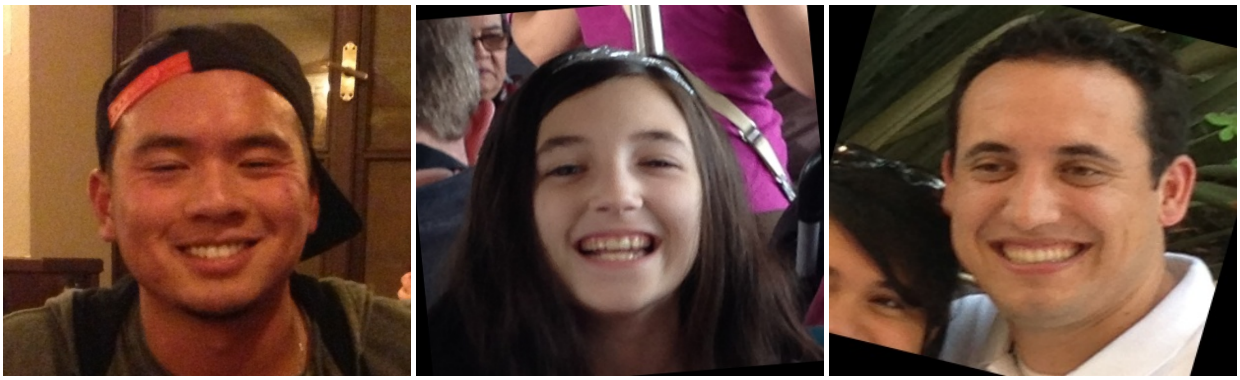


Figura 3.5: Exemplos de imagens do *dataset Adience Benchmark*.

Com o objetivo de realizar os testes, além de escolher o *dataset* correto (Tabela 3.6), foram definidos para cada tipo de RN's qual das métricas apresentadas na Seção 2.4 devem ser escolhidas, essas são importantes para medir a qualidade da rede, determinando se houve ou não prejuízo nos modelos de ML comprimidos.

- **Classificação de Imagem**: Acurácia Top 1 (acc_1) e Acurácia Top 5 (acc_5), ver Seção 2.4.4. As métricas erro Top 1 e Top 5 são exatamente o oposto da acurácia, por isso serão suprimidas.
- **Classificação de idade**: Acurácia (Seção 2.4.1).
- **Classificação de gênero**: Acurácia, *recall*, *precisão* e *f1-score* (Seção 2.4.1).

3.7 COMPARAÇÕES ENTRE OS RESULTADOS

Para a fase de comparação nem todas as redes citadas na Tabela 3.1 foram usadas, devido à dificuldade em realizar a etapa da validação dos modelos *Alexnet*, *Densenet*, *ZFnet* e *Rcnn*. Entretanto, as Tabelas 3.7 e 3.8 expõem os modelos que foram usados na comparação de resultados, assim como exibe a entropia do modelo padrão e os resultados originais das métricas (conforme a Seção 3.6).

Tabela 3.7: Resumo dos modelos de classificação de Gênero e Idade.

Modelo	Entropia	Acurácia	*Recall*	Precisão	F1-Score
Gender - Googlenet	22,439	82,379	0,921	0,786	0,848
Age - Googlenet	22,459	55,189	-	-	-

Tabela 3.8: Resumo dos modelos de classificação de imagem.

Modelo	Entropia	Acurácia Top 1	Acurácia Top 5
Squeezenet	20,221	53,77	77,138
Shufflenet	20,391	42,422	68,134
Googlenet (ILSVRC)	22,653	67,774	88,34
Efficientnet	23,513	77,734	93,684
Caffenet	25,213	56,264	79,522
Mobilenet	21,673	69,3	88,934
Resnet	24,734	77,214	93,614
VGG	25,661	91,816	73,646

Ao averiguar os 10 modelos presentes nas Tabelas 3.7 e 3.8, temos que a média das entropias é 22,896 bits/pesos (bpp). A unidade para verificar a entropia é bits por pesos ou *bpp*. Os gráficos, a seguir, são de taxa (entropia dos pesos quantizado, em bits/pesos) e distorção (acurácia do modelo ou outras métricas relacionadas). As legendas das próximas figuras têm o seguinte significado:

- **MIDRISE**: Corresponde à quantização uniforme com *midrise*.
- **MIDTREAD**: Corresponde à quantização uniforme com *Midtread*.
- **NON-UNIFORM**: Corresponde à quantização não uniforme, baseado na lei μ.
- **MINIFLOAT**: são as quantizações pelas representações de pontos flutuantes.

Para as figuras que comparam o resultado *deadzone*, as legendas numéricas correspondem ao δ usado para variar o tamanho da zona central. Nos gráficos das próximas seções, os valores das métricas (seja acurácia, *f1-score*, *recall*, precisão, acurácia Top1 ou acurácia Top5) serão representados por uma linha horizontal pontilhada.

3.7.1 Uniforme × Não Uniforme

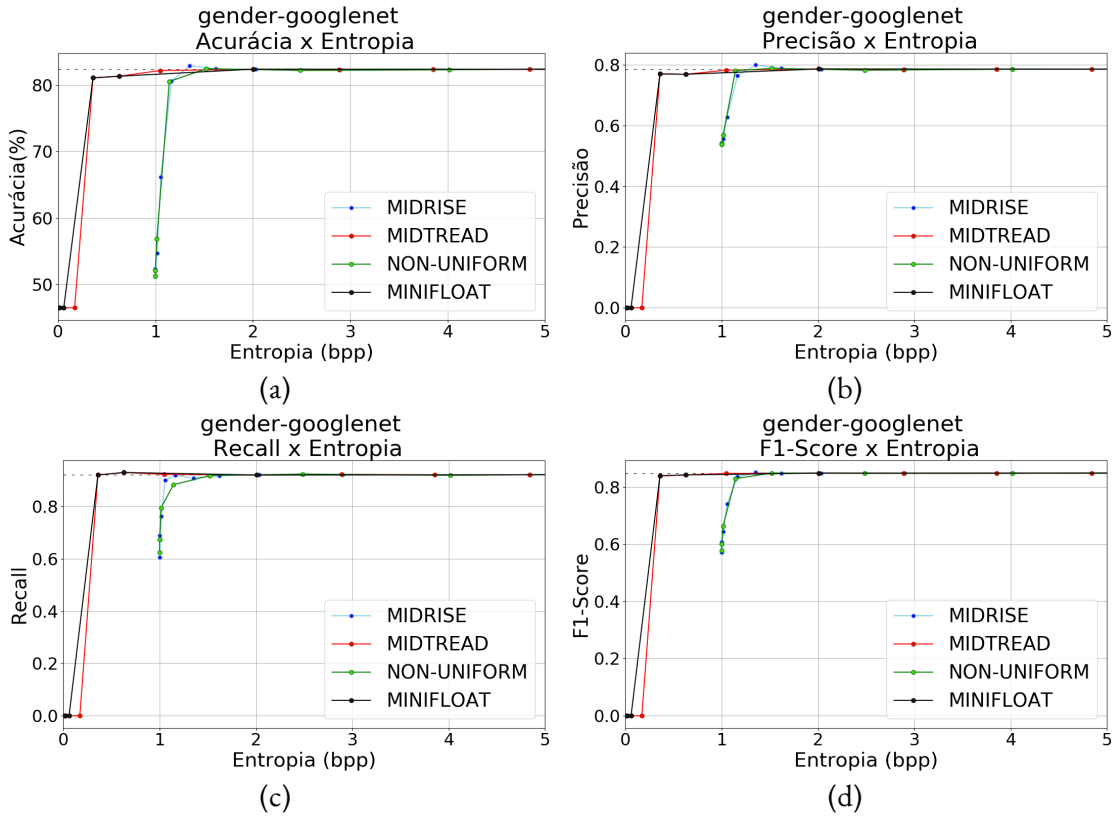


Figura 3.6: Comparações de taxa e distorção (entropia × acurácia) entre quantização uniforme e não uniforme para redes de classificação de gênero.

Para o modelo de classificação de gênero, *Gender - Lenet*, temos os resultados na Figura 3.6. Ao analisá-la, podemos perceber que é possível chegar em torno de 2 bits/pesos, com comportamentos similares para as 4 métricas usadas para *Gender - Googlenet*. Também, nessa figura, vemos que os melhores métodos são *minifloat* e *midtread*.

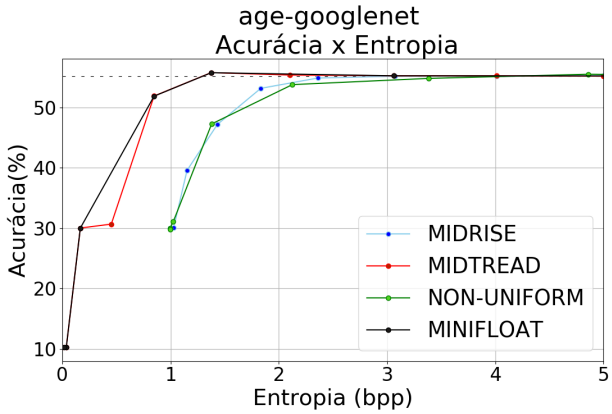


Figura 3.7: Comparações de taxa e distorção (entropia × acurácia) entre quantização uniforme e não uniforme para redes de classificação de idade.

No modelo *Age - Googlenet*, novamente, consegue-se chegar em taxas perto de 2 bpp, mantendo a acurácia próxima ao valor original, conforme a Figura 3.7. Inclusive, nota-se que para os métodos *midtread* e *minifloat* acontece um leve aumento na acurácia ao baixar as taxas, o que não é usual, embora possível. Ao comparar os métodos, vemos que a *midtread* e *minifloat* possuem os melhores resultados, sendo o comportamento dos dois quantizadores similares.

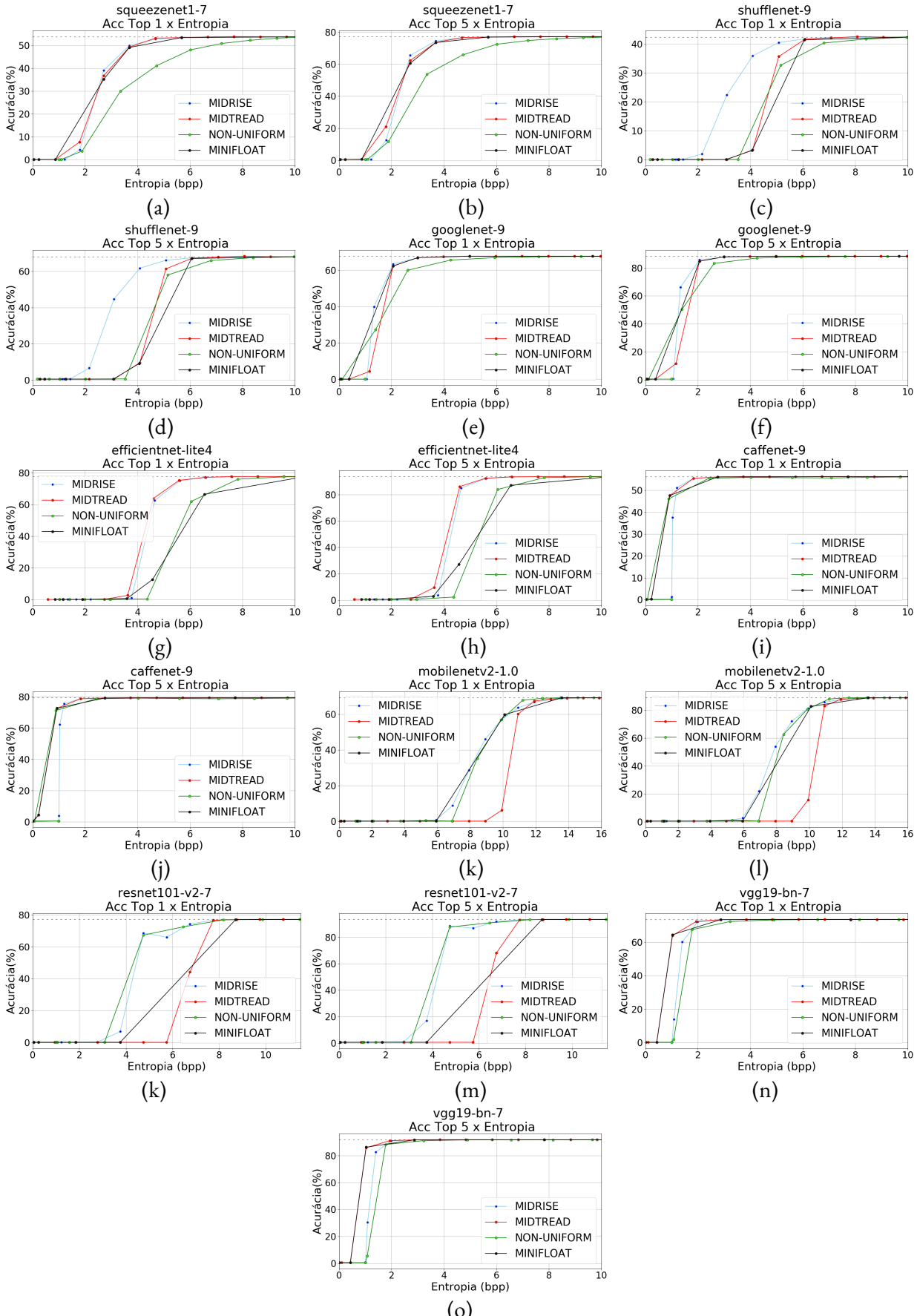


Figura 3.8: Comparações taxa e distorção (entropia × acurácia) entre quantização uniforme e não uniforme para redes de classificação de imagem.

A Figura 3.8 contém todos os resultados e compara os métodos não uniforme e uniforme dos modelos que classificam imagens.

Ao observar os gráficos, não se nota um método que se sobressaia para todas as redes. Na *Squeezenet*, vemos melhores resultados para *midtread* e *midrise* (o método *minifloat* também possui um bom desempenho), alcança uma entropia menor que 5 bits/pesos, com resultados similares ao original. Na RN *Shufflenet*, temos um bom desempenho produzido pela quantização *midrise*, pois possibilita uma entropia ao redor de 5 bits, sem prejuízo ao funcionamento da rede. Para *GoogLenet*, os métodos *minifloat* e *midrise* têm os melhores resultados e conseguem ter entropia perto de 2 bits, com pouco detrimento das métricas acurácias Top 1 e Top 5. Na *Efficientnet* e *Caffenet*, os métodos *midrise* e *midtread* se destacam. Para a *Efficientnet*, a taxa alcançada é um pouco maior que 5 bits/pesos; para a *Caffenet*, chega-se à taxa de 2 bpp. A RN *VGG* tem uma melhor resposta para o método *midtread*, embora os 4 métodos possuam resultados similares, conseguindo chegar a uma entropia por volta de 2 bits/pesos.

Por fim, as duas redes que possuem resultados divergentes das demais são a *Mobilenet* e a *Resnet*. Para a *Mobilenet*, a curva referente ao quantizador não uniforme é a que apresenta melhor desempenho, porém as taxas são bem elevadas (em comparação com as demais) sendo maiores que 10 bpp. A *Resnet* tem a *midrise* e a não uniforme como os melhores quantizadores e entropia de aproximadamente 8,5 bits/pesos. Em geral, vemos que os métodos uniformes possuem melhores resultados para as redes, a exceção das duas últimas citadas.

3.7.2 *Deadzone*

com o método *deadzone*, conforme Figura 3.9, é possível chegar a baixas taxas de entropia com a acurácia em níveis similares. Como descrito na Seção 2.5.2, variando o *deadzone* podemos implementar todas quantizações uniformes, o valor de $\delta = 0.5$ representa o quantizador *midtread* e o valor $\delta = 0.0$ representa o quantizador *midrise*. Nos gráficos 3.9(b) e 3.9(c), em três oportunidades a precisão cresce bastante, e em duas ocasiões, o *recall* aumenta. Nesses 5 momentos, enquanto a precisão aumenta, o *recall* diminui, e vice-versa. Por isso, o uso da métrica f1-*score* torna a comparação mais justa e estável.

No modelo para a classificação de idade, com o método *deadzone*, vemos novamente que para alguns δ, a acurácia cresce para entropia em volta de 2 bits/pesos.

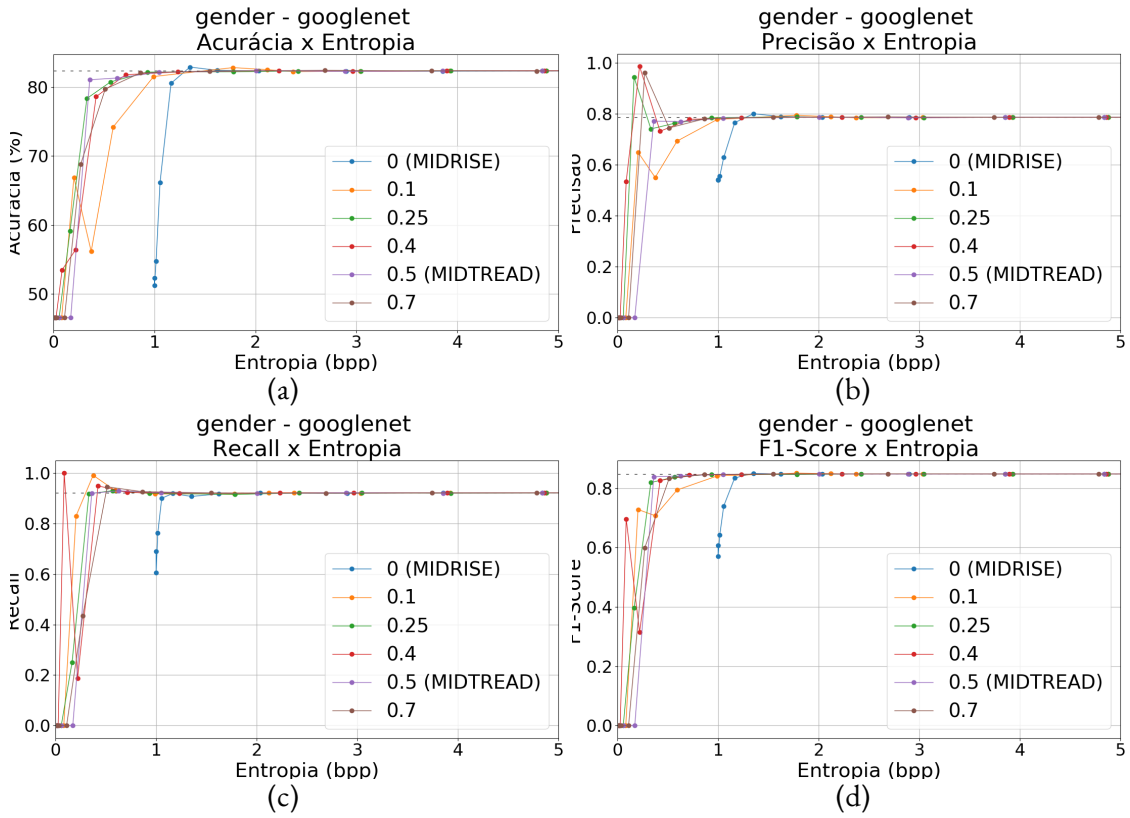


Figura 3.9: Comparações de taxa e distorção (entropia × acurácia) entre quantização uniforme e não uniforme para redes de classificação de gênero. O tamanho da *deadzone* é indicado na legenda.

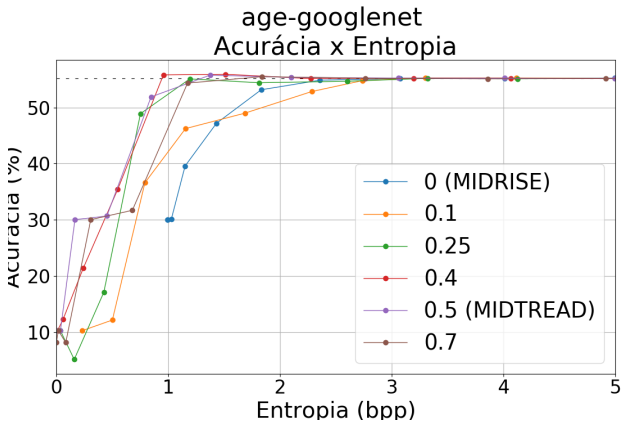


Figura 3.10: Comparações de taxa e distorção (entropia × acurácia) entre diferentes tamanhos de *deadzone* para várias redes de classificação de idade. O tamanho da *deadzone* é indicado na legenda.

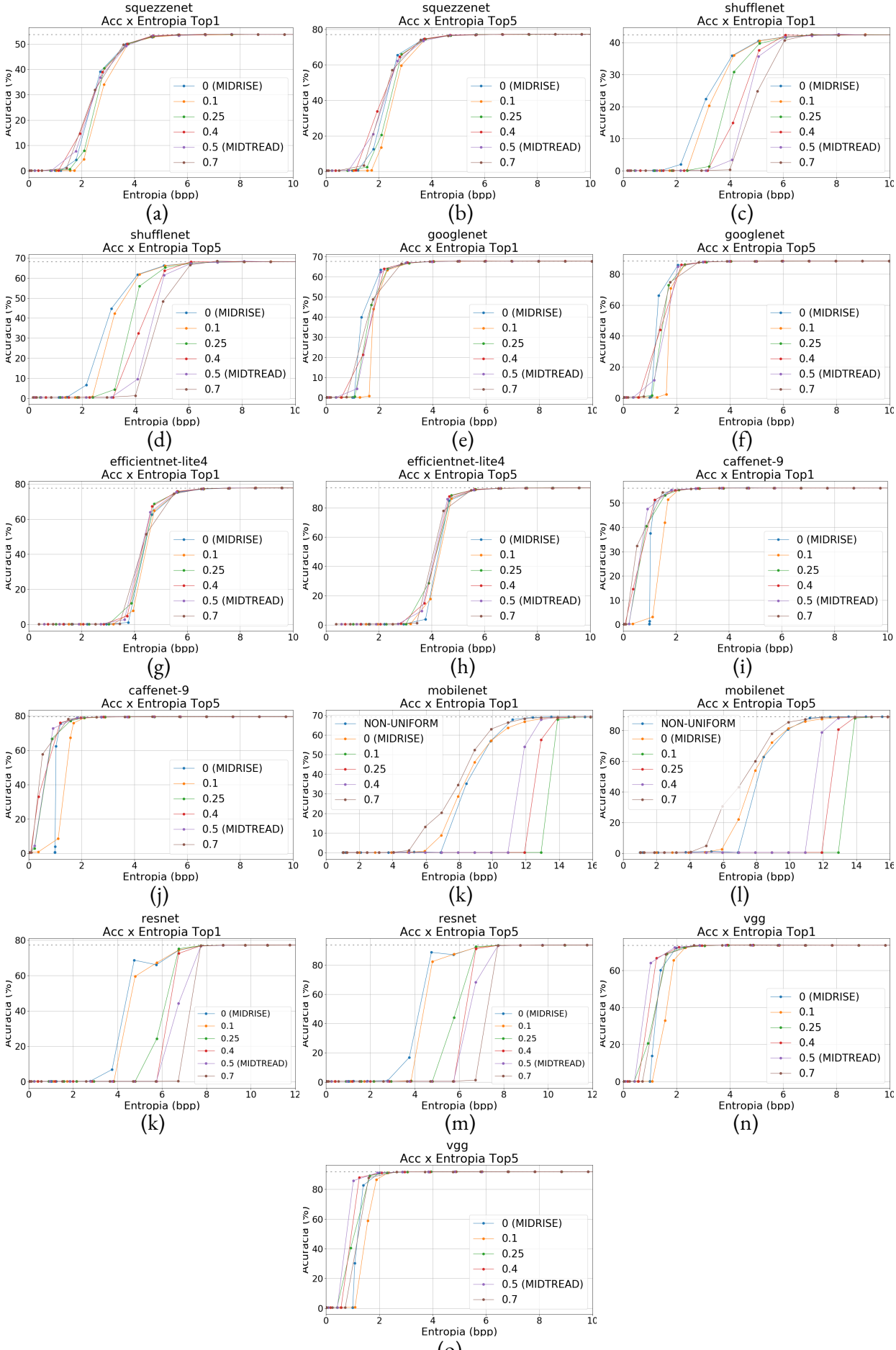


Figura 3.11: Comparações de taxa e distorção (entropia × acurácia) entre diferentes tamanhos de *deadzone* para várias redes de classificação de imagem. O tamanho da *deadzone* é indicado na legenda.

A Figura 3.11 contém as curvas referentes ao quantizador *deadzone* comparadas com os melhores quantizadores entre não uniforme e uniforme. A rede Squeezenet mostra um resultado similar para os 4 casos de *deadzone* e para os quantizadores uniformes (*midrise* e *midtread*). Para a Shufflenet, conseguimos ver que *deadzone* com $\delta = 0.4$ tem um melhor desempenho, com taxas aproximadamente 6 bits/pesos. A RN *Googlenet* mostra os comportamentos dos 6 quantizadores muito próximos uns aos outros, resultando em entropia um pouco maior que 2 bpp. A *efficientnet*, como a *Googlenet*, mantêm bons resultados para todos os quantizadores *deadzone*, conseguindo chegar a taxas de um pouco mais de 6 bpp. Os gráficos da *Caffenet* expõem um resultado melhor (um pouco) para a *midtread*, para entropia de 2 bits. A *VGG* possui um resultado melhor para *deadzone* com $\delta = 0.7$, os níveis entrópicos chegam perto de 2 bpp.

Para a rede *Mobilenet*, o quantizador não uniforme havia se destacado na comparação com os uniformes. Ao confrontar com os métodos de *deadzone*, vemos que *deadzone* com $\delta = 0.7$ tem um bom desempenho chegando a competir com o método não uniforme. Embora se aproximem, ainda é possível verificar um melhor desempenho para a quantização não uniforme.

As curvas referentes ao quantizador *deadzone* ($\delta = 0,1$, $\delta = 0,25$ e $\delta = 0,7$) para a *Resnet* possuem um resultado similar com a *midrise* ($\delta = 0.0$) e com a não uniforme. O método com *deadzone* mostra um desempenho um pouco melhor que a *midrise* ($\delta = 0.0$) e a não uniforme; para esse método, a entropia é um pouco inferior 7 bpp.

3.7.3 Síntese dos resultados

As Tabelas 3.9 e 3.10 sintetizam os resultados para os 10 modelos. Nelas são mostrados o melhor método (uniforme, não uniforme ou *deadzone*) e o nível que pode ser atingido. Ademais, aparecem informações sobre o desempenho atingido para as métricas e a diferença para o modelo base. A última coluna é a taxa de compressão (CR, *compression rate*), calculada assumindo um codificador entrópico perfeito e os pesos possuem 32 bits:

$$CR = \frac{32}{H}, \tag{3.1}$$

podemos calcular a redução de espaço pela fórmula:

$$RE = \frac{32 - H}{32} \tag{3.2}$$

Pela análise da Tabela 3.10, constatamos que a quantização *deadzone* possui o melhor resultado para quase todas as redes, a exceção da *Mobilenet* (melhor resultado pela não uniforme). Entretanto, essa teve um resultado satisfatório alcançado pela *deadzone* $\delta = 0.7$, com um prejuízo máximo de 1% para a acc_1 e sua entropia reduzida para 11,9 bpp (diminuindo 0,5 bpp em relação ao método apresentado na Tabela 3.10).

Trabalhos encontrados na literatura não são usados para essa quantidade de modelos, geralmente, se restringem a dois. Ao analisar os estudos que usam retreino, vemos para os métodos que

Tabela 3.9: Resumo dos melhores resultados da rede em classificação de Idade e *Gênero*.

Modelo	**Melhor Método**	**Nível**	**Acurácia (Dif. %)**	***Recall* (Dif. %)**	**Precisão (Dif. %)**	**F1-*Score* (Dif. %)**	**Entropia (Dif.)**	**CR**	**RE**
Gender Googlenet	*Midtread*	7	82,918 (-0,538)	0,908 (0,014)	0,8 (-0,017)	0,735 (0,001)	1,351 (22,439)	23,7	0,957
Age Googlenet	*Deadzone* ($\partial = 0,4$)	6	55,802 (-0,613)	-	-	-	0,96 (21,50)	33,33	0,97

Tabela 3.10: Resumo dos melhores resultados da rede.

Modelo	**Melhor Método**	**Nível**	**Acc. Top 1 (Diferença %)**	**Acc. Top 5 (Diferença %)**	**Entropia (Diferença)**	**CR**	**RE**
Squeezenet	*Midtread*	8	53,2 (0,57)	76,578 (0,56)	4,67 (15,542)	6,85	0,854
Shufflenet	*Midrise*	9	41,77 (0,652)	61,512 (0,626)	6,078 (14,314)	5,26	0,81
Googlenet	*Deadzone* ($\partial = 0,4$)	8	67,03 (0,744)	87,938 (0,402)	3,06 (19,594)	10,45	0,904
Efficientnet	*Deadzone* ($\partial = 0,7$)	9	77,194 (0,54)	93,374 (0,74)	6,528 (16,985)	4,9	0.796
Caffenet	*Midrise*	8	55,5 (0,79)	78,958 (0,564)	1,828 (23,385)	17,5	0,942
Mobilenet	Não Uniforme	13	69,03 (0,27)	88,926 (0,018)	12,414 (9,257)	2,58	0,612
Resnet	*Deadzone* ($\partial = 0,7$)	13	76,632 (0,582)	93,322 (0,292)	7,736 (16,997)	4,13	0,758
VGG	*Deadzone* ($\partial = 0,7$)	9	73,576 (0,07)	91,698 (0,118)	2,653 (23,007)	12,06	0,917

usam *prunning* há uma redução da VGG em 49× (redução de espaço de 0,98) [16], um CR de 7,5% [58]. Zhao et al demonstram uma redução de 73% da rede *VGG-19*. Nesta pesquisa, porém, chegou-se a uma redução de espaço de 0,917 e uma taxa de compressão de 12,06. Assim, concluiu-se que é um pouco pior que o estudo de Han et al [18] e melhor em comparação aos estudos de Han [16] e Dong [58]. Em confronto a pesquisa de Faraone et al [21] que usa quantização geral com retreino, conseguiu-se melhorar a taxa mantendo a acurácia Top1 e acurácia Top5 com prejuízos menores que 1%.

Acerca dos trabalhos que não envolvem retreino, podemos mencionar Dupuis et al que conseguem reduzir em 5× o montante de dados (0,8 de economia de espaço) para as modelos *Squeezenet* e *Resnet* [63] (com perda de no máximo 1% para acurácia Top1). Do nosso lado, conseguiu-se uma economia de 0,854 e 0,758 para *Squeezenet* e *Resnet*, respectivamente. Portanto, mantiveram-se resultados similares com uma perda menor (0,582% no máximo) que a relatada por Dupuis et al [63].

3.8 RETREINAMENTO

Depois de realizar a compressão para todos os métodos, escolheu-se a rede *Squeezenet* e a versão quantizada por *deadzone*, com ∂ igual 0,4 e 9 bits de níveis. A RN comprimida selecionada tinha uma pequena degradação em seu funcionamento: 0,098% para acc_1 e 0,158% para Top 5, a entropia, por sua vez, estava em 5,688 bits/pesos.

No retreinamento, utilizamos um *dataset* menor, composto por cerca de 34000 imagens [71]. Por sua vez, nos treinamentos usamos 150 e 300 épocas. Para realizar a validação durante o treinamento a quantidade total do conjunto dados foi dividido em 80% para treinar e 20% para validação. Após o retreinamento, a rede teve seu desempenho medido na etapa de validação de 50000 imagens (igual ao que ocorreu com demais métodos). Ao terminar a validação, pode-se constatar que a rede teve uma piora em seu funcionamento em relação ao modelo usado para retreinar (tanto para 150 quanto para 300 épocas) e a entropia voltou a níveis elevados, perto das taxas da RN original.

Em outra tentativa, utilizamos um número menor de imagens: 17000 imagens ao invés de 34000. Novamente, os resultados não foram bons. Porém, o retreinamento com 34000 imagens, obteve-se um desempenho melhor em relação ao anterior, com 17000. Provavelmente, o que explica o resultado ruim para o retreinamento é a quantidade de imagens usadas. As redes ILSVRC (que é o caso da *Squeezenet*) são treinadas originalmente por 1,28 milhões de imagens, portanto, para se ter um melhor desempenho no retreinamento, deve-se recorrer a um número similar ao usado para treiná-la originalmente.

4 CONCLUSÕES

Com o objetivo de realizar a compressão de RN's sem retreinamento no formato ONNX, um método para realizar a compressão dos pesos de uma RN foi apresentado. Foram estudados métodos de quantização para a codificação de RN's. Em que as distribuições dos pesos mostraram-se similares à distribuição Laplaciana.

Para analisar o melhor método de quantização, foi realizada a comparação entre as quantizações uniforme, não uniforme e *deadzone*. Entre as 10 redes usadas na etapa de validação, somente a *Mobilenet* não possui a melhor performance alcançada pelo método *deadzone* (o modo de quantização não uniforme obteve o melhor resultado). Nessa rede o método de *deadzone* com $\partial = 0.7$ teve o segundo melhor resultado. Para as demais redes, sempre obtivemos um melhor resultado pelos métodos de *deadzone* (lembrando que *midrise* e *midtread* são casos de quantização *deadzone*).

Em relação à compressão do modelo de classificação de gênero, conseguiu-se uma diminuição de 22 bits de entropia, com um prejuízo de 0,54% na acurácia. Para a rede classificação de idade, o funcionamento melhorou (acréscimo de 0,6% na acurácia) com a redução de 21 bits de taxa. Entretanto, para os oito modelos de classificação de imagem, tanto para acc. Top1, quanto para acc. Top5, alcançou-se um detrimento máximo de 0,8%. Originalmente, obteve-se uma média de entropia de 23 bits/peso. Mas, após a escolha da melhor compressão conseguiu-se uma média de 5,62 bits/peso. Quando considerado os 10 modelos, a entropia média vai de 22,896 bits/peso para 4,73. Por sua vez, a taxa de compressão ótima para os modelos de classificação de imagem foi no mínimo de 2,58 (*Mobilenet*) e no máximo de 17,5 (*Caffenet*), na média obteve-se aproximadamente 8. Para o modelo *Gender – Googlenet*, a CR alcançada foi de 23,7 e a CR para *Age Googlenet* é de 33,33.

Quanto às limitações deste trabalho, uma delas reside em não poder realizar o retreinamento da rede. Tal restrição ocorre devido ao alto poder computacional necessário para a execução do retreino. Muitas vezes, esse poder computacional não é encontrado em equipamentos como celulares, dispositivos IoT's e sistemas embarcados. Outro ponto negativo do retreinamento é a necessidade de conhecimento prévio do modelo, uma vez que a etapa de treinamento exige o conhecimento do *dataset* para ser usado. Nós também nos limitamos em utilizar modelos no formato ONNX, uma vez que ele é utilizado por vários arcabouços de DL e ML, permitindo a interoperabilidade.

Em suma, constata-se que nosso método é eficaz para dar guias para a codificação dos pesos em um formato de intercâmbio como o ONNX e realizar a compressão das RN's sem necessitar o retreino. A compressão sem retreinamento mostra-se vantajosa quando se verifica o tempo necessário para compressão com retreinamento, geralmente, muito maior que o tempo necessário pela quantização dos pesos. Entretanto, os resultados alcançados por esse trabalho são similares aos encontrados na literatura, com um grande diferencial de realizar a compressão para 10 modelos. Além disso, o método apresenta-se apto para ser usado em diferentes redes e até mesmo para modelos com

diversas funcionalidades. Cabe salientar que o método proposto consegue atingir os objetivos propostos no MPEG-NRR [65] e alcança os requisitos, expostos na seção 2.9.3, que são necessários na proposta do MPEG-NNR.

Para trabalhos futuros, sugere-se realizar a comparação para mais RN's, como as 4 redes que não foram usadas na etapa de validação. Outras redes com finalidades diversas (manipulação de imagem, *autoencoder*, compreensão de máquinas, entre outros) podem passar pelo método exposto neste trabalho. Embora não seja o foco da pesquisa, não há restrição para combinação do método escolhido com outros. Ainda, é possível combiná-lo com os métodos de *prunning* [16], [17], [58] ou outro método de compressão com retreinamento (citados na Seção 2.9.1). De outro lado, como trabalhos futuros, para os modelos de classificação de imagem ILSVRC, indica-se usar as métricas de ordenamento (Seção 2.4.3) MRR (Seção 2.4.3.2) para ser utilizada como métrica (além de acc_1 e acc_5), calculando a posição média em que o elemento *ground truth* é encontrado na saída do modelo.

REFERÊNCIA BIBLIOGRÁFICA

[1] E. B. M. B. Mohssen Mohammed, Muhammad Badruddin Khan, *Machine Learning. Algorithms and Applications*. CRC, 2017.

[2] S. Angra and S. Ahuja, "Machine learning and its applications: A review," in *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, 2017, pp. 57–60.

[3] D. Delen, "A comparative analysis of machine learning techniques for student retention management," *Decision Support Systems*, vol. 49, no. 4, pp. 498–506, 2010.

[4] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine learning*, vol. 30, no. 2, pp. 195–215, 1998.

[5] D. Feng, L. Rosenbaum, and K. Dietmayer, "Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3266–3273.

[6] J. Perols, "Financial statement fraud detection: An analysis of statistical and machine learning algorithms," *Auditing: A Journal of Practice & Theory*, vol. 30, no. 2, pp. 19–50, 2011.

[7] T. Xu, G. Han, X. Qi, J. Du, C. Lin, and L. Shu, "A hybrid machine learning model for demand prediction of edge-computing-based bike-sharing system using internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7345–7356, 2020.

[8] F. Ahamed and F. Farid, "Applying internet of things and machine-learning for personalized healthcare: Issues and challenges," in *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*, 2018, pp. 19–21.

[9] T. Lynn, P. T. Endo, A. M. N. C. Ribeiro, G. B. N. Barbosa, and P. Rosati, *The Internet of Things: Definitions, Key Concepts, and Reference Architectures*. Cham: Springer International Publishing, 2020, pp. 1–22. [Online]. Available: https://doi.org/10.1007/978-3-030-41110-7_1

[10] W. Sun, J. Liu, and Y. Yue, "Ai-enhanced offloading in edge computing: When machine learning meets industrial iot," *IEEE Network*, vol. 33, no. 5, pp. 68–74, 2019.

[11] Y. Zhang, W. Ding, and C. Liu, "Summary of convolutional neural network compression technology," in *2019 IEEE International Conference on Unmanned Systems (ICUS)*. IEEE, 2019, pp. 480–483.

[12] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[15] J. Bai, F. Lu, K. Zhang *et al.*, "Onnx: Open neural network exchange," https://github.com/onnx/onnx, 2019.

[16] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[17] L. Li, Z. Li, Y. Li, B. Kathariya, and S. Bhattacharyya, "Incremental deep neural network pruning based on hessian approximation," in *2019 Data Compression Conference (DCC)*. IEEE, 2019, pp. 590–590.

[18] W. B. Zhao, Y. Li, and L. Shang, "Fuzzy pruning for compression of convolutional neural networks," in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2019, pp. 1–5.

[19] T. Serra, A. Kumar, and S. Ramalingam, "Lossless compression of deep neural networks," 2020.

[20] J.-K. Kim, M.-Y. Lee, J.-Y. Kim, B.-J. Kim, and J.-H. Lee, "An efficient pruning and weight sharing method for neural network," in *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, 2016, pp. 1–2.

[21] J. Faraone, N. Fraser, M. Blott, and P. H. Leong, "Syq: Learning symmetric quantization for efficient deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4300–4309.

[22] M. V. Tonin and R. de Queiroz, "Codificação de redes neurais sem retreino," in *XXXIX Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2021) (SBrT 2021)*, Fortaleza/Virtual, Brazil, Sep. 2021.

[23] K. Warwick, *Artificial Intelligence: The Basics*. Routledge, 2011.

[24] T. Taulli, *Artificial Intelligence Basics: A Non-Technical Introduction*. Apress, 2019. [Online]. Available: https://books.google.com.br/books?id=zOOmDwAAQBAJ

[25] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*, 1st ed. Springer Publishing Company, Incorporated, 2018.

[26] A. C. P. d. L. F. Carvalho, A. d. P. Braga, and T. B. Ludermir, *Fundamentos de redes neurais artificiais*. DCC/IM, COPPE/Sistemas, NCE-UFRJ, 1998.

[27] E. B. Gorgens, H. G. Leite, H. d. N. Santos, and J. M. Gleriani, "Estimação do volume de árvores utilizando redes neurais artificiais," *Revista Árvore*, vol. 33, pp. 1141–1147, 2009.

[28] S. S. Haykin, *Neural networks and learning machines*, 3rd ed. Upper Saddle River, NJ: Pearson Education, 2009.

[29] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[30] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[31] Institute of Electrical and Electronics Engineers, "Ieee standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, 2008.

[32] F. López. (2020, oct) Onnx: Preventing framework lock in. [Online]. Available: https://towardsdatascience.com/onnx-preventing-framework-lock-in-9a798fb34c92

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[37] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[38] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," 2017.

[39] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[40] A. Zheng, *Evaluating Machine Learning Models: A Beginner's Guide to Key Concepts and Pitfalls*. O'Reilly Media, 2015. [Online]. Available: https://books.google.com.br/books?id=OFhauwEACAAJ

[41] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, p. 422–446, Oct. 2002. [Online]. Available: https://doi.org/10.1145/582415.582418

[42] W. B. Croft, D. Metzler, and T. Strohman, *Search engines: information retrieval in practice*, 1st ed. Boston: Addison-Wesley, Feb. 2010. [Online]. Available: http://www.amazon.com/Search-Engines-Information-Retrieval-Practice/dp/0136072240

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[44] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications*. USA: Prentice-Hall, Inc., 1996.

[45] S. Smith, *Digital Signal Processing: A Practical Guide for Engineers and Scientists*, ser. Demystifying technology series : by engineers, for engineers. Elsevier Science, 2003. [Online]. Available: https://books.google.com.br/books?id=PCrcintuzAgC

[46] A. K. Jain, *Fundamentals of Digital Image Processing*. USA: Prentice-Hall, Inc., 1989.

[47] W. A. Pearlman and A. Said, *Digital Signal Compression: Principles and Practice*. Cambridge University Press, 2011.

[48] R. E. Crochiere, "A mid-rise/mid-tread quantizer switch for improved idle-channel performance in adaptive coders," *The Bell System Technical Journal*, vol. 57, no. 8, pp. 2953–2955, 1978.

[49] D. Taubman and M. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer Publishing Company, Incorporated, 2013.

[50] Sayood, *Introduction to Data Compression*, 3rd ed. Morgan, 2006.

[51] B. Widrow and I. Kollár, *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. USA: Cambridge University Press, 2008.

[52] A. Papoulis and U. Pillai, *Probability, random variables and stochastic processes*, 4th ed. McGraw-Hill, Nov. 2001.

[53] N. Johnson, S. Kotz, and A. Kemp, "Univariate discrete distribution," *Technometrics*, vol. 36, 02 1994.

[54] A. A. Salvia, "Reliability application of the alpha distribution," *IEEE Transactions on Reliability*, vol. R-34, no. 3, pp. 251–252, 1985.

[55] J. S. deCani and R. A. Stine, "A note on deriving the information matrix for a logistic distribution," *The American Statistician*, vol. 40, no. 3, pp. 220–222, 1986. [Online]. Available: http://www.jstor.org/stable/2684541

[56] G. J. Sullivan, "Optimal entropy constrained scalar quantization for exponential and laplacian random variables," in *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 1994, pp. V–265.

[57] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, Mar. 1951.

[58] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," 2017.

[59] X. Zhu, W. Zhou, and H. Li, "Adaptive layerwise quantization for deep neural network compression," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2018, pp. 1–6.

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[62] S. Seo and J. Kim, "Hybrid approach for efficient quantization of weights in convolutional neural networks," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2018, pp. 638–641.

[63] E. Dupuis, D. Novo, I. O'Connor, and A. Bosio, "Sensitivity analysis and compression opportunities in dnns using weight sharing," in *2020 23rd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, 2020, pp. 1–6.

[64] P. Haase, H. Schwarz, H. Kirchhoffer, S. Wiedemann, T. Marinc, A. Marban, K. Müller, W. Samek, D. Marpe, and T. Wiegand, "Dependent scalar quantization for neural network compression," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 36–40.

[65] I. O. F. S. O. I. D. NORMALISATION, "Use cases and requirements for compressed representation of neural networks," ISO/IEC JTC1/SC29/WG11, Macau SAR, CN, Approved WG 11 document N17924, Oct. 2018.

[66] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.

[67] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[68] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[69] S. Lapuschkin, A. Binder, K.-R. Müller, and W. Samek, "Understanding and comparing deep neural networks for age and gender classification," in *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 1629–1638. [Online]. Available: https://doi.org/10.1109/ICCVW.2017.191

[70] G. Levi and T. Hassner, "Age and gender classification using convolutional neural networks," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) workshops*, June 2015. [Online]. Available: https://osnathassner.github.io/talhassner/projects/cnn_agegender

[71] I. Figotin, "Imagenet 1000 (mini)," https://www.kaggle.com/ifigotin/imagenetmini-1000, 09 2020.