# CLASSIFICAÇÃO DE IMAGENS COM ARTEFATOS DE COMPRESSÃO: UMA ABORDAGEM FIM-A-FIM

ANDREY OTACÍLIO OLIVEIRA DOS REIS

DISSERTAÇÃO DE MESTRADO
EM ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

# Classificação de Imagens com Artefatos de Compressão: Uma Abordagem Fim-A-Fim

Andrey Otacílio Oliveira dos Reis

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

---

Prof. Dr. Daniel Guerreiro e Silva (Universidade de Brasília)
(Orientador)

---

Prof. Dr. Hugo Valadares Siqueira (Universidade Tecnológica Federal do Paraná)
(Examinador Externo)

---

Prof. Dr. Eduardo Peixoto Fernandes da Silva (Universidade de Brasília)
(Examinador Interno)

---

Prof. Dr. Francisco Assis de Oliveira Nascimento (Universidade de Brasília)
(Examinador Interno)

Brasília/DF, Maio de 2024.

**FICHA CATALOGRÁFICA**

DOS REIS, ANDREY OTACÍLIO OLIVEIRA

Classificação de Imagens com Artefatos de Compressão: Uma Abordagem Fim-A-Fim. [Brasília/DF] 2024.

xv, 70p., 210 x 297 mm (ENE/FT/UnB, Mestre, Dissertação de Mestrado, 2024).

Universidade de Brasília, Faculdade de Tecnologia.

Departamento de Engenharia Elétrica.

1. Classificação de imagens
2. Compressão de imagens
3. Redes neurais
4. JPEG
5. Redução de artefatos
6. Aprendizado fim-a-fim

I. ENE/FT/UnB
II. Título (série)

**REFERÊNCIA BIBLIOGRÁFICA**

DOS REIS, ANDREY OTACÍLIO OLIVEIRA (2024). Classificação de Imagens com Artefatos de Compressão: Uma Abordagem Fim-A-Fim. Dissertação de Mestrado, Publicação PPGENE.DM-813/24, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 70p.

**CESSÃO DE DIREITOS**

AUTOR: Andrey Otacílio Oliveira dos Reis

TÍTULO: Classificação de Imagens com Artefatos de Compressão: Uma Abordagem Fim-A-Fim.

GRAU: Mestre ANO: 2024



---

Andrey Otacílio Oliveira dos Reis

Universidade de Brasília (UnB)

Campus Darcy Ribeiro

Faculdade de Tecnologia - FT

Departamento de Engenharia Elétrica(ENE)

Brasília - DF CEP 70919-970

*Para meu pai Ailton dos Reis (in memorian), com toda minha gratidão.*

# AGRADECIMENTOS

Eu agradeço primeiramente à Deus, Jesus Cristo, por todos os aspectos do desenvolvimento desse trabalho. Aos meus familiares e amigos, em especial à minha mãe Maria Angélica de Oliveira dos Reis, pelo apoio, confiança e orações. Aos meus orientadores, Daniel Guerreiro e Silva e Edson Mintsu Hung, pelos valiosos ensinamentos e instrução acadêmica. E à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela concessão da bolsa de mestrado.

# RESUMO

A classificação refinada de imagens é uma modalidade de inferência muito importante em visão computacional, devido à sua utilidade em abordar problemas que possuem um número elevado de classes. Além de exigir modelos com maior capacidade de aprendizado, o conjunto de imagens usadas no treinamento deve conter uma grande quantidade de amostras de boa qualidade. A presença de compressão com perdas, porém, ao degradar a qualidade do sinal, pode dificultar drasticamente a tarefa do classificador. Uma vez que a compressão muitas vezes viabiliza a composição dos conjuntos de treinamento dos modelos, o seu impacto nesse tipo de aplicação não pode ser ignorado. Nesse trabalho, propomos uma arquitetura de Rede Neural Artificial (RNA) capaz de mitigar significativamente os prejuízos causados pela compressão JPEG. Para tanto, ela conta com uma dupla ramificação de redes que serão treinadas em conjunto. Os dois ramos, um para Redução de Artefatos da Compressão (RAC) e outro para classificação, são conectados de forma que a saída do primeiro é a entrada do segundo. O ramo de RAC consiste em uma RNA de geração de imagens responsável pela redução dos efeitos de perdas no sinal comprimido. O ramo de classificação, por sua vez, utiliza-se de uma RNA pré-treinada para receber essas imagens reconstruídas como entrada e executar a classificação. Dessa forma, o treinamento fim-a-fim é capaz tanto de melhorar a qualidade do sinal, priorizando características importantes para a classificação, quanto de se adaptar a receber imagens restauradas com certa degradação para realizar a inferência. Nos dois conjuntos de imagens que utilizamos, Caltech 200 Cub e Oxford 102 Flower, aumentamos a acurácia média para 10 fatores de qualidade (FQs) diferentes em 46.54% e 5.81%, respectivamente. Apesar da pouca flexibilidade do modelo em relação aos FQs e a necessidade de um treinamento adicional, o nosso trabalho evidencia uma correlação entre a eficiência do codificador e o desempenho do classificador, além de apresentar uma arquitetura que extrai proveito desse aspecto para melhorar o processo de classificação de imagens comprimidas.

Palavras-chave: classificação de imagens, compressão de imagens, redes neurais, JPEG, redução de artefatos, aprendizado fim-a-fim

# ABSTRACT

Fine-grained image classification is a very important category of classification in computer vision due to its usefulness in tackling problems with a large number of classes. As well as requiring models with greater learning capacity, the datasets used for training must contain a large number of good quality samples. However, lossy compression can drastically hinder the classifier's task by degrading the signal quality. Compression and its impact on this type of application cannot be ignored, since it is something that even enables building the training datasets for the models. In this work, we propose a Neural Network (NN) architecture capable of mitigating the damage caused by JPEG compression. To do so, it relies on a double branch structure that is trained together. The two branches, one for Compression Artifacts Reduction (CAR) and the other for classification, are connected in such a way that the output of the first is the input of the second. The CAR branch consists of an image generation NN responsible for reducing the distortion effects in the compressed signal. In turn, the classification branch uses a pre-trained NN to receive these reconstructed images as input and perform the classification. In this way, end-to-end training is able to improve both the signal quality by prioritizing important features for classification and to adapt to receiving restored images with a certain amount of degradation in order to carry out inference. In the two datasets we used, Caltech 200 Cub and Oxford 102 Flower, we increased the average accuracy for 10 different quality factors (QFs) by 45.6% and 5.81%, respectively. Despite the model's lack of flexibility with regard to QFs and the need for additional training, our work shows a strong correlation between codec efficiency and classifier performance. It also presents an architecture that takes advantage of this aspect to improve the compressed image classification process.

Keywords: image classification, image compression, neural network, JPEG, artifacts reduction, end-to-end learning

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# GLOSSÁRIO

| | |
|---|---|
| AOLM | *Attention Object Location Module* |
| APPM | *Attention Part Proposal Module* |
| CCT | *Compact Convolutional Transformer* |
| CNN | *Convolutional Neural Network* |
| FC | *Fully-connected* |
| FQ | Fator de Qualidade |
| IA | Inteligência Artificial |
| JPEG | *Joint Photographic Experts Group* |
| MLP | *Multilayer Perceptron* |
| MMAL-Net | *Multi-branch and Multi-scale Attention Learning* |
| MMFF | *Multi-scale feature upsampling block* |
| MSFU | *Multi-modal feature fusion block* |
| N2N | *Noise2Noise* |
| RAC | Redução de Artefatos da Compressão |
| RNA | Rede Neural Artificial |
| RNP | Rede Neural Profunda |
| TC | Taxa de Compressão |
| ViT | *Vision Transformer* |

CAPÍTULO 1

# INTRODUÇÃO

A compressão de imagens consiste em um conjunto de técnicas de codificação que são utilizadas para reduzir o tamanho em bits do sinal. Essencialmente, a aplicação da compressão nas imagens visa possibilitar o seu armazenamento e transmissão. Por essa razão a compressão é amplamente usada e pode ser considerada como um padrão no tratamento do sinal. Existem dois tipos de compressão de imagens: com ou sem perdas. A principal forma de se medir a eficiência de um compressor de imagens (*codec*) é a partir da redução do espaço ocupado na memória e a preservação do sinal decodificado.

A compressão sem perdas é aquela em que o sinal decodificado é igual ao sinal da fonte. Apesar da sua eficiência ser considerada elevada em alguns casos, a sua principal aplicação é em situações em que possíveis perdas não podem ser admitidas. Arquivos de textos e aplicativos são exemplos de dados que não admitem perdas ao serem armazenados ou transmitidos, pois qualquer tipo de degradação pode tornar esses dados inutilizáveis. Contudo, as imagens, assim como outros sinais digitais, não possuem o mesmo problema em relação à degradação gerada pela compressão com perdas. Além disso, a redução de bits do objeto diminui expressivamente, comparado com os métodos sem perdas (MIANO, 1999; HUSSAIN *et al.*, 2018).

As perdas que são acrescidas pelo *codec*, mesmo para níveis elevados de compressão, não degradam significativamente a qualidade da imagem (PRASANNA *et al.*, 2021). Métricas como PSNR e SSIM podem indicar uma redução considerável na preservação do sinal e, mesmo assim, a apuração visual pode não ser tão afetada (HUYNH-THU; GHANBARI, 2012). Nesse sentido, a compressão com perdas representou um grande avanço nas aplicações com imagem, possibilitando em muitos cenários a sua utilização, sobretudo em aplicações que necessitam grandes volumes de dados, por exemplo aquelas ligadas à Inteligência Artificial (IA).

A compressão com perdas não necessariamente prejudicará a aplicação em que será usada a imagem. Porém, em certos cenários só é possível utilizar o sinal se a compressão alcançar

taxas de eficiência que possibilitem a transmissão e/ou armazenamento do sinal. Por essa razão não existe um limite em que a imagem pode ser comprimida. Sendo assim, uma redução drástica na qualidade da imagem pode ser negativamente impactante para o consumo humano ou computacional. Em modelos de classificação de imagens baseados em Redes Neurais Profundas (RNPs), essa forma de compressão pode ser bastante prejudicial (LAU *et al.*, 2003) e reduzir consideravelmente o desempenho do classificador.

A Figura 1.1 apresenta um esquemático de como grande parte das RNPs de classificação são treinadas, testadas e utilizadas na prática. A partir das imagens capturadas é composta uma base de dados (*dataset*) que será usada tanto no treinamento quanto no teste da rede. Antes disso, todas ou muitas dessas imagens irão passar pelo processo de compressão. Uma vez que o modelo treinado está pronto para ser usado em inferências, ele certamente irá se deparar com imagens que também tiveram que passar pela etapa de compressão com perdas.

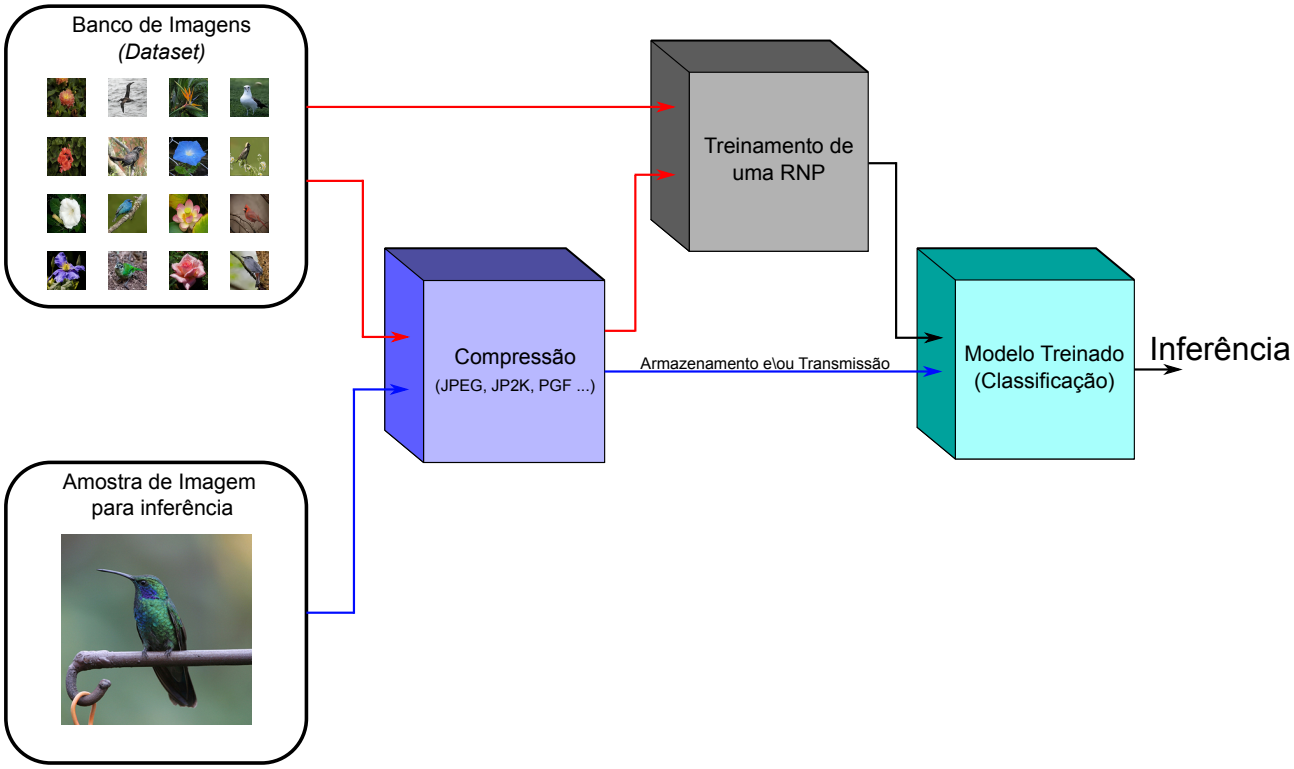



**Figura 1.1.** Compressão com perdas apresentada como etapa essencial no processamento de imagens que serão usadas na classificação.

Os resultados apresentados em Jo *et al.* (2021) e Benbarrad *et al.* (2021) evidenciam os impactos negativos de níveis elevados de compressão na classificação de imagens. Os diferentes tipos de artefatos gerados na compressão são responsáveis pela degradação da qualidade do sinal, cada artefato está associado a uma operação do codificador (EBRAHIMI *et al.*, 2004).

Apesar das operações inversas realizadas na decodificação serem as responsáveis pela recuperação da imagem, o sinal resultante não é igual ao original devido à presença de artefatos. Nesse sentido, técnicas de redução de artefatos baseadas em RNPs (DONG *et al.*, 2015; AMARANAGESWARAO *et al.*, 2020) conseguem apresentar resultados expressivos em melhorar a qualidade das imagens comprimidas.

A aplicação de uma RNP de redução de artefatos no processo de classificação pode ser vantajosa, uma vez que a melhora na qualidade do sinal é um fator determinante para aumentar o desempenho do classificador (YANG *et al.*, 2021). Além disso, por se tratar de um processo baseado em rede neural, a Redução de Artefatos da Compressão (RAC) pode ser acoplada à RNP de classificação. Dessa forma, ocorre a possibilidade de um treinamento conjunto de duas arquiteturas com finalidades diferentes, porém complementares.

Neste trabalho, propomos uma RNP desenvolvida especialmente para classificar imagens JPEG. O nosso objetivo é atenuar ao máximo todas as perdas de desempenho proporcionadas pela degradação da imagem pela compressão. Para tanto, implementamos uma rede com dupla ramificação em uma abordagem fim-a-fim. Em seguida, diversos treinamentos e testes foram realizados com dois *datasets* distintos aplicados ao nosso modelo.

O JPEG (WALLACE, 1991) é um tipo de codificador que adiciona perdas ao sinal. Apesar de décadas desde o seu desenvolvimento pelo Joint Photographic Experts Group (JPEG) esse *codec* de imagem conseguiu se estabelecer durante todo esse tempo até os dias atuais como a ferramenta de compressão mais utilizada. Neste trabalho, todas a imagens que utilizamos são comprimidas exclusivamente com esse codificador.

O conteúdo restante desta dissertação está disposto da seguinte forma. No próximo capítulo, realizamos a revisão bibliográfica a respeito do tema de pesquisa. Em seguida, no capítulo 3, apresentamos a fundamentação teórica que descreve as ferramentas que utilizamos com mais detalhes. No capítulo 4 é apresentada a metodologia na qual a nossa RNP foi desenvolvida. Os experimentos que realizamos para treinar e testar o nosso modelo são apresentados no capítulo 5. Em seguida, os resultados e a discussão de todos os testes realizados estão no capítulo 6. Finalmente, concluímos esta dissertação no capítulo 7.

CAPÍTULO 2

# REVISÃO BIBLIOGRÁFICA

Os impactos do codificador JPEG na classificação de imagens possuem certa atenção na comunidade acadêmica, assim como formas de mitigar a degradação dos modelos. Neste trabalho buscamos atenuar o problema da compressão com perdas utilizando Redes Neurais Artificiais (RNAs) capazes de reduzir artefatos. A seguir serão apresentados os temas relacionados com este trabalho e sua respectiva bibliografia mais recente.

## 2.1 IMPACTOS DA COMPRESSÃO NA CLASSIFICAÇÃO DE IMAGENS

A avaliação das RNAs de classificação de imagem depende em grande parte das entradas escolhidas. A apuração da qualidade de um banco de imagens é uma etapa fundamental na sua composição. O ideal é conseguir os melhores sinais possíveis para que tanto o treinamento quanto o teste dos modelos de classificação possam ser executados adequadamente. Isso se torna algo ainda mais importante em cenários em que o alto desempenho dos classificadores é indispensável. Na aplicação em diagnósticos médicos, por exemplo, não apenas a acurácia deve ser elevada, mas outras métricas como a área abaixo da curva *Receiver Operating Characteristic* (AUC ROC), sensibilidade e especificidade também (AGGARWAL *et al.*, 2021). Nesse intuito, Jo *et al.* (2021) apresentam os impactos da degradação de mamografias por compressão JPEG e JPEG 2000. Em seus experimentos os modelos são significativamente prejudicados apenas em casos extremos, nos quais a taxa de compressão (razão entre o tamanho do arquivo original pelo tamanho do arquivo comprimido) ultrapassa 5000. Ou seja, o que se percebe neste estudo é a possibilidade de manter o desempenho das RNAs mesmo após uma compressão moderada nas imagens.

No trabalho de Benbarrad *et al.* (2021) é explorado o ambiente da Indústria 4.0, em que a Visão Computacional em cooperação com a Internet das Coisas é utilizada para a análise de

componentes de aço. O tipo de sinal aplicado nesse caso são imagens de superfícies metálicas que podem ter algum tipo de defeito, como arranhões. A ideia principal dos autores é avaliar tanto os impactos que a compressão JPEG pode causar nos classificadores quanto propor uma solução para o possível problema de degradação. Assim como para as mamografias, os experimentos aqui mostraram a possibilidade de se preservar o desempenho dos modelos para determinados níveis de compressão.

Ambos os trabalhos apresentados usaram uma abordagem de aumento de dados para atenuar o problema de perda de qualidade dos modelos devido à compressão. A base de dados de treino foi aumentada com a adição de imagens comprimidas e em seguida os respectivos modelos passaram pelo treinamento. Esse aumento da base de imagens proporcionou uma elevação na taxa de compressão limite em que os modelos começam a perder desempenho. O melhor método observado é treinar as RNAs com as imagens comprimidas com a mesma taxa de compressão das que vão ser aplicadas no teste.

Outra técnica interessante para atenuar os problemas da compressão na classificação de imagens surge da otimização do próprio codificador JPEG. Em Li *et al.* (2020), os valores da tabela de quantização do JPEG são os parâmetros a serem otimizados para aumentar a eficiência da compressão e o desempenho da classificação simultaneamente. A otimização bayesiana em cooperação com a busca aleatória é o método aplicado. O resultado mostrou que essa otimização pode funcionar, no entanto, com melhora significativa apenas para a busca aleatória, em que a técnica bayesiana serviu mais como um refinamento do processo. De maneira semelhante, Amer *et al.* (2023) propõem um codificador JPEG com fator de qualidade adaptativo, o qual visa escolher o melhor custo benefício entre a eficiência da compressão e a acurácia do classificador.

## 2.2 REDUÇÃO DE ARTEFATOS DA COMPRESSÃO COM APRENDIZADO PROFUNDO

A eficiência de um codificador está associada à preservação da qualidade do sinal, à medida que ocorre a compressão. A principal estratégia para evitar perdas de qualidade é desenvolver o codificador que utilize processos que proporcionem melhores resultados, nesse sentido. No entanto, mesmo com técnicas mais sofisticadas, existe um *tradeoff* entre qualidade e diminuição

do tamanho do sinal que não pode ser contornado ao se desenvolver o codificador. Os artefatos que são gerados em decorrência das etapas de codificação para determinada taxa de compressão só poderão ser atenuados por operações externas a esse processo. Por esse motivo, técnicas de redução de artefatos ganharam destaque no decorrer da história dos codificadores de sinais.

Inicialmente técnicas que envolvem somente filtros, como as apresentadas em List *et al.* (2003) e Wang *et al.* (2013) se mostraram capazes de realizar *deblocking* nos sinais corrompidos. Os dois trabalhos indicam a possibilidade de usar filtros baseados nas estatísticas da imagem. A *Shape-Adaptative Discrete Cosine Transformer* (SA-DCT) (FOI *et al.*, 2007) vai além do *deblocking*, conseguindo realizar *deringing* e até mesmo reduzir artefatos de distorção de cores em imagens coloridas. Geralmente, a redução de artefatos baseada em filtros e transformadas adaptativas suavizam drasticamente texturas e pontas nas imagens. Dessa forma, um outro artefato não produzido em codificadores que usam *block-DCT* surge em decorrência dessa suavização excessiva, o *blurring*.

A partir da difusão da aplicação de Inteligência Computacional na codificação, geração e classificação de imagens, a possibilidade de utilizar aprendizado de máquinas para reduzir artefatos, em especial *Convolutional Neural Networks* (CNNs), começou a ser explorada. As *Artifacts Reduction Convolutional Neural Networks* (AR-CNN) propostas em (DONG *et al.*, 2015) ultrapassam o desempenho da SA-DCT tanto em qualidade objetiva quanto em qualidade subjetiva. A estrutura desse modelo de RAC é baseado na *Super-resolution Convolutional Neural Network* (SR-CNN), que consiste em uma ferramenta de *upsizing*.

A maior desvantagem das primeiras redes neurais de RAC consistia na necessidade de conhecer o fator de qualidade (FQ) das imagens comprimidas, e somente assim treinar as redes individualmente para cada FQ. Isso tornava a aplicação desses modelos inviáveis, uma vez que seria necessário treinar uma quantidade muito grande de redes e contar com informação da FQ, que nem sempre poderia ser apresentada, a depender do codificador. Por essa razão, Jiang *et al.* (2021) e Amaranageswarao *et al.* (2020) estabelecem modelos que não precisam conhecer o FQ das imagens comprimidas para fazer sua reconstrução. Além disso, uma RN treinada apenas uma vez é capaz de aplicar a redução de artefatos para uma diversidade de FQs.

A maioria dos *datasets* de imagens para treinamento de classificadores já possui algum nível de compressão. Porém, os experimentos deste trabalho realizam uma segunda compressão

nos sinais, para que se possa testar os classificadores diante de cenários com algum nível de degradação não observado em tempo de treinamento. O problema da reconstrução de imagens duplamente comprimidas já possui certa atenção (YOON; CHO, 2023). O foco maior tem sido em casos nos quais a segunda taxa de compressão é maior do que a primeira, o exato cenário da nossa aplicação. Isso ocorre em muitas situações em que é necessária uma compressão adicional ao sinal para possibilitar a sua utilização.

## 2.3 APRENDIZADO FIM-A-FIM COM REDES NEURAIS

A ideia de treinar RNAs utilizando modelagem fim-a-fim não é recente. No trabalho de Abbass (2003) é aplicada a otimização multiobjetivo para aumentar a eficiência do treinamento das RNAs evolucionárias. Já Roth *et al.* (2006) usam essa mesma otimização para que a tarefa de detecção de objetos alcance melhores desempenhos.

Apesar da otimização ser fim-a-fim, as propostas apresentadas nesses dois trabalhos não possuem as características do que propomos aqui. Algo que seria mais próximo é a rede de múltipla ramificação para segmentação de esclerose (ASLANI *et al.*, 2019). Neste modelo, são usadas três ResNets (HE *et al.*, 2016) em paralelo. As ResNets são CNNs residuais compostas por cinco blocos de convolução.

A entrada do segmentador consiste na imagem de ressonância magnética de três tipos: FLAIR, T1w e T2w. Cada tipo de sinal é inserido em um dos ramos da CNN. A saída de cada bloco das ResNets, além das conexões residuais, é conectada a blocos MMFF [1], que por sua vez são conectadas a blocos MSFU[2].

As MMFFs são compostas de duas camadas de convolução de *kernels* 1×1 e 3×3, para cada entrada, além de realizar uma concatenação entre as saídas de cada uma delas. Cada MSFU, no entanto, recebe a saída da MMFF residual e a saída da última MMFF. Esse bloco concatena as duas informações e aplica duas convoluções (1×1 e 3×3). Essa técnica basicamente consiste em aumentar as *features* de baixa resolução adicionando-as com as de alta resolução.

Nesse trabalho, apesar da múltipla ramificação de RNAs, cada uma conta com uma

[1] *Multi-scale feature upsampling block* (Tradução livre: Bloco de aumento de *feature* multi-escala).
[2] *Multi-modal feature fusion block* (Tradução livre: Bloco de fusão de *feature* multi-modal).

entrada e saída específica. A CNN em si possui apenas uma saída que é utilizada para o cálculo da função de custo de treinamento. Nesse caso os autores propõem a função de custo *Dice loss* (SUDRE *et al.*, 2017) para ser minimizada.

CAPÍTULO 3

# FUNDAMENTAÇÃO TEÓRICA

A nossa proposta utiliza diversas técnicas computacionais que deverão ser detalhadas, a fim de justificar o seu uso neste trabalho. Em um primeiro momento, nosso objetivo é observar as possíveis causas dos prejuízos causados pelo codificador JPEG nos modelos de classificação. Em seguida, compreender o funcionamento desses modelos e como eles interagem com o sinal comprimido, para finalmente apresentar uma forma de amenizar a redução de desempenho dos classificadores, a partir de algum método próprio relacionado à compressão JPEG. O processo de codificação JPEG e suas consequências, a classificação de imagens com redes neurais convolucionais e transformador visual, a redução de artefatos da compressão com RNPs, e os princípios de aprendizado fim-a-fim serão abordados nas próximas seções.

## 3.1 JPEG

O codificador de interesse do nosso trabalho é o JPEG. Usamos esse algoritmo de compressão com perdas pelo fato de ser o mais utilizado. Mesmo com outros *codecs* mais eficientes, como o JPEG 2000 (TAUBMAN *et al.*, 2002), o JPEG tem uma eficiência próxima dos codificadores mais modernos. A seguir, será apresentado o funcionamento do codificador, que é uma etapa fundamental para compreender os artefatos gerados por ele. Em seguida, esses artefatos serão melhor detalhados e o motivo pelo qual eles prejudicam a classificação ficará mais evidente.

### 3.1.1 Estrutura do codificador

A Figura 3.1 apresenta a estrutura padrão do JPEG. Inicialmente, a imagem deve ser convertida do sistema de cores RGB (*red, green, blue*) para o sistema YCbCr (luminância e

crominância). Essa conversão aumenta consideravelmente a eficiência do codificador, uma vez que permite maiores perdas de informação nos canais de crominância que visualmente são menos impactantes nas imagens. Antes de começar o processo de codificação, a imagem é dividida em blocos de 8×8 pixels. Todas as operações são realizadas para cada bloco separadamente, a começar pela transformada discreta do cosseno.

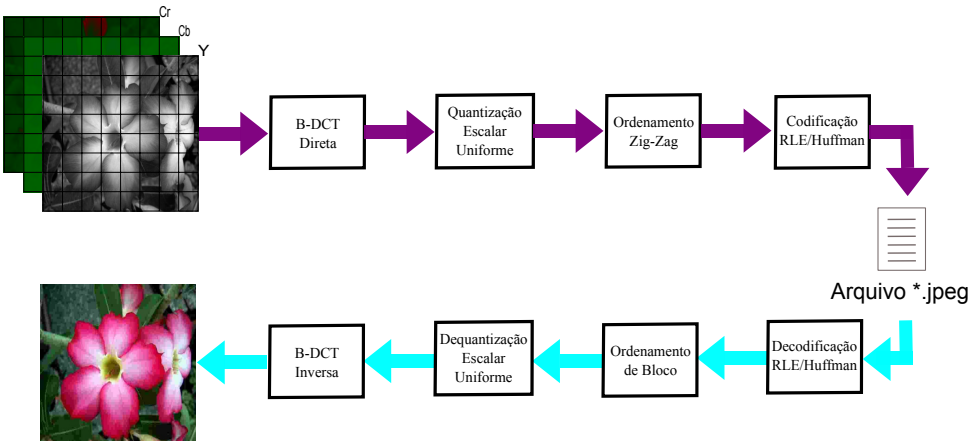



**Figura 3.1.** Estrutura do codificador JPEG.

#### 3.1.1.1 Transformada

A transformada aplicada nos blocos da imagem é a *Discrete Cosine Transform* (DCT). Ela é a responsável por colocar o sinal no domínio da frequência. Feito isto, os coeficientes mais próximos da cauda do bloco (altas frequências) podem ser quantizados com relativamente poucos bits. Essa transformada explora a falta de sensibilidade visual humana para altas frequências, apesar de ser computacionalmente mais eficiente obter os coeficientes da DCT pela transformada rápida de Fourier. A seguir apresentamos a B-DCT direta, uma forma de obter os coeficientes de uma DCT em duas dimensões:

$$\theta_{ij} = a[i,j] \sum_{x=0}^{7} \sum_{y=0}^{7} I[x,y] cos \frac{(2y+1)i\pi}{16} cos \frac{(2x+1)j\pi}{16}, \tag{3.1}$$

em que $a[0,0..7] = a[0..7,0] = 1/8$, $a[1..7,1..7] = 2/8$, $I[x,y]$ são os pixels da imagem e $\theta_{ij}$ os coeficientes da DCT.

Já a B-DCT inversa é calculada conforme a equação:

$$I[x,y] = \sum_{i=0}^{7}\sum_{j=0}^{7} a[i,j]\theta_{ij} cos\frac{(2y+1)i\pi}{16} cos\frac{(2x+1)j\pi}{16}. \tag{3.2}$$

#### 3.1.1.2 Quantização

Após a transformação para o domínio da frequência, os coeficientes resultantes da DCT passam pela quantização escalar uniforme *midtread*. Essa é a etapa na qual perdas são impostas ao sinal, pois informações são descartadas. Durante essa operação, também é definido o fator de qualidade (FQ) da codificação. Matematicamente, a quantização é expressa por:

$$l_{ij} = \lfloor \frac{\theta_{ij}}{Q_{ij}} + 0.5 \rfloor, \tag{3.3}$$

em que $\theta_{ij}$ é o coeficiente da DCT, $Q_{ij}$ é o passo de quantização e $l_{ij}$ é o rótulo (*label*) do coeficiente quantizado.

O valor de $Q_{ij}$ é indicado na tabela de quantização, a Figura 3.2 mostra um exemplo. Nesse caso, a tabela se refere à qualidade padrão do JPEG, ou seja, com FQ de 50. Além disso, existem duas tabelas, uma para o canal de luminância e outra para o canal de crominância, na qual na primeira os valores são menores, o que indica um passo de quantização mais curto e menos descarte de informação. Isso é possível porque a luminância, como mencionado anteriormente, é mais importante na construção de imagens para o consumo humano.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 11 | 10 | 16 | 24 | 40 | 51 | 61 |
| 12 | 12 | 14 | 19 | 26 | 58 | 60 | 55 |
| 14 | 13 | 16 | 24 | 40 | 57 | 69 | 56 |
| 14 | 17 | 22 | 29 | 51 | 87 | 80 | 62 |
| 18 | 22 | 37 | 56 | 68 | 109 | 103 | 77 |
| 24 | 35 | 55 | 64 | 81 | 104 | 113 | 92 |
| 49 | 64 | 78 | 87 | 103 | 121 | 120 | 101 |
| 72 | 92 | 95 | 98 | 112 | 100 | 103 | 99 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 17 | 18 | 24 | 47 | 99 | 99 | 99 | 99 |
| 18 | 21 | 26 | 66 | 99 | 99 | 99 | 99 |
| 24 | 26 | 56 | 99 | 99 | 99 | 99 | 99 |
| 47 | 66 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 64 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |

**Figura 3.2.** Tabelas de quantização para o JPEG com fator de qualidade 50. À esquerda passos para luminância e à direita para crominância.

Em um cenário de compressão mais agressiva, a tabela de quantização terá uma aparência

mais próxima à que está na Figura 3.3. O FQ igual a 5 estabelece passos de quantização 10 vezes maiores do que na qualidade padrão do codificador. Isso representa eliminação de componentes do sinal importantes na construção da imagem e que não serão recuperados.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 160 | 110 | 100 | 160 | 240 | 400 | 510 | 610 |
| 120 | 120 | 140 | 190 | 260 | 580 | 600 | 550 |
| 140 | 130 | 160 | 240 | 400 | 570 | 690 | 560 |
| 140 | 170 | 220 | 290 | 510 | 870 | 800 | 620 |
| 180 | 220 | 370 | 560 | 680 | 1090 | 1030 | 770 |
| 240 | 350 | 550 | 640 | 810 | 1040 | 1130 | 920 |
| 490 | 640 | 780 | 870 | 1030 | 1210 | 1200 | 1010 |
| 720 | 920 | 950 | 980 | 1120 | 1000 | 1030 | 990 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 170 | 180 | 240 | 470 | 990 | 990 | 990 | 990 |
| 180 | 210 | 260 | 660 | 990 | 990 | 990 | 990 |
| 240 | 260 | 560 | 990 | 990 | 990 | 990 | 990 |
| 470 | 660 | 990 | 990 | 990 | 990 | 990 | 990 |
| 990 | 990 | 990 | 990 | 990 | 990 | 990 | 990 |
| 990 | 990 | 990 | 990 | 990 | 990 | 990 | 990 |
| 990 | 640 | 990 | 990 | 990 | 990 | 990 | 990 |
| 990 | 990 | 990 | 990 | 990 | 990 | 990 | 990 |

**Figura 3.3.** Tabelas de quantização para o JPEG com fator de qualidade 5. À esquerda passos para luminância e à direita para crominância.

#### 3.1.1.3 Codificação de Entropia

A última etapa do JPEG consiste na aplicação do algoritmo de Huffman no *Run Lenght Encoding* (RLE). Basicamente, devido à alta quantidade de zeros resultantes da quantização, esse método codifica a sequência de zeros consecutivos. Os códigos para cada possibilidade são tabelados e podem ser analisados em Sayood (2017). Os componentes DC são codificados separadamente por um processo que também é tabelado, mas que representa códigos associados à diferença de cada rótulo com o anterior.

Antes da codificação, os coeficientes quantizados devem ser organizados em zigue-zague (Figura 3.4), isso permite que o agrupamento de valores nulos, inclusive os que estão na cauda do bloco, sejam codificados com um simples *End of Block* (EOB).

### 3.1.2 Artefatos da Compressão

As operações executadas pelo JPEG são responsáveis por proporcionar uma compressão muito eficiente, porém também são responsáveis pela degradação do sinal. Como indicado em Jakulin (2002), essa degradação devido ao processo de codificação é conhecida como artefatos da compressão. A seguir será descrito cada um dos artefatos do JPEG.

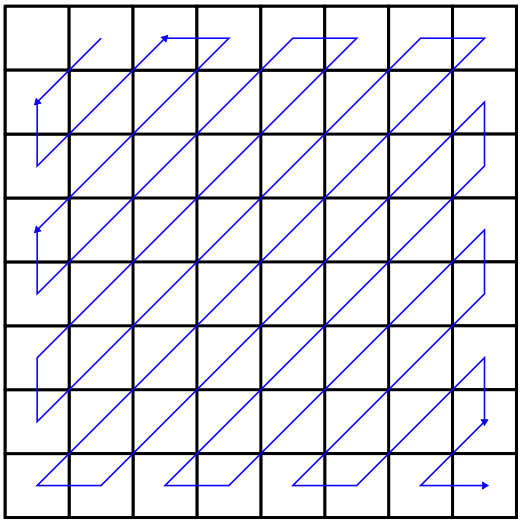


**Figura 3.4.** Padrão zigue-zague para vetorização dos coeficientes quantizados da DCT.

#### 3.1.2.1 *Blocking*

Todas as etapas do JPEG são realizadas em blocos 8×8 dos canais YCbCr da imagem. Quando características das imagens são perdidas no processo de quantização, tais blocos começam a se sobressair e o efeito de *blocking* acontece. A Figura 3.5 ilustra a ocorrência desse tipo de artefato em uma imagem do *dataset* usado.

**Figura 3.5.** Artefato *Blocking* gerado na compressão JPEG. Imagem original à esquerda e comprimida à direita.

O *blocking*, assim como os outros artefatos, varia em função da resolução da imagem. Quanto menor for a resolução, maior será a degradação da imagem devido aos artefatos, pois cada bloco ocupará uma maior parte do sinal. Essa informação é muito importante ser observada, pois geralmente as imagens de *datasets* de classificação possuem uma resolução diminuída para aplicação nos modelos.

#### 3.1.2.2 *Ringing*

Os coeficientes de altas frequências resultantes da DCT possuem pouca participação na construção visual da imagem. No entanto, quando a quantização descarta valores cada vez mais expressivos na alta frequência, o resultado é a perda de detalhes da imagem. Um fenômeno análogo ao efeito de Gibbs pode ser observado em imagens, com listras e mudanças abruptas de cores (Figura 3.6).

**Figura 3.6.** Artefato *Ringing* gerado na compressão JPEG. Imagem original à esquerda e comprimida à direita.

#### 3.1.2.3 Distorção de Cores

A Figura 3.7 retrata a distorção de cores que pode ocorrer quando a compressão é muito grande. Esse comportamento ocorre devido aos passos de quantização para as camadas de crominância serem maiores do que para a camada de luminância. Em situações drásticas de compressão, isso faz com que a crominância seja distorcida ao ponto de corromper significativamente o sinal.

#### 3.1.2.4 *Blurring*

O *blurring* é o único artefato que geralmente não apresenta incidência no JPEG. A característica essencial desse artefato é a de deixar a imagem com aspecto de embaçamento. Em cenários de compressão excessiva, é possível visualizá-lo no JPEG 2000. Todavia, no nosso caso, a reconstrução de imagens com modelos de aprendizado profundo por um lado atenua a maioria dos artefatos, mas por outro gera *blurring*, como pode ser visto na Figura 3.8.

**Figura 3.7.** Distorção de cores produzida pelo *codec* JPEG. Imagem original à esquerda e comprimida à direita.

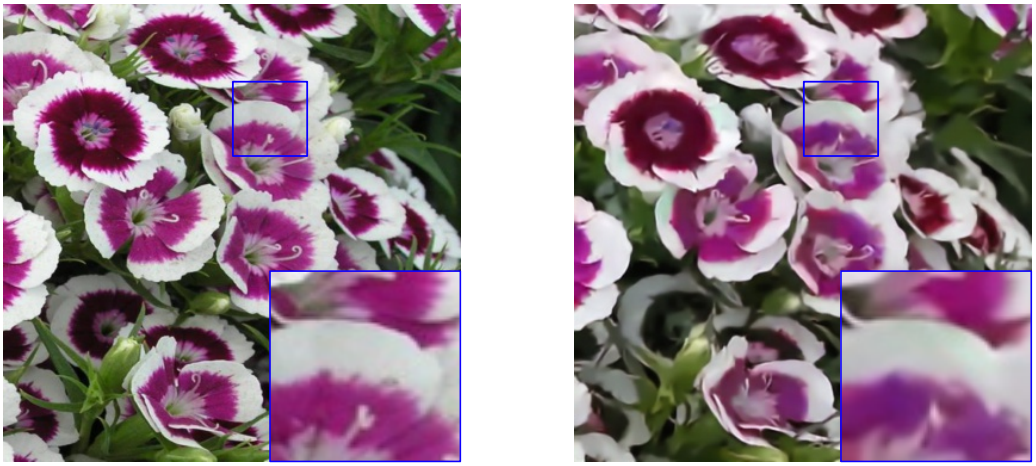


**Figura 3.8.** Artefato *Blurring* de imagem reconstruída pelo RAC. Imagem original à esquerda e comprimida à direita.

## 3.2 CLASSIFICAÇÃO REFINADA DE IMAGENS

A classificação refinada de imagens é aquela que possui uma quantidade bem grande classes, na ordem de centenas. Um problema de classificação refinada exige modelos que apresentem uma capacidade de discriminação maior. Além disso, a avaliação desses modelos não necessita de uma análise interclasses. A principal métrica de avaliação dos resultados de um classificador refinado é a acurácia (PENG *et al.*, 2017):

$$\text{Acurácia}(\%) = \frac{\#\text{ de classificações certas}}{\#\text{ total de classificações}}. \tag{3.4}$$

Uma vez que a quantidade de classes é muito grande e geralmente não existe um desbalanceamento entre elas, qualquer outra métrica será pouco diferente da acurácia. Por exemplo,

a F1-Score geralmente terá o mesmo valor da acurácia considerando duas casas decimais. Isso ocorrerá, pois a F1-Score é calculada considerando o desequilíbrio entre as classes. Uma classificação com poucas classes tende a ter uma F1-Score que diverge da acurácia, assim como também acontecerá divergência entre precisão, *recall* e outras métricas.

Os nossos experimentos foram realizados usando dois modelos de classificação refinada. No próximo capítulo serão apresentados em detalhes os *datasets* utilizados. Porém, nas próximas seções, vale a pena fundamentar as técnicas de aprendizado profundo em que esses modelos foram estruturados: redes neurais convolucionais e transformador visual. E por fim, ainda neste capítulo, apresentaremos o método de aprendizado fim-a-fim, o qual é a base do classificador que desenvolvemos.

### 3.2.1 Redes Neurais Convolucionais

As redes neurais artificiais constituem hoje a principal ferramenta de aprendizado de máquina. Em especial, os modelos de aprendizado profundo, que correspondem às redes que possuem diversas camadas, são os que estão no estado da arte com resultados expressivos em suas aplicações. A Figura 3.9 ilustra um exemplo de *Multilayer Perceptron* (MLP) com três camadas, sendo elas: uma de entrada, uma oculta e uma de saída. A camada de uma MLP é constituída por uma quantidade fixa de unidades de ativação (nós) que são conectadas com unidades de outras camadas por meio de arestas. Cada aresta possui um peso que multiplica com o valor do nó anterior e soma com os valores das outras arestas do nó atual. Esse processo é conhecido como propagação direta.

A descrição matemática da propagação direta aplicada à rede da Figura 3.9 pode ser observada pelas equações a seguir:

$$a_1^{(2)} = g(\Theta_{10}^1 x_0 + \Theta_{11}^1 x_1 + \Theta_{12}^1 x_2), \tag{3.5}$$

$$a_2^{(2)} = g(\Theta_{20}^1 x_0 + \Theta_{21}^1 x_1 + \Theta_{22}^1 x_2), \tag{3.6}$$

$$a_3^{(2)} = g(\Theta_{30}^1 x_0 + \Theta_{31}^1 x_1 + \Theta_{32}^1 x_2), \tag{3.7}$$

$$h_{1\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^2 a_0^{(2)} + \Theta_{11}^2 a_1^{(2)} + \Theta_{12}^2 a_2^{(2)} + \Theta_{13}^2 a_3^{(2)}), \tag{3.8}$$

$$h_{2\Theta}(x) = a_2^{(3)} = g(\Theta_{20}^2 a_0^{(2)} + \Theta_{21}^2 a_1^{(2)} + \Theta_{22}^2 a_2^{(2)} + \Theta_{23}^2 a_3^{(2)}), \tag{3.9}$$

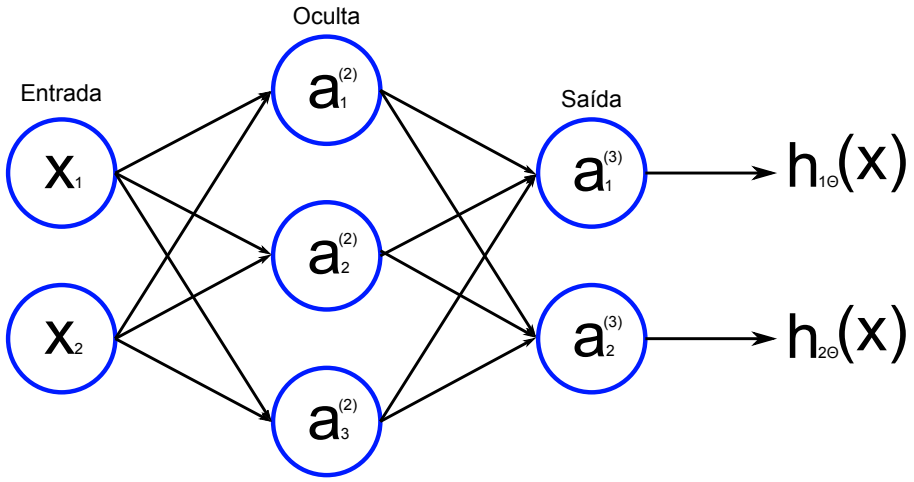



**Figura 3.9.** Representação esquemática de uma MLP como três camadas totalmente conectadas.

em que $x_k$ são as entradas do MLP, $\Theta_{ij}^{l}$ são os parâmetros da l-ésima camada, $g()$ é a função de ativação, a qual comumente é a *Rectified Linear Unit* (ReLU): $g(z) = max(0,z)$, com a curva apresentada na Figura 3.10. As unidades $a_0$ omitidas na representação da RNA são chamadas de unidades de polarização e não aparecem, pois são constantes e iguais a 1 (AGGARWAL, 2018).

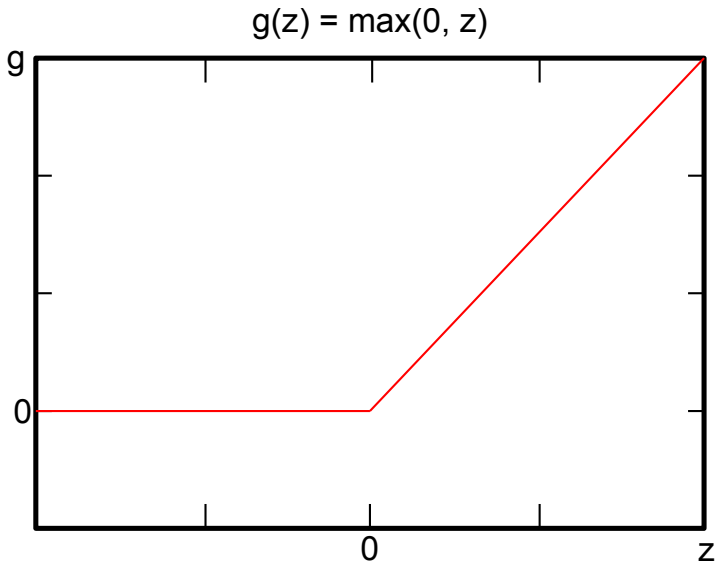



**Figura 3.10.** Curva característica da função de ativação ReLU.

O treinamento de uma RNA consiste na busca dos parâmetros (pesos) $\Theta$ da rede neural que otimizam o valor da função objetivo denominada função de custo. A título de exemplo, a função de custo baseada na entropia cruzada é dada por

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}[y_k^{(i)}\log(\hat{y}_k^{(i)}) + (1-y_k^{(i)})\log(1-\hat{y}_k^{(i)})] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2, \qquad (3.10)$$

em que $\hat{y}$ é a saída da rede, i.e. $h_\Theta(x)$, $y$ é a saída correta (*ground-truth*), $m$ é o tamanho do conjunto de treinamento, $L$ é o número de camadas total da rede, $s_l$ é o número de unidades na camada $l$ (com exceção da unidade de polarização) e $K$ é o número de unidades de saída. Repare ainda na existência de um somatório triplo, essa parcela é denominada termo de regularização, que é uma técnica utilizada na prevenção de *overfitting* e consiste em uma etapa importante para aumentar a capacidade de generalização do modelo. O produto $\lambda$ é o hiper-parâmetro de regularização (BALDI; SADOWSKI, 2013).

O método mais utilizado para realizar a minimização da função de custo é a descida do gradiente. No caso de uma rede neural com múltiplas camadas, há uma dificuldade em calcular o gradiente diretamente. Neste caso, o método mais usado para esse intuito é o algoritmo de retropropagação (CILIMKOVIC, 2015). Em relação à rede MLP citada como exemplo, o algoritmo de retropropagação (*backpropagation*) apresentará os seguintes passos:

1. Inicializar com $a^{(1)} := x^{(t)}$;
2. Executar a propagação direta para calcular $a^{(l)}$ para $l$ = 1,2,...,$L$;
3. A partir de $y^{(t)}$, calcular $\sigma^{(L)} = a^{(L)} - y^{(t)}$;
4. Calcular $\sigma^{(L-1)}$, $\sigma^{(L-2)}$, ..., $\sigma^{(2)}$ a partir de $\sigma^{(l)} = ((\Theta^{(l)})^T \sigma^{(l+1)}) \cdot g'(a^{(l)})$. Em que:
   - $g'(a_j^{(l)}) = 1$ se $a_j^{(l)} > 0$;
   - $g'(a_j^{(l)}) = 0$ se $a_j^{(l)} \leq 0$;
5. Usando $\Delta_{i,j}^{(l)} := \Delta_{i,j}^{(l)} + a_j^{(l)} \sigma_i^{(l+1)}$ obter
   - $D_{i,j}^{(l)} := \frac{1}{m}(\Delta_{i,j}^{(l)} + \lambda \Theta_{i,j}^{(l)})$ se $j \neq 0$;
   - Caso contrário, $D_{i,j}^{(l)} := \frac{1}{m}\Delta_{i,j}^{(l)}$.

Esse algoritmo pode ocorrer de forma iterada até encontrar um valor mínimo para a função de custo. A otimização, todavia, apenas ocorre com o uso do resultado da retropropagação aplicado à descida do gradiente:

1. Embaralhar o conjunto de treinamento;

2. Para todas as amostras de entrada e $l = 1,2,...,L$ atualizar os parâmetros da rede através da equação:

$$\Theta_{i,j}^{(l)} = \Theta_{i,j}^{(l)} - \alpha D_{i,j}^{l}. \tag{3.11}$$

Apesar da diversidade de suas aplicações, a arquitetura de uma RNA apresentada até o momento pode não ter um desempenho aceitável quando as entradas são imagens. Sendo assim, outro tipo de RNA conhecida como *Convolutional Neural Network* (CNN) consegue ter um melhor desempenho neste cenário. Os motivos para que isso seja possível são importantes e serão descritos em breve, porém antes devemos detalhar mais um pouco o funcionamento de uma CNN.

A CNN apresenta a mesma estrutura de uma RNA convencional, com a diferença de que a entrada e saída de cada camada são bidimensionais. Além dos parâmetros que serão ajustados, que na realidade representam o núcleo da operação de convolução dada por:

$$S(i,j) = (K * I)(i,j) = \sum_{m} \sum_{n} I(i-m,j-n)K(m,n), \tag{3.12}$$

em que $S$ é a saída, $I$ é o sinal de imagem e $K$ é o núcleo (*Kernel*) da convolução.

A Figura 3.11 mostra o equivalente a unidades de ativação para uma CNN. Cada passo da convolução aplicada seria correspondente a uma unidade de ativação e os componentes de cada *Kernel* seriam os parâmetros da rede. Esse exemplo mostra um núcleo 2×2 com *stride* (passo do núcleo de convolução) igual a 1.

A representação da camada de uma CNN pode ser visualizada na Figura 3.13. A convolução e a função de ativação já foram citadas e exemplos de suas respectivas operações foram mostrados. O *Pooling* (GHOLAMALINEZHAD; KHOSRAVI, 2020) é o terceiro componente da camada. Um exemplo popular de *pooling* é o *max pooling* (Figura 3.12), que basicamente dentro de uma janela pré-dimensionada seleciona o valor máximo como saída. Por exemplo, em uma janela 2×2, o resultado da operação é um único valor que será o maior entre os quatro valores selecionados. Essa etapa, com o *stride* apropriado, além de descartar *features* redundantes ou desprezíveis, preserva características importantes do sinal para o aprendizado da rede.

Na classificação, a última camada de uma CNN é geralmente idêntica à de uma RNA convencional (*fully connected*). A função de ativação também muda para estimar o valor de

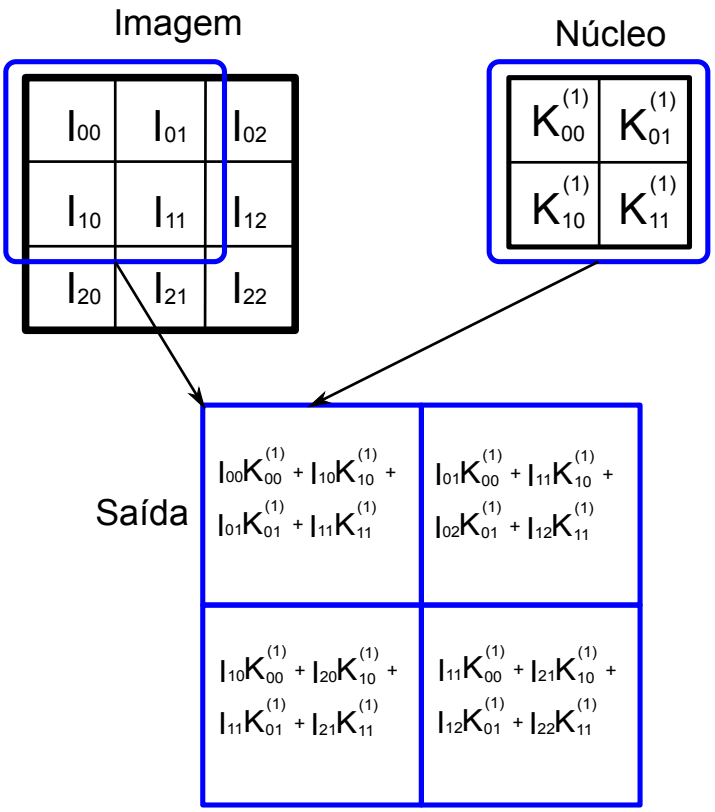



**Figura 3.11.** Operação de convolução aplicada por uma CNN em um sinal de imagem.

probabilidade, sendo no caso a função *softmax*. A saída ainda é apurada e aquela unidade com maior probabilidade representa a classe determinada pela CNN.

A seguir serão detalhadas as motivações de aplicação de uma CNN não só em classificação, mas também no processamento de imagens em geral. As informações a respeito de redes neurais profundas aqui apresentadas podem ser encontradas em Goodfellow *et al.* (2016).

#### 3.2.1.1 Conectividade Esparsa

As imagens que são as entradas das CNNs podem ter milhares ou milhões de pixels. Contudo, as características mais significativas do sinal estão comumente presentes em poucos pixels próximos uns dos outros, como arestas e contornos. A Figura 3.14 mostra que, para uma determinada saída, apenas as unidades de entrada próximas influenciam no cálculo, quando a convolução é aplicada. Diferentemente do que acontece para a camada *fully-connected*, na qual todas as unidades estão interconectadas e, assim, a entrada mais distante impacta na computação de uma determinada saída.

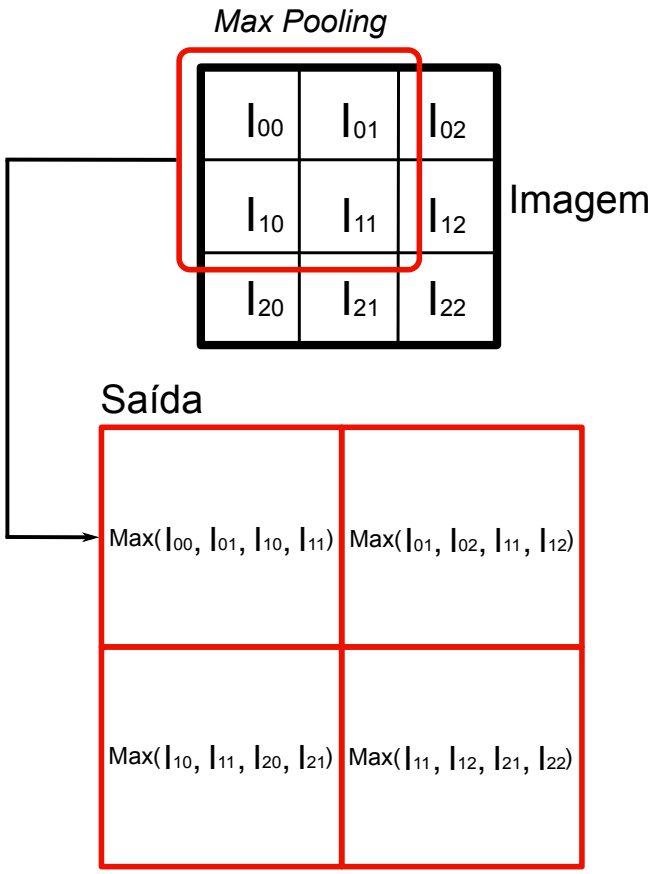



**Figura 3.12.** Operação de *max pooling* aplicada por uma CNN em um sinal de imagem.

#### 3.2.1.2 Compartilhamento de Parâmetros

O núcleo usado na convolução é aplicado ao longo de todo o sinal. Isso permite que os parâmetros utilizados para uma determinada entrada sejam sucessivamente reutilizados nas outras entradas (Figura 3.15). Esse processo aumenta drasticamente a eficiência do treinamento da CNN.

#### 3.2.1.3 Equivariância

Uma característica da operação de convolução em imagens é que se trata de uma operação linear e invariante à translação (LIT). Isso descreve que, se a entrada for deslocada, a saída muda da mesma forma. Esse comportamento é importante, pois possibilita a obtenção de características importantes da imagem, como as bordas. A Equação 3.13 apresenta a descrição matemática para esse efeito.

$$f(g(x)) = g(f(x)). \tag{3.13}$$

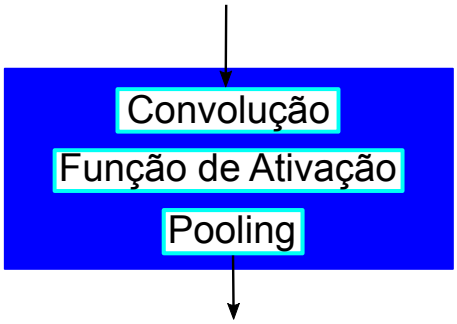



**Figura 3.13.** Exemplo de estrutura da camada de uma CNN.

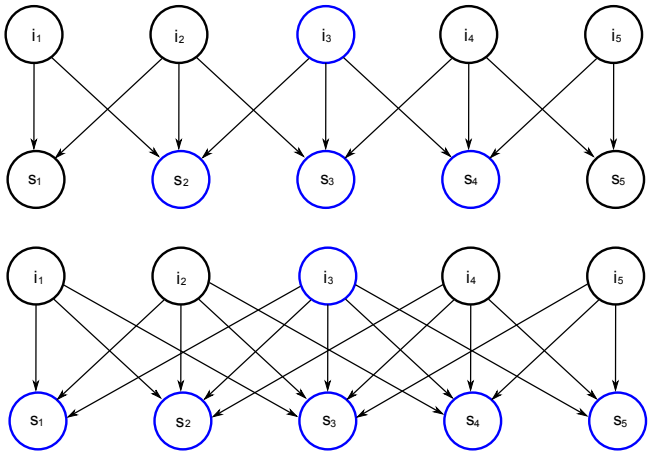



**Figura 3.14.** Característica da camada convolucional e *fully-connected*. Imagem superior apresenta a esparsividade de uma camada convolucional e a inferior mostra o comportamento de uma *fully-connected*.

### 3.2.2 Modelo de Transformador Visual

Além da CNN, outra técnica de DL para classificação que utilizamos foi o Transformador Visual (*Vision Transformer*, ViT) elaborado por Dosovitskiy *et al.* (2020). Baseado na proposta de Vaswani *et al.* (2017) para o processamento de linguagem natural, a ViT utiliza somente a arquitetura *Transformer*, sem haver camadas recorrentes ou convolucionais.

O modelo baseado em transformador também é conhecido como modelo de atenção. Isso se deve pelo fato do principal componente desse modelo ser uma estrutura de atenção, o *Multi-Head Attention*. A Figura 3.16 apresenta a arquitetura desse mecanismo. A Equação 3.14 é a operação realizada pela estrutura de atenção, denominada *Scaled Dot-Product Attention*.

$$\text{Atenção}(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \tag{3.14}$$

em que $Q$ são os dados de consulta, $K$ são as chaves, $V$ são os valores e $d_k$ é a dimensão de

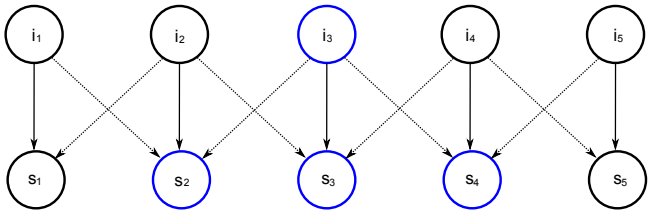



**Figura 3.15.** Compartilhamento de parâmetros possibilitado pela convolução.

cada exemplo na matriz $K$ e $Q$.

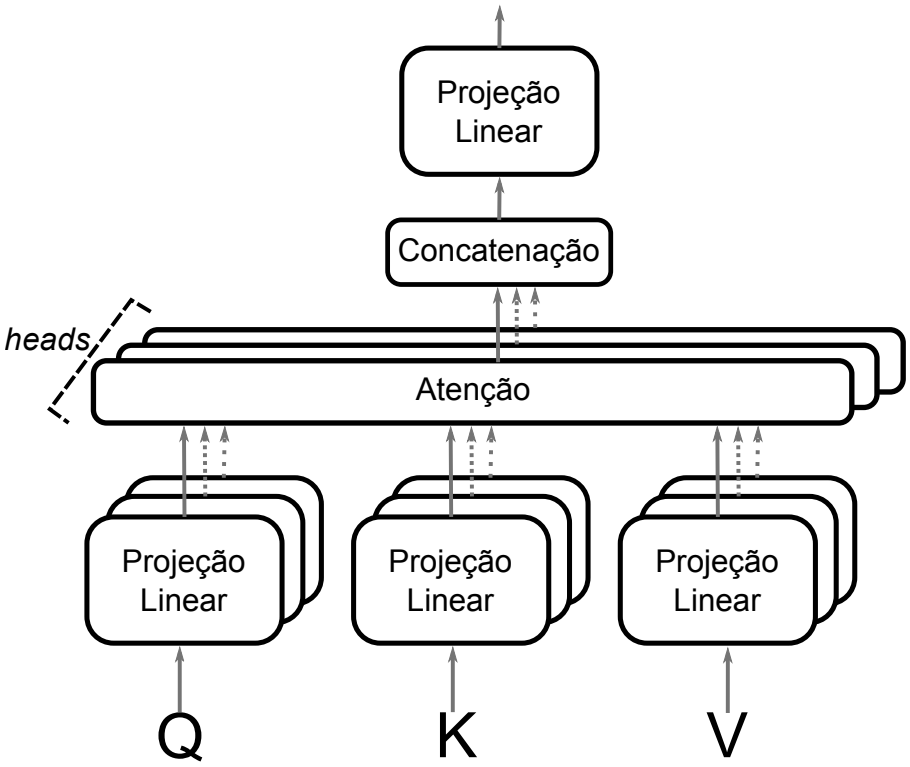



**Figura 3.16.** Arquitetura do *Multi-head Attention*, principal componente de um modelo baseado em transformador.

A operação linear se trata de uma projeção aprendida com parâmetros $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$ e $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ para as entradas $Q$, $K$ e $V$, respectivamente.

A concatenação é o processo no qual cada saída da etapa de atenção, denominadas *head*, são unidas em uma matriz antes da última projeção linear. A quantidade de *heads* é indicada pela letra $h$. A partir desse parâmetro é possível encontrar a dimensão do modelo:

$$d_{modelo} = h \cdot d_k. \tag{3.15}$$

Além do *Multi-head Attention*, o modelo transformador é composto por adição residual mais normalização e camadas de RNA com função de ativação ReLU. A saída, por sua vez,

utiliza a função *softmax*, que retorna valores entre 0 e 1, ou seja, estimativa de probabilidades.

Na entrada ainda é necessário adicionar informação de posição dos *tokens* (entradas vetorizadas) que serão inseridos. O motivo desse procedimento está no fato da ausência de operações com essa finalidade, uma vez que não existe convolução nem recorrência nesse modelo. A ordem é uma informação importante no aprendizado e não pode ser descartada. A codificação de posição, como é chamado esse processo, aplica as Equações 3.16 e 3.17 no dado de posição que será agregado ao respectivo *token*.

$$PE_{(p,2i)} = \text{sen}\left(\frac{p}{10000^{2i/d_{modelo}}}\right) \tag{3.16}$$

$$PE_{(p,2i+1)} = \cos\left(\frac{p}{10000^{2i/d_{modelo}}}\right) \tag{3.17}$$

em que $p$ é a posição do *token* na sequência, $i$ é a dimensão do vetor e $d_{modelo}$ é a dimensão do modelo.

Em relação ao processamento de imagens com *self-attention*, Wang *et al.* (2018) apresentou um modelo que mesclava CNN com mecanismos de atenção. Isso mostrou a possibilidade de usar transformadores em aplicação de visão computacional. Apesar de não ser a primeira implementação puramente a base de transformadores (RAMACHANDRAN *et al.*, 2019), o ViT foi certamente a que conseguiu mostrar a grande capacidade dessa técnica na classificação de imagens.

A arquitetura aqui continua bem similar em relação ao processamento de linguagem natural. A única diferença estrutural se encontra na entrada, que precisa ser particionada antes de passar pela primeira projeção linear e codificação de posição. A Figura 3.17 apresenta o modelo completo do ViT. É possível observar uma semelhança muito grande com o modelo canônico, exatamente pela ausência de convoluções. Para a classificação, utiliza-se uma camada *fully-connected* com *softmax* nas saídas.

O bloco em que está inserido o *multi-head attention* é denominado codificador de transformador (*transformer encoder*) e $L$ é a quantidade desses blocos a serem incorporados ao ViT.

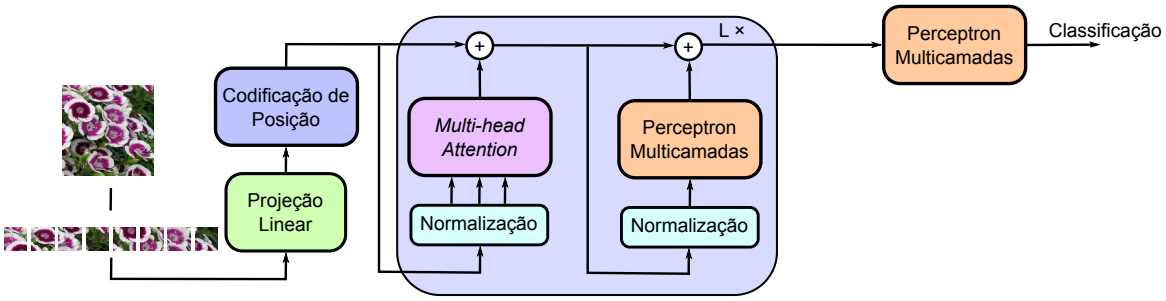



**Figura 3.17.** Classificador ViT.

## 3.3 REDUÇÃO DE ARTEFATOS COM CNNS

Outro componente essencial deste trabalho é a redução de artefatos com CNNs. Como apresentado na primeira seção do capítulo, o JPEG é uma compressão com perdas que degrada o sinal de diversas formas. Por se tratar de um padrão no armazenamento de imagens, os impactos negativos das perdas do *codec* podem ser observados nas aplicações de visão computacional. Agora será explicado como utilizar o próprio aprendizado profundo como meio para atenuar os efeitos das perdas do codificador.

A *Artifacts Reduction Convolutional Neural Network* [1] (DONG *et al.*, 2015) foi a primeira CNN a ser usada na reconstrução de imagens corrompidas pelo JPEG, o que evidenciou a eficiência dessa aplicação. Anteriormente, a aplicação de filtros não possuía boa generalização e causava outros problemas. Tendo em vista a experiência bem sucedida na geração de imagens com a SR-CNN, uma arquitetura baseada nesse modelo de *upsizing* foi desenvolvida, a AR-CNN.

### 3.3.1 SR-CNN

A SR-CNN busca gerar uma imagem de alta resolução a partir de outra de baixa resolução. Inicialmente, uma imagem $X$ é reduzida para outra de menor resolução $Y$, que em seguida é inserida no modelo $F_{SR}(Y)$, que visa aproximar a saída do *ground-truth*, $X$. A CNN desse modelo possui as seguintes finalidades: extração e representação de *patches* da imagem, mapeamento não linear e reconstrução. Cada um desses objetivos consiste em uma camada da arquitetura: $F_1$, $F_2$ e $F_3$, respectivamente.

[1] Tradução livre: Rede Neural Convolucional de Redução de Artefatos.

A extração e representação de *patches* é realizada com uma camada convolucional $W_1$, com as dimensões $c \times d_1 \times d_1 \times l_1$. Em que $c$ é a quantidade de canais da imagem, $d_1$ a dimensão do núcleo e $l_1$ a quantidade de núcleos que serão convoluídos com o sinal inteiro por vez. Após a operação, a função ReLU ainda é aplicada para se obter a saída. Em suma, temos:

$$F_1(Y) = ReLU(W_1 * Y + B_1). \tag{3.18}$$

A mesma operação é aplicada na camada de mapeamento não linear, porém, nessa etapa, a convolução vai ter um núcleo de dimensão unitária, 1×1. Aqui ocorre a projeção dos *patches* extraídos em um novo objeto de maior dimensão. Nesse caso, a saída dessa camada serão os mapas de características da imagem de alta resolução. Diversas camadas com núcleo 1×1 podem ser adicionadas para o aumento de não linearidade, porém o aumento de consumo de recursos computacionais em alguns casos torna essa estratégia inviável.

Por fim, na camada de reconstrução $F_3$ que é a saída do modelo ($F_{SR}(Y) = F_3(Y)$), a convolução é aplicada com núcleo $d_3 \times d_3$. A finalidade dessa camada é simular um filtro de média, o que é comum nesse tipo de procedimento. O ReLU é omitido nessa última camada e sua saída tem as dimensões da imagem aumentadas, apresentando um bom desempenho. Espera-se que $F_{SR}(Y)$ seja muito semelhante a $X$.

Vale a pena ressaltar que a função de custo usada no treinamento desse tipo de modelo é baseada no erro quadrático médio:

$$L_2(\Theta) = \frac{1}{m}\sum_{i=1}^{m}(F_{SR}(Y_i) - X_i)^2. \tag{3.19}$$

### 3.3.2 AR-CNN

A arquitetura apresentada anteriormente da SR-CNN utiliza três camadas convolucionais para aumentar a resolução de imagens. A AR-CNN parte do mesmo ponto, com adição de uma camada na etapa de extração de *features* do sinal.

Nesse caso, $Y$ será uma imagem comprimida com JPEG e $X$ sua versão original, sem compressão. Dessa forma, a CNN deverá aproximar sua saída $F_{AR}(Y)$ do sinal $X$. O treina-

mento supervisionado é o mesmo aplicado à SR-CNN, inclusive utilizando a mesma função de custo (vide Equação 3.19). A Figura 3.18 apresenta um esquemático da AR-CNN. Existem três camadas que são idênticas às que estão na SR-CNN, no entanto, com uma camada adicional.

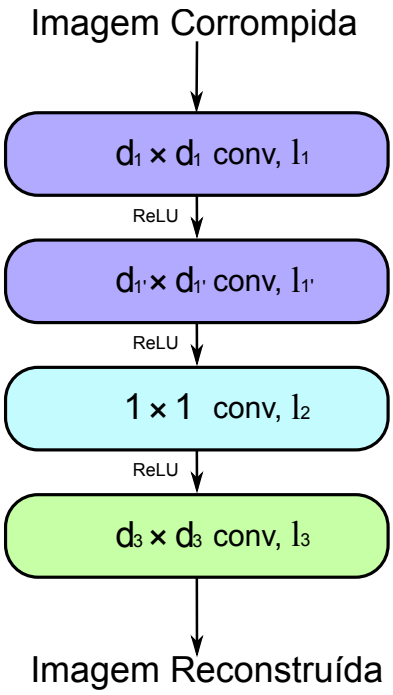



**Figura 3.18.** AR-CNN.

A ideia de adicionar $F_{1'}(Y)$ é permitir que a rede consiga obter mais *features* que aquelas que uma camada de extração só conseguiria adquirir. Essa camada é responsável por limpar padrões ruidosos do sinal, para que de fato a CNN consiga desempenhar a sua finalidade.

Aparentemente, a AR-CNN apenas aumenta a profundidade da SR-CNN. Porém, isso aconteceria somente se a camada adicional fosse a $F_2(Y)$ que, como explicado anteriormente, aumenta a inserção de não linearidade e complexidade da rede. Nesse caso, a camada adicional no modelo aumenta a obtenção de *features* de baixo nível utilizadas no *denoising*.

### 3.3.3 FBCNN

Apesar do seu desempenho satisfatório, a AR-CNN não é mais considerada eficiente em comparação com os modelos mais recentes (YANG *et al.*, 2023). Isso se deve muito pela sua limitação de operação, em que é necessário que se treine a CNN para cada fator de qualidade em que será aplicado. Esse requisito é um problema em dois sentidos: primeiro, há a exigência de

diversos treinamentos para diversos FQs; e em segundo lugar, há a necessidade de se conhecer o FQ da entrada em que será utilizada. Para contornar esses problemas, a *Flexible Blind Artifacts Removal Network* (FBCNN)[2] foi apresentada por Jiang *et al.* (2021) como uma ferramenta viável.

A arquitetura da FBCNN é constituída de quatro partes: desacoplador, preditor de fator de qualidade, controlador flexível e reconstrutor. A primeira e a última parte lembram as etapas que a AR-CNN realiza, no entanto, essa rede adiciona mais duas estruturas responsáveis pela sua flexibilidade. A Figura 3.19 apresenta o esquemático dessa arquitetura, no qual é possível observar uma dupla ramificação de RNA responsável pela classificação de FQ e geração de imagens.

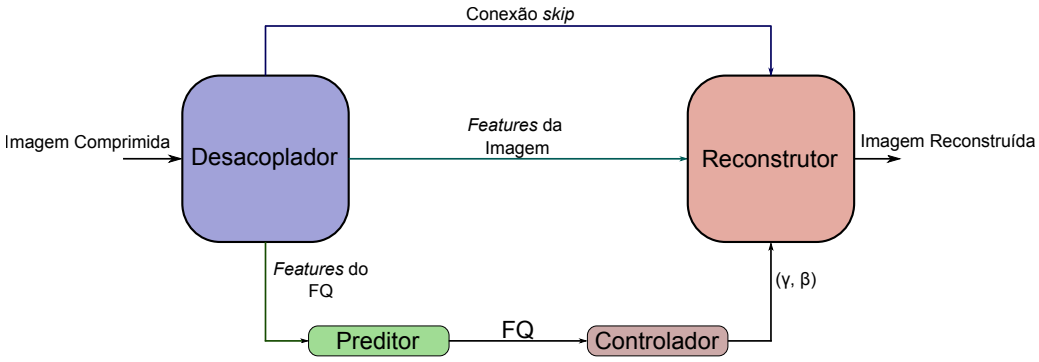



**Figura 3.19.** FBCNN.

O desacoplador funciona como um extrator de *features* de baixo nível. Para tanto, são utilizados quatro blocos residuais de camadas convolucionais com núcleo de dimensão 3×3 e *stride* 2. A cada bloco ocorre redução dos mapas de características pela metade e aumento dos canais de saída (na ordem: 64, 128, 256 e 512). Na saída de cada bloco ainda é aplicada a função ReLU, além da conexão residual com o reconstrutor. A saída do desacoplador segue para o reconstrutor e para um ramo de fator de qualidade onde, a partir de blocos residuais, são extraídas *features* de alto nível e, com uma camada de *pooling* de média global, são obtidas informações do fator de qualidade da imagem.

A próxima parte é o preditor de fator de qualidade que, como o nome já indica, infere o possível fator de qualidade do sinal de entrada. Nessa fase, as 512 saídas geradas pelo desacoplador são inseridas em uma RNA de três camadas que prediz o fator de qualidade da imagem, $FQ_{pred}$. Usando $FQ_{real}$ é possível calcular, através da Equação 3.20, a função custo

[2]Tradução livre: Rede de Remoção de Artefatos às Cegas Flexível

$L_1$ para essa etapa:

$$L_{1,FQ} = \frac{1}{m}\sum_{i=1}^{m} |FQ^i_{real} - FQ^i_{pred}|. \tag{3.20}$$

Em seguida, o controlador flexível recebe o $FQ_{pred}$ como entrada e o processa em uma RNA de quatro camadas. As três primeiras camadas do controlador geram as escalas que serão usadas pelos blocos do reconstrutor. A sua saída, contudo, gera pares de parâmetros de modulação $(\gamma, \beta)$ que serão transmitidos para o reconstrutor em cada escala.

Finalmente, o reconstrutor recebe as *features* de imagens do desacoplador, as escalas dos seus blocos e os parâmetros $(\gamma, \beta)$. Cada bloco no reconstrutor, denominados blocos de atenção ao QF, são semelhantes aos usados no desacoplador. Porém, realizam *upsizing* da entrada e modulam as *features* com os parâmetros $(\gamma, \beta)$. As operações de multiplicação e adição usadas nesse procedimento são de elemento por elemento. As Equações 3.21-3.23 são as operações realizadas no último bloco:

$$I_{pred} = \gamma \cdot F_{entrada} + \beta, \tag{3.21}$$

$$L_{1,saida} = \frac{1}{m}\sum_{i=1}^{m} |I^i_{real} - I^i_{pred}|. \tag{3.22}$$

Logo,

$$L_{1,total} = L_{1,saida} + \sigma \cdot L_{1,FQ}, \tag{3.23}$$

em que $\sigma$ é um parâmetro de balanceamento entre os custos de saída e o FQ.

## 3.4 APRENDIZADO FIM-A-FIM

Até o momento, apresentamos diversas ferramentas computacionais (compressão JPEG, redes CNN e ViT), porém a finalidade deste trabalho se encontra na junção desses componentes para implementar um classificador robusto à compressão. O aspecto base dessa abordagem se encontra no método de aprendizado fim-a-fim.

Um modelo interessante para a compreensão de um treinamento fim-a-fim é apresentado em Liu *et al.* (2015). Os autores utilizam duas saídas com diferentes atribuições para realizar segmentação facial. A ideia presente nesse trabalho mostra bem a descrição matemática que

será usada para treinar o nosso classificador.

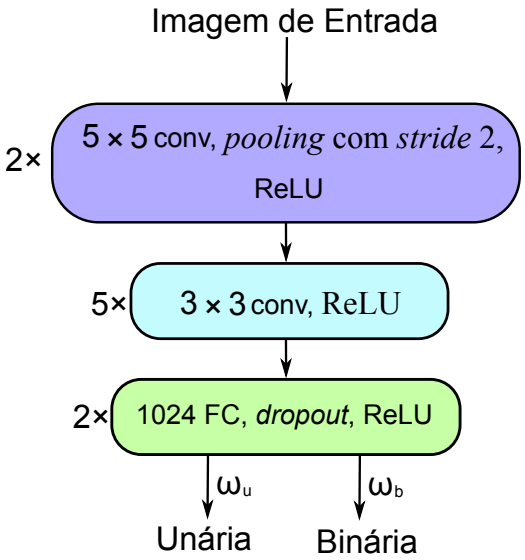



**Figura 3.20.** Arquitetura do classificador de Liu *et al.* (2015).

A arquitetura do modelo, apresentada na Figura 3.20, consiste em uma CNN com dois ramos *fully-connected* de saída. Uma das saídas é chamada de unária e a outra de binária. A saída unária é responsável por calcular o custo de atribuir um rótulo a um determinado pixel da imagem. Em relação à saída binária, é realizada a medida de custo de se criar um novo rótulo ou não para a segmentação do sinal.

A Equação 3.24 mostra o custo da saída unária dado pela função de custo *softmax*:

$$L_{soft,u} = -\log \frac{\exp((W_u^l)^T F_i(X))}{\sum_{l=1}^{L} \exp((W_u^l)^T F_i(X))}, \tag{3.24}$$

em que $X$ é a imagem de entrada, $F$ é a última saída convolucional da rede, $l$ é o rótulo, $L$ é a quantidade de rótulos, $W_u^l$ são os parâmetros da RNA unária para a classe $l$ e $i$ é o pixel a ser rotulado.

Para a saída *pairwise* (binária) será calculada a função de custo logística:

$$L_{log,b} = -log \frac{1}{1 + \sum_{l=1}^{L} exp(W_b^T F_i(X))}, \tag{3.25}$$

em que $W_b$ são os parâmetros da RNA binária.

Por fim, a função de custo total do modelo é dada pela soma das Equações 3.24 e 3.25:

$$L_{total} = L_{soft,u} + L_{log,b}. \tag{3.26}$$

As técnicas computacionais apresentadas neste capítulo são usadas em diferentes etapas deste trabalho. Inicialmente, a compressão JPEG é aplicada ao conjunto de imagens para montar o cenário de classificação de imagens comprimidas. As RNAs, com suas funções específicas (RAC ou classificação), são utilizadas na composição do nosso classificador. Por fim, o método de aprendizado fim-a-fim é usado para treinar o modelo de classificação de imagens JPEG.

CAPÍTULO 4

# METODOLOGIA

O nosso trabalho apresenta um modelo de classificação que consegue atenuar ao máximo a perda de desempenho gerada pela compressão JPEG em altas taxas. Com esse intuito, propomos um classificador de dupla ramificação: o primeiro ramo é responsável pelo tratamento do sinal e o segundo, sendo ele uma CNN ou uma ViT, é responsável pela classificação. Dessa forma, aplicamos uma abordagem fim-a-fim para treinar o modelo. No ramo RAC os artefatos tendem a ser reduzidos e *features* importantes para classificação, aumentadas. Em seguida, devido a esse procedimento, o ramo classificador consegue realizar de forma melhorada a sua função.

A arquitetura que apresentamos aqui consiste em duas estruturas dependentes uma da outra, mas com funções específicas na classificação. O treinamento conjunto de duas estruturas, uma voltada para a redução de artefatos e a outra para a classificação, pretende fazer com que aspectos que poderiam ser perdidos pela aplicação isolada de cada uma delas sejam preservados. A seguir apresentamos os ramos e, em seguida, o funcionamento da arquitetura completa do nosso modelo.

## 4.1 RAMO DE RAC: U-NET PARA REDUÇÃO DE ARTEFATOS

A U-Net foi originalmente desenvolvida para realizar segmentação de imagens médicas (RONNEBERGER *et al.*, 2015). Nessa aplicação, o treinamento da CNN consiste basicamente em usar uma imagem segmentada como *ground-truth*. A saída da U-Net também será uma imagem que, por exemplo, segmentará um câncer. Esse comportamento permite que a U-Net seja usada para detecção de objetos (WU *et al.*, 2022) e até mesmo na restauração de imagens.

A U-Net é composta de dois caminhos que guardam uma relação de simetria (são elementos diferentes de ramos): caminho de contração e caminho de expansão. A Figura 4.1

apresenta um esquemático da arquitetura da CNN. O caminho de contração é composto por quatro blocos que possuem duas camadas convolucionais de núcleo 3×3 cada, nas quais, após todas as camadas, é usada a ReLU. Na saída do bloco é aplicado o *max pooling* com *stride* 2, o que reduz pela metade a dimensão do mapa de características. Além de uma conexão com um dos blocos em paralelo ao caminho simétrico. Encerrada a contração, inicia-se o caminho de expansão, que da mesma forma é composto por quatro blocos de convolução 3×3 seguidos pela função ReLU. No lugar do *pooling* é realizado o processo inverso, a utilização de uma convolução 2×2 para o aumento de resolução. Por fim, ainda existe um bloco no final do modelo semelhante aos outros usados na rede, porém com a adição de uma camada convolucional 1×1. Em alguns casos, pode ser necessária a aplicação de *crop* (recorte) após cada camada, para evitar problemas com as bordas dos mapas de características gerados.

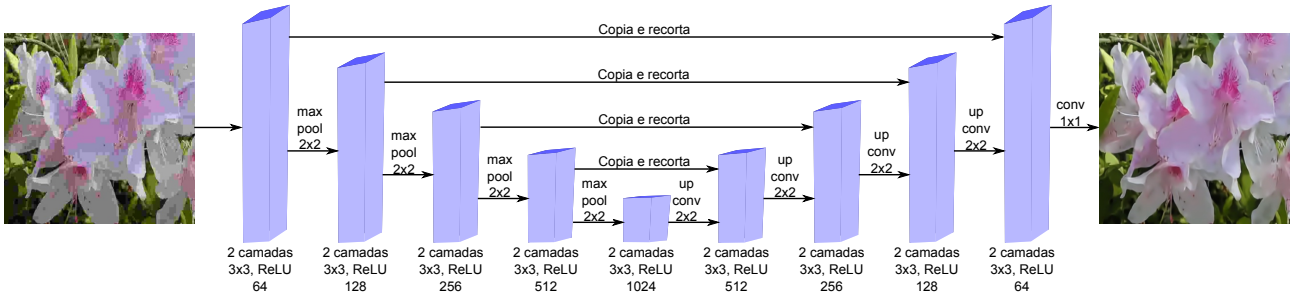



**Figura 4.1.** Arquitetura da U-Net. Exemplo de aplicação na RAC.

O caminho de contração da U-Net é responsável por extrair informações de baixa resolução do sinal de entrada. Quando a redução é realizada, muita informação também é descartada nesse processo, principalmente aquelas que seriam importantes para o caminho de expansão. A conexão paralela de copiar e recortar os *patches* gerados contorna esse problema, ao ser aplicada após cada bloco convolucional antes que o *downsampling* seja realizado. A expansão, por sua vez, extrai as *features* de alta resolução que vieram do caminho de contração ou que foram preservadas pela conexão paralela.

Em Lehtinen *et al.* (2018), a U-Net é utilizada na remoção de ruídos em imagens corrompidas. Diversas distribuições são usadas para gerar ruídos, entre elas, a gaussiana, Poisson e Bernoulli. O que os autores propõem é a ausência de imagens sem degradação para treinar o modelo. O modelo *noise2noise* (N2N) é treinado com imagens corrompidas como alvo, porém a entrada é um sinal ainda mais ruidoso. Dessa forma, é possível ensinar a rede a remover ruído mesmo sem imagens limpas.

A motivação de usarmos algo parecido ao N2N é pela ocorrência de compressão preexistente na maioria dos *datasets* de classificação de imagens. Nesse caso, é feita uma dupla compressão, uma de menor e outra de maior taxa. A nossa proposta também parte de usar a U-Net, porém inserindo-a em um classificador e utilizando rótulos com um nível menor de distorção do que o apresentado no trabalho original do N2N (LEHTINEN *et al.*, 2018).

Pelo fato da saída ser uma imagem e a U-Net conseguir fazê-la se aproximar de outra imagem escolhida como alvo, o mesmo conceito que foi empregado nos exemplos de RAC (AR-CNN e FBCNN) vistos no Capítulo 3 pode ser usado também com essa CNN, ou seja, a saída ser uma reconstrução e o alvo, por sua vez, ser o sinal sem distorção. Dessa forma, a partir de uma imagem sem ou com pouca compressão $Y$, aplicamos o codificador JPEG para obter o sinal comprimido $X$ que será a entrada da U-Net. A saída $F_{rac}(X)$ deverá ser a imagem restaurada pelo modelo U-Net RAC, $F_{rac}$. O treinamento é feito com $Y$ sendo usado como rótulo para a geração da imagem reconstruída.

A função de custo utilizada é a entropia cruzada, que se baseia no cálculo das probabilidades de saída gerada, para verificar sua proximidade em relação à saída desejada. A Equação 4.1 mostra as adaptações necessárias para que essa função de custo seja aplicada a uma saída de duas dimensões e que apresenta mais de um canal:

$$L_{CE,RAC} = -\frac{1}{s}\sum_{i=1}^{w}\sum_{j=1}^{h}\log Y_{ij} P(F_{rac}(X))_{ij}), \tag{4.1}$$

em que $w$ e $h$ são as dimensões da imagem, $s$ é o seu tamanho ($w \times h$) e $F_{rac}(X)$ é a saída da U-Net. A função $P$, dada pela Equação 4.2, é a função *softmax* usada no cálculo de probabilidade do pixel assumir determinado valor.

$$P(F_{rac}(X)_{ij}) = \frac{\exp(F_{rac}(X)_{ij})}{\sum_{m=1}^{w}\sum_{n=1}^{h}\exp(F_{rac}(X))_{mn})}. \tag{4.2}$$

O comportamento de $L_{CE,RAC}$ é de diminuir à medida que os valores da saída $F_{rac}(X)$ se aproximem do *ground truth* $Y$. Repare que a função *softmax* é calculada individualmente para cada pixel da imagem. Em seguida, todos os valores obtidos para cada pixel são somados para se obter a função de custo total. No caso da imagem que tiver mais de um canal, esse custo é calculado para cada um deles. Por exemplo, em uma imagem colorida, esse custo é calculado

para cada canal e depois somado. Dessa forma, $L_{CE,RAC}$ será computado por

$$L_{CE,RAC} = L^{red}_{CE,RAC} + L^{green}_{CE,RAC} + L^{blue}_{CE,RAC}, \tag{4.3}$$

em que $L^{red}_{CE,RAC}$, $L^{green}_{CE,RAC}$ e $L^{blue}_{CE,RAC}$ são os custos associados respectivamente aos canais vermelho, verde e azul da imagem colorida.

## 4.2 RAMO DE CLASSIFICAÇÃO

O segundo ramo do nosso modelo é responsável pela classificação da imagem. Após o processamento do sinal pelo ramo RAC espera-se que a degradação devido à compressão tenha sido atenuada. Dessa forma, os artefatos não devem gerar os prejuízos de outrora, para o ramo classificador. Além disso, a RNA é treinada para que o processo de redução de artefatos impacte minimamente as *features* que podem ser importantes na etapa de classificação, através do aprendizado fim-a-fim.

A constituição desse ramo difere da aplicada no primeiro. Não é necessária que a arquitetura seja fixa para que ele desempenhe sua função da forma esperada, pois a depender do *dataset*, uma estrutura de classificação pode ser melhor do que outra. A motivação de se usar a U-Net para a RAC ocorre porque existem aplicações que mostram sua funcionalidade em imagens duplamente corrompidas (CALVARONS, 2021). Como os *datasets* de classificação refinada, em sua maioria, apresentam um certo nível de compressão com perdas, a proposta é usar aquilo que funciona com imagens distorcidas tanto na entrada quanto nos alvos do treinamento. Isso possibilita que os experimentos sejam realizados codificando os *datasets* que já apresentam compressão. Dessa forma, a U-Net é uma arquitetura fixa no nosso modelo, pois tem a capacidade de restaurar o sinal duplamente degradado com um desempenho aceitável. Diferentemente do que acontece no ramo de classificação.

Nesse caso, é possível flexibilizar qual será a arquitetura usada no ramo de classificação, em função do seu desempenho em um determinado cenário, o que permite uma escolha mais apropriada da estrutura de um classificador a depender do *dataset* escolhido. A seguir serão apresentadas duas dessas estruturas, baseadas em dois classificadores distintos: um deles CNN e o outro ViT. Mesmo utilizando esses dois tipos diferentes de arquitetura, a extensão dos

resultados que iremos apresentar não é possível. Todavia, evidenciamos a possibilidade de se utilizar o treinamento fim-a-fim, a partir de um classificador com desempenho elevado, de modo que a compressão não seja tão prejudicial quando for aplicada.

### 4.2.1 MMAL-Net (*Multi-branch and Multi-scale Attention Learning*)

A MMAL-Net[1] (ZHANG *et al.*, 2021) é uma CNN de classificação refinada de imagens, ou seja, um classificador convolucional próprio para problemas que envolvam uma quantidade elevada de classes. A Figura 4.2 apresenta a estrutura do classificador, que possui a ResNet50 (KOONCE; KOONCE, 2021), CNN residual de 50 camadas, como a base para extração de *features* em cada ramo. A mesma ResNet é utilizada em cada ramo, ou seja, os mesmos parâmetros. A saída se trata de uma RNA *fully-connected* (FC) que realiza o processo de classificação. Essa CNN aplica uma técnica denominada SCDA (WEI *et al.*, 2017) que, a partir de um modelo pré-treinado, consegue gerar informações extras a respeito da entrada. Esse método aplica dois tipos de operações a serem realizadas nas saídas dos dois primeiros ramos, chamados de AOLM e APPM.

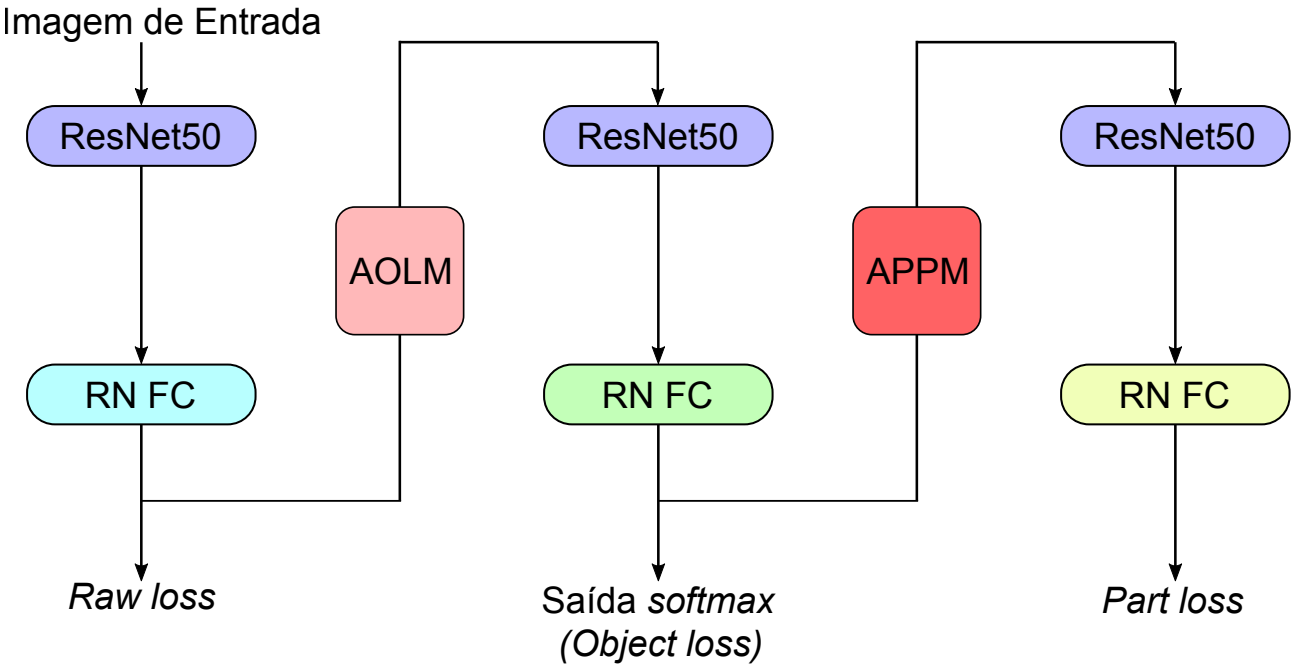



**Figura 4.2.** Arquitetura da MMAL-Net.

O AOLM (*Attention Object Location Module*[2]) é responsável por localizar os pixels do

[1]Tradução livre: Aprendizado de Atenção Multirramos e Multiescala.
[2]Tradução livre: Módulo de Localização de Objeto e Atenção.

objeto principal para classificar a imagem. Esse módulo utiliza os mapas de características de saída do primeiro ramo para calcular um limite, $t$ (Equação 4.4), que será o *treshold* para indicar se um determinado pixel faz parte ou não do objeto de interesse:

$$t = \frac{\sum_{i=k}^{c} \sum_{i=1}^{w} \sum_{j=1}^{h} F_k(X)_{ij}}{w \times h}, \tag{4.4}$$

em que $F(X)_k$ é o k-ésimo mapa de características da saída do ramo, $w$ e $h$ são as dimensões desse e $c$ é a quantidade de canais (número de mapas de características).

A partir de $t$ é possível para cada posição determinar se o valor daquele elemento corresponde ou não ao objeto desejado. Dessa forma, é possível gerar uma máscara $M$ através da expressão:

$$M'_{ij} = \begin{cases} 1, & \text{se } F(X)_{ij} > t \\ 0, & \text{caso contrário} \end{cases}. \tag{4.5}$$

Uma forma de melhorar o desempenho desse mascaramento é utilizar as saídas das últimas duas camadas convolucionais para fazer uma interseção entre as duas. Sendo assim, considere $M'$ e $M''$ como as máscaras geradas usando as últimas duas camadas de convolução, logo é possível obter a máscara de identificação do objeto a partir de:

$$M = M' \cap M''. \tag{4.6}$$

O outro módulo, denominado APPM (*Attention Part Proposal Module*[3]), consiste no mesmo processo de localização de objetos do AOLM, porém que aplica janelas aos mapas de características, ao invés de usá-las completas. Dessa forma, o valor de *treshold* é:

$$t_J = \frac{\sum_{i=1}^{c} \sum_{i=1}^{w_J} \sum_{j=1}^{h_J} F_{J,k}(X)_{ij}}{w_J \times h_J}. \tag{4.7}$$

O erro (custo) de cada ramo é calculado a partir da entropia cruzada. A saída do classificador é apurada por uma camada *softmax*. O valor total da função de custo é dado pela soma dos custos de todos os ramos, isto é:

$$L_{CE,CLASS} = L_{raw} + L_{object} + L_{part}. \tag{4.8}$$

[3]Tradução livre: Módulo Proposição de Atenção de Partes

### 4.2.2 *Compact Convolutional Transformer*(CCT)

O classificador baseado no modelo ViT que escolhemos é o CCT[4] (HASSANI *et al.*, 2021). Esse modelo busca contornar o paradigma do uso de *big data* através da compactação da ViT, tendo em vista que é uma arquitetura que necessita de muitos dados para ser treinada.

Os modelos compactos reduzem a quantidade de parâmetros para serem otimizados, o que proporciona uma quantidade menor de dados necessários para o treinamento. No caso da CCT, a quantidade de módulos *multi-head attention* é reduzida significativamente em relação ao ViT original, além do número de camadas da RNA e das projeções lineares realizadas.

A principal diferença da CCT é a redução de parâmetros em relação ao ViT, porém outra distinção marcante é a aplicação de CNN no processo de *tokenização* da entrada. No ViT a imagem é dividida em *patches* que passarão por uma operação linear para serem inseridos no bloco transformador. Na versão compacta convolucional, esse processo é substituído pelas operações de convolução e *max pooling*. Outra operação adicional da CCT é o *sequence pooling* aplicado após o transformador. A Figura 4.3 apresenta um esquemático da CCT com essas diferenças citadas.

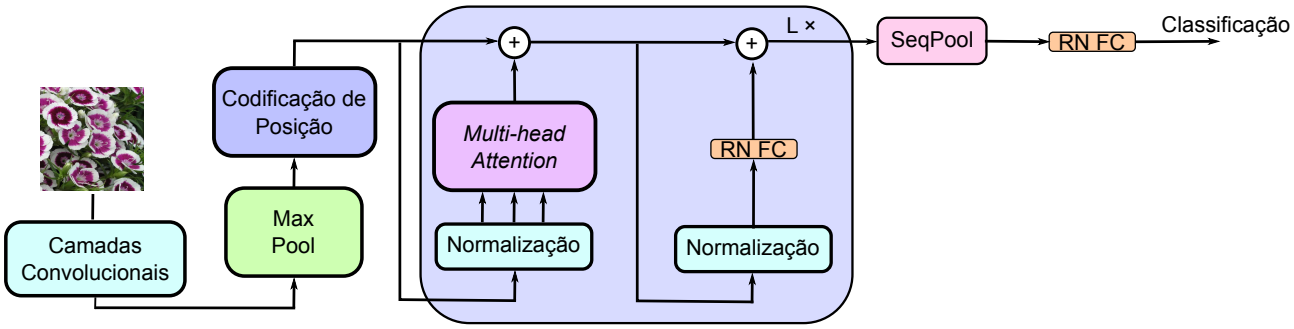



**Figura 4.3.** Arquitetura da CCT.

Para a função de custo, utilizamos uma versão suavizada da entropia cruzada (Equação 4.9), em que inicialmente se deve calcular a $L_{CE}$ e a sua média em relação à quantidade de classes $L_{CE,CLASS}$.

$$L_{CE,CLASS} = (1 - \mu) \cdot L_{CE} + \frac{\mu}{N_{CLASS}}. \tag{4.9}$$

em que $\mu$ é a suavização escolhida e $N_{CLASS}$ é o número de classes do modelo.

No nosso caso, utilizamos a CCT-14/7×2 que possui 14 *transformer encoders* (o $L$ é 14)

[4]Tradução livre: Transformador Convolucional Compacto

e 7 camadas convolucionais para tokenização com um núcleo 2×2.

## 4.3 N2N-CAR

A partir dessas três RNAs apresentadas para o ramo de RAC e classificação, em (REIS *et al.*, 2023) realizamos um primeiro procedimento para amenizar os impactos negativos da compressão na classificação, a partir da redução de artefatos. Para tanto, utilizamos o modelo RAC apresentado na Seção 4.1, denominado N2N-CAR. Foram realizados testes para 10 fatores de qualidade do JPEG com o sinal comprimido e reconstruído (aplicado ao RAC). Em um cenário de fatores de qualidade (FQs) muito baixos (intervalo de 1 a 10), foi possível reduzir a perda de desempenho significativamente.

A primeira etapa de desenvolvimento dessa ferramenta foi comprimir os *datasets* utilizados para os 10 FQs citados. Em seguida, utilizamos a mesma RNA apresentada na Seção 4.1 e a treinamos com esses *datasets*, um treinamento para os FQs de 1 a 10. Dessa forma, obtivemos um modelo especializado em restaurar imagens comprimidas para cada FQ. A etapa de teste consistiu em utilizar os modelos de classificação (MMAL-Net e CCT) com seus respectivos parâmetros apresentados pelos seus autores (ASLANI *et al.*, 2019; HASSANI *et al.*, 2021) para verificar a acurácia, após a restauração de cada *dataset*.

A Figura 4.4 apresenta a estrutura da metodologia do N2N-CAR aplicada na restauração de imagens para classificação. Os detalhes dos *datasets* utilizados serão apresentados no próximo capítulo. No total, 20 acurácias foram obtidas, 10 para cada tipo de RNA (MMAL-Net e CCT). Essas acurácias serão utilizadas para fins de comparação com o método fim-a-fim abordado neste trabalho.

## 4.4 MODELO DE DUPLA RAMIFICAÇÃO

Agora que vimos as duas estruturas que podem compor o nosso classificador, é possível sintetizar a RNA de dupla ramificação. As Figuras 4.5 e 4.6 apresentam os esquemáticos do modelo *dual*. O ramo de RAC para as duas arquiteturas é o mesmo, contudo o ramo de classificação pode ser comutado a depender do *dataset* que será usado para a inferência. Essa

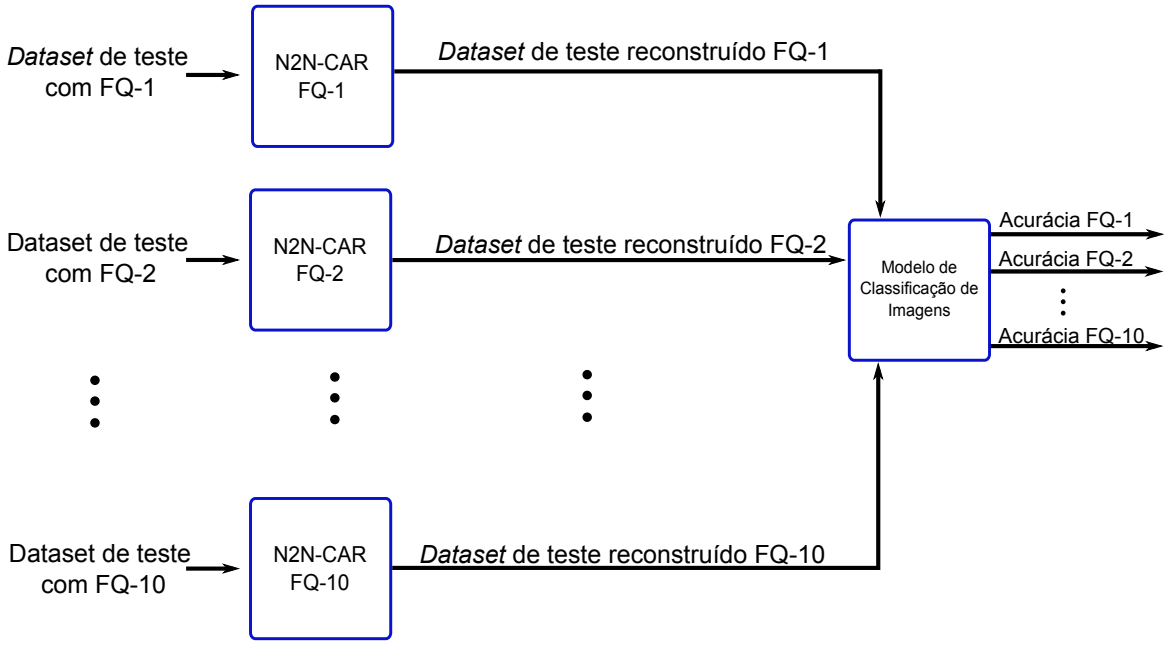



**Figura 4.4.** Estrutura da metodologia abordada em (REIS *et al.*, 2023) com o restaurador N2N-CAR.

flexibilidade permite que a classificação alcance um desempenho melhor para cada cenário.

A saída do primeiro ramo consiste em uma versão adaptada da redução de artefatos para o processo de classificação. Os mapas de características resultantes dessa etapa são a entrada do segundo ramo, que se aproveitará de um sinal menos degradado. Problemas associados ao impacto negativo da RAC serão apresentados nos próximos capítulos, todavia a RAC "pura", como indicada na Seção 3.1.2, quando reduz os artefatos intrínsecos do JPEG, tende a gerar artefatos adicionais. O ideal pode ser uma espécie de balanceamento entre os artefatos da RAC e do JPEG. Essa é uma das justificativas para fazer o acoplamento entre esses dois processos.

Após cada ramo, é calculada a respectiva função de custo atribuída àquela etapa, $L_{CE,RAC}$ e $L_{CE,CLASS}$. Em qualquer caso, o valor da função de custo que será utilizada no treinamento é calculada pela soma de ambas:

$$L_{CE,TOTAL} = L_{CE,RAC} + L_{CE,CLASS}. \tag{4.10}$$

Dessa forma, o passo 3 do algoritmo de retro-propagação apresentado na Seção 3.2.1 será calcular $\sigma^{(L)}$ a partir de $y^{(t)}$ aplicado na função de custo dada pela Equação 4.10, ou seja,

$$\sigma^{(L)} = L_{CE,TOTAL}. \tag{4.11}$$

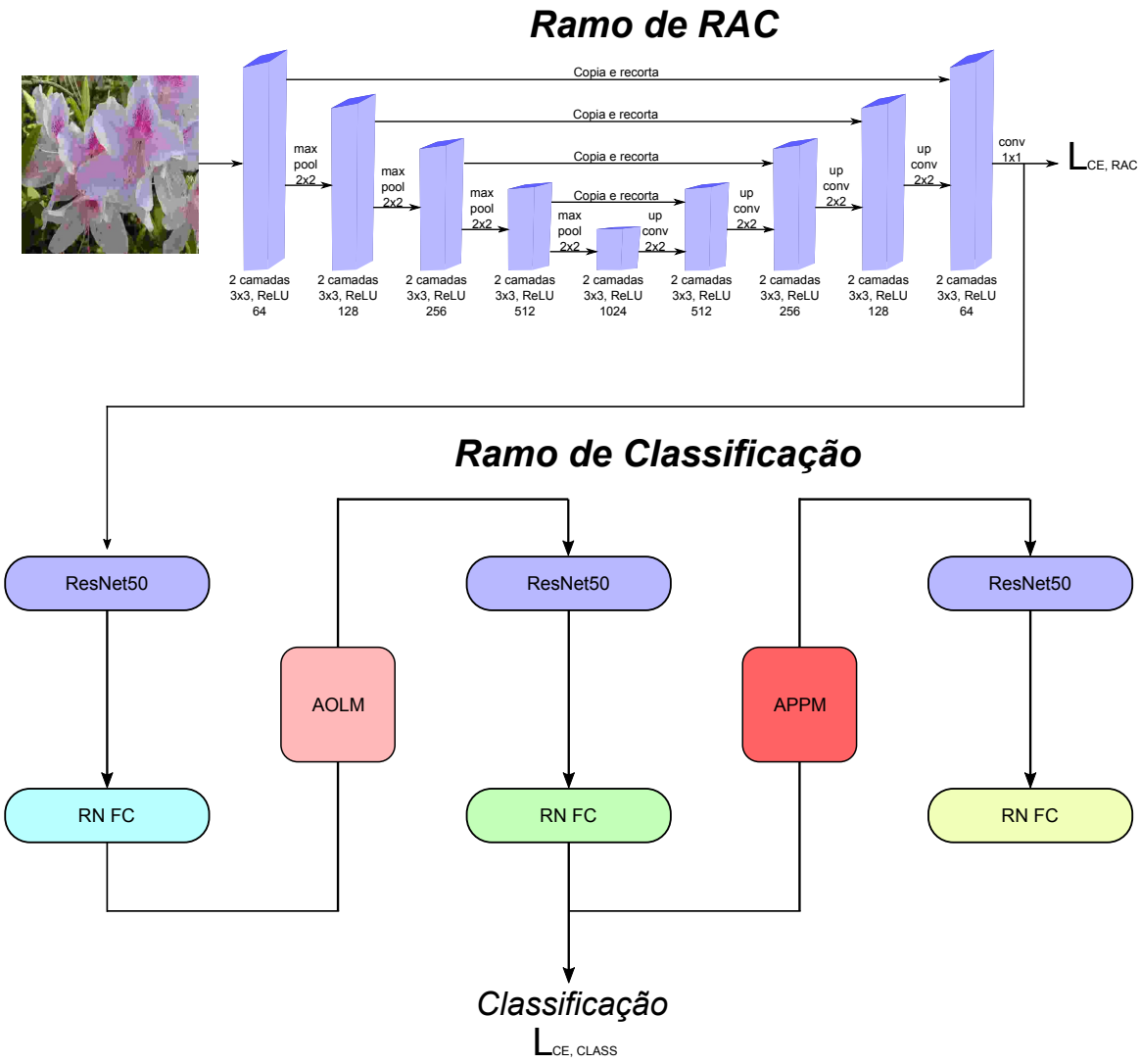



**Figura 4.5.** Arquitetura da RNA dual com ramo de classificação MMAL-Net.

O comportamento fim-a-fim da nossa proposta ocorre porque os dois ramos conectados tem seus parâmetros conjuntamente ajustados durante o treinamento, usando não apenas a função de custo de saída do classificador ($L_{CE,CLASS}$) para otimizar os parâmetros da rede, mas também é utilizada a função de custo do ramo responsável pela redução de artefatos. Essa meta dupla do modelo pretende se aproveitar da possível correlação entre reduzir artefatos e melhoria de desempenho do classificador, para ir além do uso isolado desses procedimentos. Pelo fato da saída final coincidir com a do ramo de classificação, esse terá prioridade no treinamento devido à retro-propagação. Como a nossa proposta final é de um classificador, então essa é a intenção.

A aplicação desses dois ramos é justificada pela possível correlação que pode existir entre reduzir artefatos e melhorar o desempenho da classificação. Se o objetivo da primeira etapa for

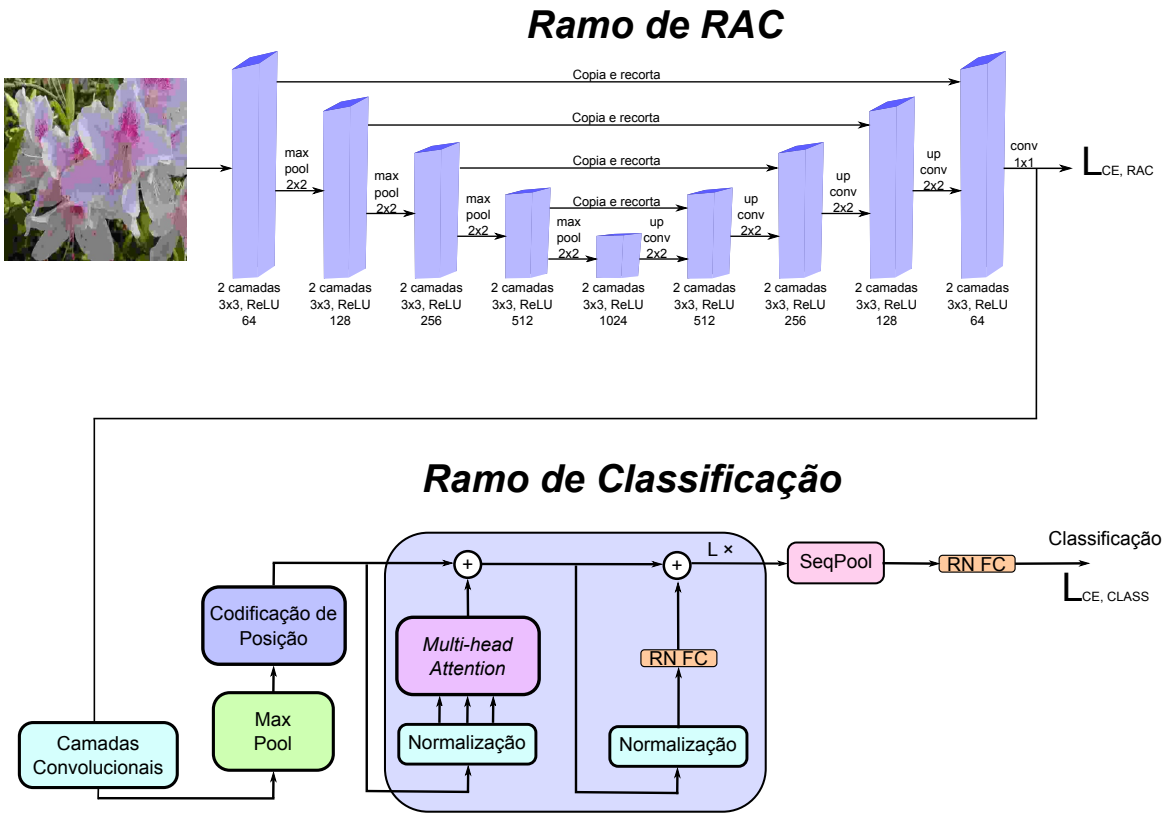



**Figura 4.6.** Arquitetura da RNA dual com ramo de classificação CCT.

alcançado e existir essa correlação, a segunda etapa também cumprirá com sua função, porém dessa vez obtendo um melhor desempenho caso não houvesse nenhum procedimento anterior (RAC). Sendo assim, é necessário haver o treinamento correspondente de cada ramo, ou seja, RAC e classificação.

Além disso, o ramo de classificação também é treinado para se adaptar às entradas que serão imagens reconstruídas. Nesse caso, os parâmetros da rede serão otimizados para processar e fazer a inferência de um sinal com outras características que se distinguem das imagens sem ou com compressão. O ramo de RAC nesse sentido irá reduzir a capacidade de generalização, uma vez que o uso isolado da RAC em um *dataset* para ser testado na classificação não influenciará nos objetivos de cada RNA separada. O desempenho do classificador irá se manter para as imagens originais, ser degradado para as corrompidas e melhorado para as reconstruídas. No entanto, ao se estabelecer o treinamento em conjunto entre as duas etapas, há uma tendência do modelo de se especializar em um determinado cenário de imagens, no qual para qualquer outro pode haver degradação. Os problemas decorrentes desse fenômeno serão apresentados no capítulo de resultados.

Esse ainda se beneficia de informações das imagens originais que serão utilizadas como *ground truth* na saída do ramo RAC, diferentemente do que ocorre no método de aumento da base de dados (*data augmentation*). Quando as imagens comprimidas adicionadas são muito distintas das imagens originais, isso pode prejudicar o treinamento, fazendo com que o modelo não atenue muito a perda de desempenho pela compressão e ainda degrade o classificador no cenário sem compressão. A nossa proposta consiste exatamente em abordar o caso no qual a alta taxa de compressão seja aplicada, ou seja, sinal comprimido muito diferente do original.

CAPÍTULO 5

# EXPERIMENTOS

A RNA proposta segue a descrição apresentada no capítulo anterior. A dupla ramificação com uma parte de RAC e outra parte de classificação consiste na abordagem fim-a-fim que usamos no nosso classificador de imagens JPEG. Resta agora apresentar como conduzimos os experimentos para treinar o modelo e testar a nossa metodologia. Os dois *datasets* utilizados, os hiperparâmetros de cada RNA e o comportamento do treinamento serão apresentados nas próximas seções.

## 5.1 *DATASET* CALTECH 200 CUB

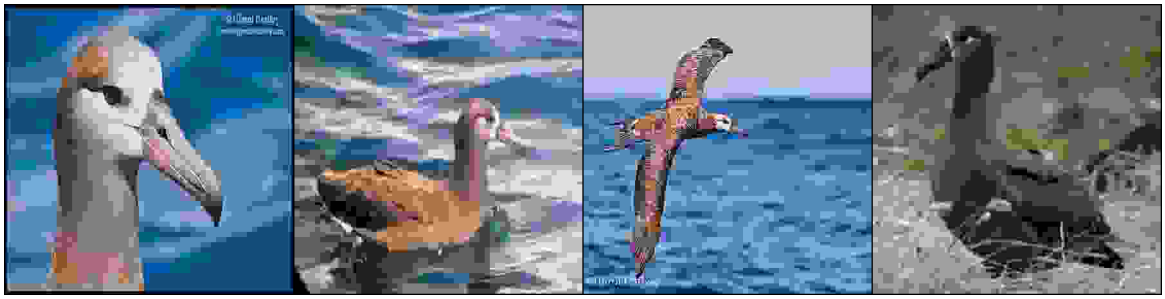
O Caltech-UCSD Birds-200-2011 (WAH *et al.*, 2011) é um *dataset* composto por imagens de pássaros de diferentes espécies. No total, 200 classes fazem parte dessa base de dados, o que a torna um problema de classificação refinada. A divisão utilizada no *dataset* consiste em 5994 exemplos para treinamento e 5794 para teste. Apesar da ausência de um conjunto de validação previamente definido, esse *dataset* é muito usado para testar desempenho de modelos classificadores.

O ramo de classificação que usamos na nossa RNA para aplicar esse *dataset* foi a MMAL-Net. Conforme o artigo que emprega este modelo, a acurácia na referida tarefa é de 89.6%. Os valores de pesos (parâmetros) aprendidos pelo modelo e que levaram a este resultado são usados para fins de inicialização do nosso treinamento.

A Figura 5.1 apresenta amostras de imagens extraídas do *dataset* Cub. As primeiras quatro imagens são do *dataset* original, no qual não aplicamos nenhuma compressão (lembrando que isso não significa ausência de compressão, a qual pode ter sido realizada por quem disponibilizou os dados). Na sequência, imagens comprimidas com JPEG e FQ igual a 5 (entradas da RNA). Por fim, as saídas intermediárias do classificador que, nesse caso, são os mapas de

características do ramo de RAC.

(a) Imagens originais



(b) Imagens comprimidas

(c) Saídas do ramo de RAC

**Figura 5.1.** Amostras do *dataset* 200 Cub.

Para os nossos experimentos geramos 10 *datasets* Cub comprimidos com FQs de 1 até 10.

## 5.2 *DATASET* OXFORD 102 FLOWER

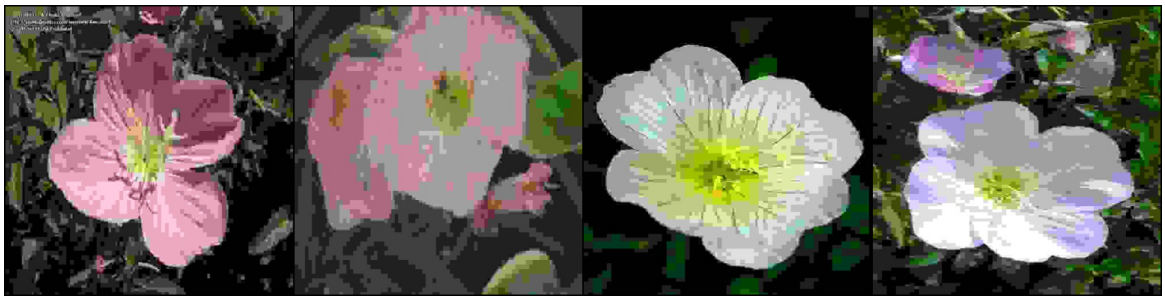
Em relação à CCT, usamos o *dataset* 102 Flower (NILSBACK; ZISSERMAN, 2008) para testá-lo. A transformada visual compacta atingiu uma acurácia de 99.9% para essa base de dados, de fato um dos melhores resultados obtidos nesse *dataset* (STOJNIC, 2024). Da mesma forma que na MMAL-Net, aqui usamos o ramo de classificação pré-treinado com os parâmetros da CCT obtidos em Hassani *et al.* (2021).

A qualidade desse *dataset* é notadamente superior em relação ao 200 Cub. Os sinais

possuem uma maior resolução e uma menor quantidade de artefatos da compressão. Há uma divisão entre imagens de treino, teste e validação com 6149, 1020 e 1020 imagens para cada conjunto, respectivamente. No total, são 102 classes que variam entre 40 e 258 imagens em cada.

Assim como para o *dataset* 200 Cub, a Figura 5.2 apresenta algumas amostras do *dataset* 102 Flower. As imagens são comprimidas com o mesmo FQ de 5. Nas saídas do ramo RAC existe uma diferença peculiar entre os dois modelos e *datasets*. Aqui, neste caso, as saídas apresentaram um comportamento mais próximo ao da reconstrução da imagem. Geramos 10 *datasets* com FQs de 1 a 10.

(a) Imagens originais



(b) Imagens comprimidas

(c) Saídas do ramo de RAC

**Figura 5.2.** Amostras do *dataset* 102 Flower.

## 5.3 DETALHES DE IMPLEMENTAÇÃO

A partir dos *datasets* comprimidos e originais, foi realizado inicialmente um treinamento isolado da rede U-Net de RAC. Sendo assim, foram obtidas 10 parametrizações dessa CNN para cada FQ, de 1 até 10. Esses pesos são o pré-treino da rede dual. Considerando os dois conjuntos de imagens, no total foram gerados 20 pré-treinos. Esse pré-treinamento é igual ao treinamento da N2N-CAR apresentada em Reis *et al.* (2023). Nesse sentido, o treinamento da U-Net de RAC é inicializado com os mesmos pesos do N2N-CAR usado para restaurar os *datasets* comprimidos.

O pré-treino do ramo de classificação foram os pesos disponibilizados pelos respectivos autores de cada modelo (ZHANG *et al.*, 2021; HASSANI *et al.*, 2021). Nesse caso, apenas um tipo de parametrização de cada RNA foi aplicado, aquela com o melhor resultado para o conjunto de imagens original. Para o *dataset* Cub os parâmetros aprendidos para acurácia de 89.6% e para o *dataset* Flower aqueles aprendidos para acurácia de 99.9%. Esses foram os parâmetros das redes usados na inicialização do nosso ramo de classificação.

A Tabela 5.1 apresenta os hiperparâmetros das RNAs de dupla ramificação que testamos. Os dois otimizadores são baseados no método de descida por gradiente, apresentado na Seção 3.2.1.

**Tabela 5.1.** Hiperparâmetros de treinamento do classificador para os dois tipos de ramos de classificação.

| **Ramo de RAC** | **Ramo de Classificação** | **Detalhes de Implementação** | | | | | |
|---|---|---|---|---|---|---|---|
| | | ***Entrada*** | ***Otimizador*** | ***Taxa de aprendizado*** | ***Batch*** | ***Épocas*** | ***# Params*** |
| U-Net | CCT-14/7×2 | 384×384 | ADAM | 0.001 | 4 | 15 | 22,7M |
| U-Net | MMAL-Net | 448×448 | SGD | 0.001 | 6 | 66 | 26,7M |

Treinamos cada classificador com uma GPU NVIDIA GeForce RTX 3090 de 24GB de memória RAM, em um computador com um processador Intel Core i9-10900K CPU de 3.70GHz com 10 núcleos e 20 threads. Além disso, o programa de implementação e treinamento dos modelos foi desenvolvido em linguagem de programação Python.

## 5.4 TREINAMENTO

Partindo dos parâmetros pré-treinados e dos hiperparâmetros apresentados previamente, realizamos o treinamento de 10 RNAs de dupla ramificação para cada *dataset*. No total, 20 modelos treinados nas imagens comprimidas com fatores de qualidade de 1 a 10.

No treinamento, o *dataset* é carregado da seguinte forma: as imagens de entrada (comprimidas), *ground truth* da RAC (imagens originais) e rótulos de classificação na saída da RNA. Cada *batch* é organizado dessa forma e o treinamento se segue até se completar todas as épocas.

As funções de custo de cada ramo são somas para se calcular o custo total. Os otimizadores estocásticos que usamos atualizam os parâmetros da rede a cada iteração, ou seja, o custo é calculado para um determinado *batch* e na sequência já ocorre o ajuste de pesos. No final de cada época ainda pode ocorrer a validação da RNA, entretanto isso depende do *dataset* usado. A seguir serão apresentadas as curvas de custos do treinamento, algumas de destaque estarão nas próximas seções, enquanto o restante pode ser encontrado no Apêndice A.

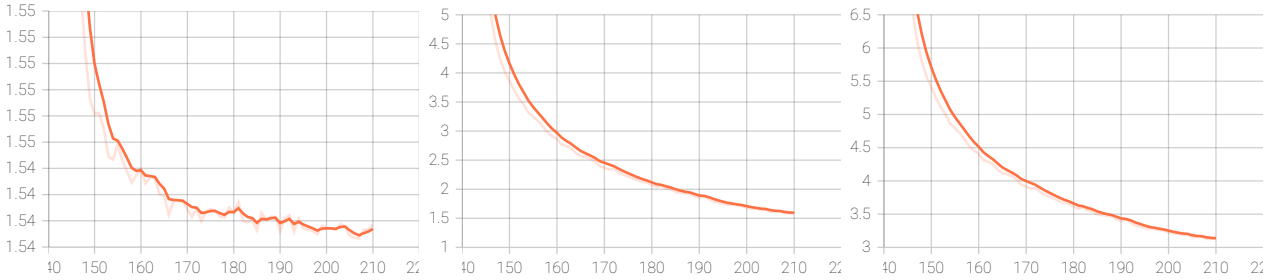
### 5.4.1 Curvas de Custo da MMAL-Net

Um dos principais componentes de análise do treinamento de uma RNA é a curva que apresenta a função de custo em relação às iterações ou épocas. A ideia é que o comportamento dessa curva seja de decaimento com aspectos assintóticos (VELIKANOV; YAROTSKY, 2021). Quando algo foge dessa regra, possivelmente o aprendizado não está acontecendo e é necessário verificar os detalhes de implementação da rede. Se desconsiderarmos falhas de desenvolvimento no algoritmo, uma possível causa de um treinamento mal sucedido seriam as inconsistências no *dataset*. Por exemplo, alvos rotulados errados geram confusão, impedindo que a otimização ocorra adequadamente (KAPLAN *et al.*, 2021).

Apesar de ser uma importante ferramenta de análise, esses gráficos não apresentam nenhuma informação definitiva do desempenho da arquitetura treinada. Isso significa que, mesmo o custo assumindo um comportamento esperado, o desempenho pode não ser o desejado. Diversos cenários em que essa distorção ocorre são muito comuns: problemas de *underfitting* e *overfitting* (POTHUGANTI, 2018), inconsistências na base de dados, desbalanceamentos

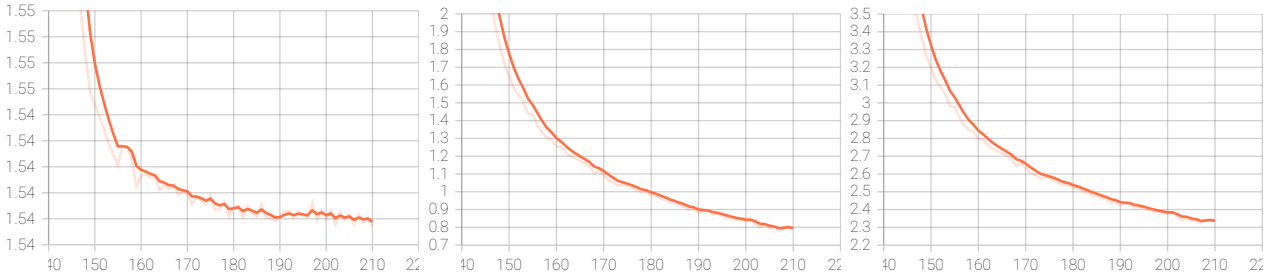
(WANG *et al.*, 2021) e falta de dados para o treinamento (BRIGATO; IOCCHI, 2021) são alguns exemplos de causas.

Geramos gráficos de custos para todos os treinamentos que realizamos com o objetivo de verificar o aprendizado dos modelos. O custo total e os dois termos de custos parciais usados na otimização foram obtidos. As Figuras 5.3-5.5 apresentam na ordem o custo da RAC, o custo da classificação e o custo total. Na saída do ramo RAC, o custo é obtido através da média de cada iteração que ocorre durante uma época. O mesmo é válido para o ramo de classificação e para o cálculo total de perdas.



(a) Custo da RAC (b) Custo da Classificação (c) Custo total

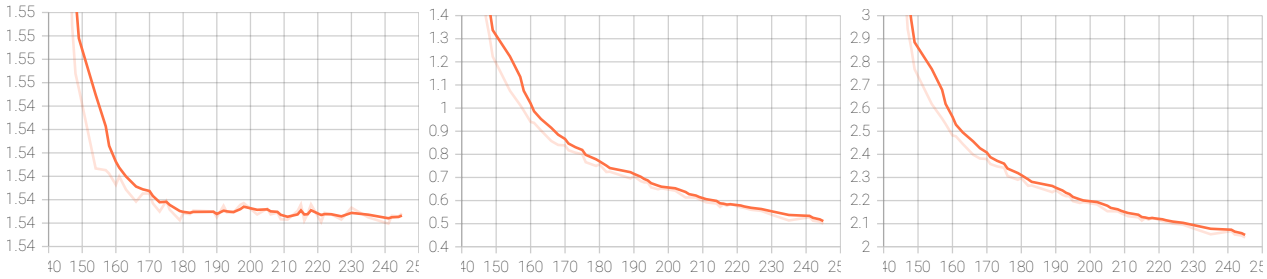
**Figura 5.3.** Treinamento com FQ igual a 1 (MMAL-Net). Em laranja claro, os valores nominais de custo, e em laranja escuro a suavização da curva.



(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura 5.4.** Treinamento com FQ igual a 5 (MMAL-Net). Em laranja claro, os valores nominais de custo, e em laranja escuro a suavização da curva.

Em todas as curvas observamos o decaimento esperado para um treinamento bem-sucedido. Contudo, a redução do custo de classificação foi mais expressiva do que a apresentada no custo da RAC. Um dos motivos para esse comportamento pode ser a quantidade de parâmetros existentes em cada um dos ramos. A estrutura de RAC tem 35 vezes menos parâmetros, o que torna as perdas geradas por ela bem menores (ZHOU, 2021). Outra justificativa seria o fato do pré-treinamento da rede de classificação ter sido realizado apenas com as imagens originais. Nesse sentido, quando imagens comprimidas são inseridas, sua curva de aprendizado terá uma




(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura 5.5.** Treinamento com FQ igual a 10 (MMAL-Net). Em laranja claro, os valores nominais de custo, e em laranja escuro a suavização da curva.

tendência de ser mais acentuada.

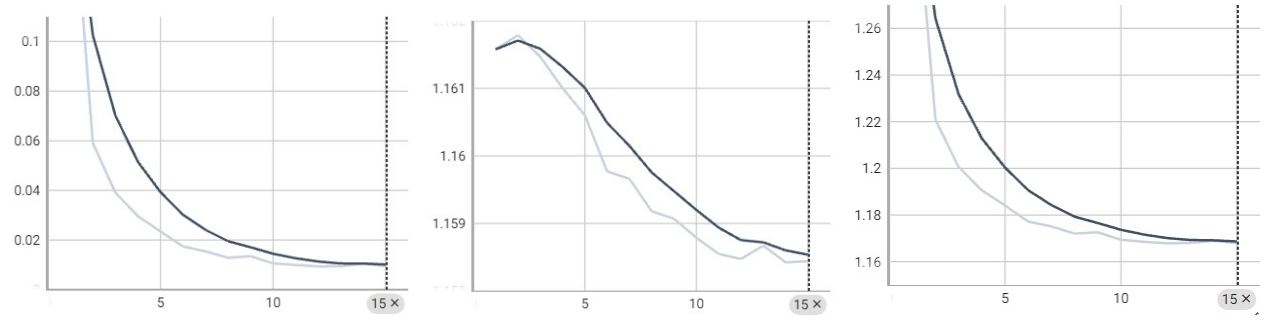
O custo total, como descrito anteriormente, é a soma dos custos de cada ramo. Para cada FQ esse valor pode variar drasticamente, unicamente por conta do custo de classificação. Quando o FQ é 1, o custo médio de classificação começa com um valor maior do que 5. Esse valor inicial tende a diminuir à medida que o FQ aumenta. Isso reflete em que a RNA de classificação precisa aprender mais sobre a entrada do que a RNA de RAC, o que está de acordo com o pré-treinamento aplicado, no qual o ramo da U-Net já vem com os parâmetros que foram treinados com as mesmas entradas que ele terá novamente.

Em contrapartida, o valor de perdas da RAC não é modificado da mesma forma em função do FQ, continuando no mesmo intervalo de decaimento com diferenças mais associadas aos aspectos do formato da curva. A redução dos custos também continua no mesmo valor. Nesse caso, a adequação das imagens geradas pela RAC não é tão significativa quanto o aprendizado do classificador em relação às suas entradas, comportamento justificado muito pelo método de pré-treinamento utilizado.

### 5.4.2 Custos da CCT

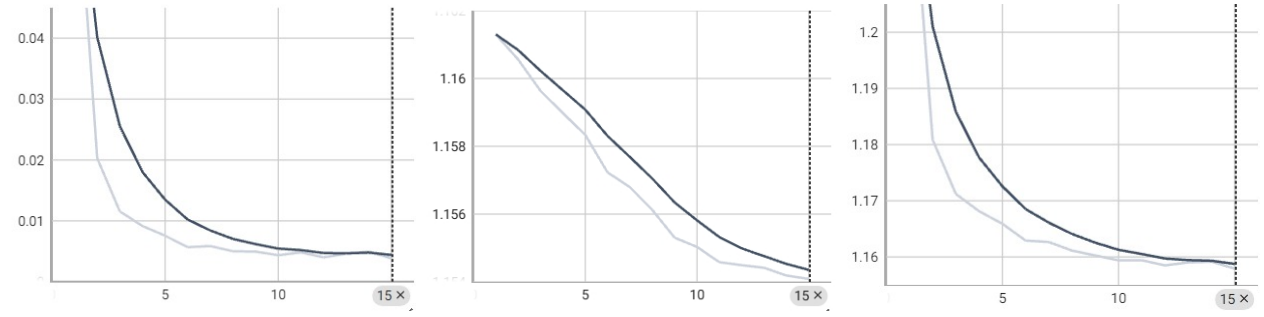
O treinamento da RNA com o ramo de classificação CCT pode ser analisado através dos gráficos das Figuras 5.6-5.8. O comportamento das curvas não é muito semelhante ao que foi apresentado no MMAL-Net: aqui, os valores médios do custo da classificação é que são quase constantes em função do FQ. Nos custos da RAC teremos sua redução à medida que o FQ aumenta, ou seja, ocorre o oposto entre os ramos dos modelos. No ramo RAC, o intervalo entre os valores iniciais e finais de custo também tende a diminuir ao longo dos experimentos.

O custo de classificação possui um decaimento menor ao longo das épocas do que o custo da RAC. Um possível motivo para que essa variação muito pequena ocorra está na qualidade do *dataset*. Como indicado anteriormente, o 102 Flower possui imagens com menor degradação por compressão prévia que, mesmo comprimidas a altas taxas, conseguem manter uma qualidade elevada. O classificador possivelmente não percebe uma diferença tão significativa em tratar uma imagem com ou sem compressão dupla.




(a) Custo da RAC (b) Custo da Classificação (c) Custo total

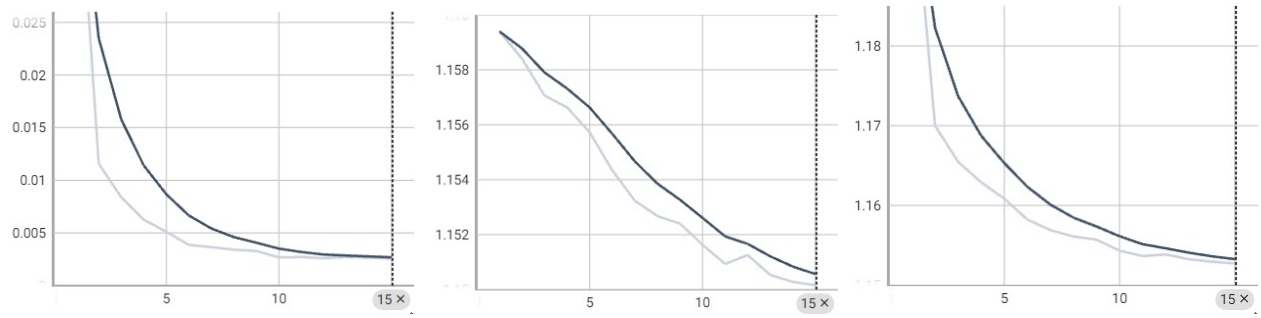
**Figura 5.6.** Treinamento com FQ igual a 1 (CCT). Em cinza claro, os valores nominais de custo, e em cinza escuro a suavização da curva.




(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura 5.7.** Treinamento com FQ igual a 5 (CCT). Em cinza claro, os valores nominais de custo, e em cinza escuro a suavização da curva.

No ramo da U-Net o custo aqui também se distingue por apresentar uma redução na ordem de $10^{-1}$ o que para a MMAL-Net só é possível visualizar na ordem de $10^{-2}$. Essa informação é muito importante, pois evidencia que mesmo com uma quantidade bem menor de parâmetros, pode haver casos nos quais, devido à estrutura do ramo de classificação, a U-Net consiga superar a redução de custos em valor absoluto. Além disso, as curvas indicam que a RAC pode adaptar mais as imagens que devam ser geradas para que a arquitetura de classificação consiga desempenhar melhor sua tarefa. Em decorrência disso, o classificador




(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura 5.8.** Treinamento com FQ igual a 10 (CCT). Em cinza claro, os valores nominais de custo, e em cinza escuro a suavização da curva.

precisa aprender menos, pois as entradas foram melhor adequadas para maximizar os ganhos da sua atividade.

Outra diferença em relação ao MMAL-Net consiste na quantidade de épocas necessárias para treinar a CCT, como veremos no próximo capítulo, isso se deve a uma maior robustez do Transformador Visual em classificar imagens comprimidas com JPEG. A classificação com CNNs é degradada drasticamente em altas taxas de compressão, comportamento evidenciado em diversas bibliografias citadas no Capítulo 2.

CAPÍTULO 6

# RESULTADOS

As curvas de custo são importantes ferramentas gráficas para a análise do treinamento de RNAs. Porém, as suas características não apresentam informações importantes para determinar completamente os seus desempenhos. Evidentemente, quando os custos não assumem o comportamento de decaimento esperado, o aprendizado do modelo falhou. No entanto, a ocorrência da redução de custos não implica que o modelo consiga generalizar os seus resultados.

Dessa forma, outras métricas de avaliação são necessárias para verificar o desempenho dos modelos. No nosso caso, a classificação refinada de imagens não exige muitas dessas métricas para a análise. A presença de muitas classes faz com que métricas como precisão, *recall* ou F1-Score sejam redundantes quando a acurácia já foi calculada.

## 6.1 RNA DE DUPLA RAMIFICAÇÃO COM A MMAL-NET

A principal forma de se avaliar o desempenho do nosso modelo é por meio de uma curva de acurácia junto ao conjunto de teste pelo fator de qualidade. A partir desse gráfico, é possível comparar o resultado da nossa técnica em relação às outras. A Figura 6.1 apresenta quatro curvas de acurácia em função do FQ, uma para cada método de classificação apresentado e inclusive para a acurácia da MMAL-Net com o *dataset* original. Com exceção da RNA que propomos neste trabalho, todas as outras técnicas utilizam a mesma MMAL-Net com os mesmos parâmetros para o teste.

O resultado para imagens somente comprimidas, em vermelho, indica o pior dos casos, em que, com FQ igual a 1, a acurácia fica abaixo de 5% e, mesmo para um FQ de 10, esse valor não ultrapassa os 60%. Claramente estamos apresentando uma compressão agressiva em que os artefatos causam muita distorção nas imagens. A Tabela 6.1 apresenta os valores de PSNR e taxa de compressão (TC) para cada *dataset* comprimido. Conseguimos alcançar, para

certos FQ, taxas com um valor próximo de 20, o que representa uma redução de 95% dos bits do tamanho do sinal. Porém, a PSNR, a análise subjetiva (presença ou não de artefatos da compressão) e a acurácia de teste indicam uma degradação significativa devido à codificação, tornando a utilização do sinal, nessas condições, inviável.

A estratégia de restaurar a base de imagens a partir de um modelo de RAC independente é aplicada em Reis *et al.* (2023) e a sua curva está em azul. A motivação de utilizar uma RNA de RAC (N2N-CAR) é aumentar a qualidade do sinal de forma que os problemas, citados anteriormente para o caso no qual se tem apenas compressão, sejam mitigados. O efeito esperado é indicado desde o mais baixo FQ, em que já é possível alcançar cerca de 40% de aumento na acurácia. Para o FQ igual a 10, esse valor se aproxima consideravelmente do nível nominal de classificação com o *dataset* original.

**Tabela 6.1.** Métricas da compressão JPEG no conjunto de imagens 200 Cub.

| **FQ** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **TC** | 22.29 | 22.28 | 21.3 | 19.57 | 18.01 | 16.67 | 15.55 | 14.61 | 13.78 | 13.05 |
| **PSNR(dB)** | 28.99 | 28.99 | 29.02 | 29.41 | 29.81 | 30.22 | 30.63 | 30.99 | 31.34 | 31.69 |

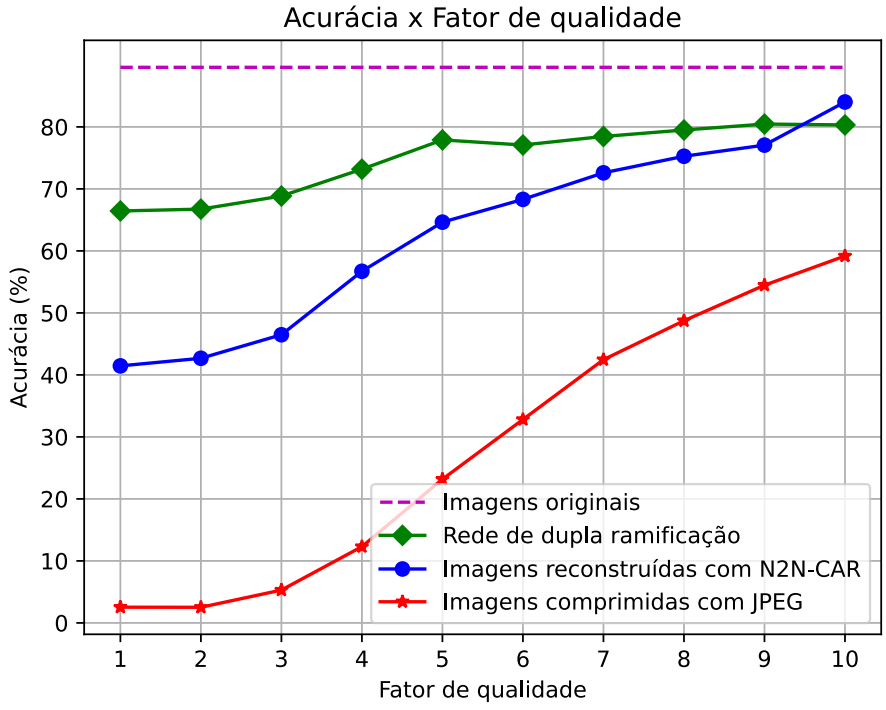



**Figura 6.1.** Gráfico da acurácia pelo fator de qualidade para a RNA de dupla ramificação com a MMAL-Net.

Por fim, a curva da RNA que propomos aqui (em verde) consegue superar a técnica passada quando a compressão é bastante agressiva. Um aumento de aproximadamente 25% quando o FQ é 1, com uma diminuição progressiva à medida que a qualidade aumenta. Os parâmetros de inicialização da rede de dupla ramificação no ramo de RAC são iguais aos parâmetros aprendidos na N2N-CAR. Dessa forma, a melhora de desempenho obtida por essa rede ocorre por conta da abordagem fim-a-fim, em que o treinamento da estrutura de RAC vai ser direcionado para melhorar o resultado da classificação.

O comportamento das curvas de se aproximarem quando o FQ aumenta se justifica pela própria tendência de a redução na compressão gerar aumento na acurácia. Certamente, em um determinado FQ, todas as curvas se encontrarão no valor original de acurácia, com exceção da característica do modelo dual. Esse resultado condiz com o fato do classificador ser outro, diferente do que acontece para os demais casos. Nessa situação, o classificador, além de possuir uma arquitetura diferente, também apresentará um desempenho diferente. Quando o FQ atinge o valor de 10, o método de reconstrução ultrapassa a rede fim-a-fim, a qual aparenta apresentar um comportamento de estabilização. Nesse caso, em um cenário de normalização com imagens cada vez menos comprimidas, a estrutura adicional de RAC na rede parece não manter um comportamento de melhora do classificador.

Na média, a nossa rede obteve uma acurácia maior de 46.54% e 11.97% em comparação com as imagens comprimidas e as que foram restauradas, respectivamente. Resultado que apresenta uma melhora significativa em um cenário de compressão agressiva.

## 6.2 RNA DE DUPLA RAMIFICAÇÃO COM A CCT

Em relação à RNA com a estrutura de classificação CCT, temos um comportamento semelhante, porém com alguns aspectos distintos. A Figura 6.2 apresenta as curvas de desempenho para essa arquitetura dual. Igualmente ao que ocorre para a MMAL-Net, um FQ baixo reduz a acurácia do modelo e, à medida que esse fator aumenta, a métrica que utilizamos também aumenta. No entanto, a primeira distinção está na redução de acurácia, que para esse caso não é tão grande, algo em torno de 15%. Essa característica pode indicar uma maior robustez do Transformador Visual em operar com imagens comprimidas, além de proporcionar uma solução

ainda mais consistente para esse problema.

As curvas em azul e em vermelho estão bem próximas uma da outra, o que indica uma menor melhora no resultado através da restauração das imagens com o N2N-CAR. Apesar de ainda ocorrer um aumento na acurácia, essa técnica é limitada, pois o classificador não é tão afetado pela degradação do sinal. Sendo assim, a RAC não consegue ser tão eficiente nesse sentido.

**Tabela 6.2.** Métricas da compressão JPEG no conjunto de imagens 102 Flower.

| **FQ** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **TC** | 5.99 | 5.99 | 5.74 | 5.32 | 4.94 | 4.61 | 4.35 | 4.08 | 3.85 | 3.62 |
| **PSNR(dB)** | 29.14 | 29.14 | 29.17 | 29.60 | 30.04 | 30.52 | 30.97 | 31.39 | 31.81 | 32.18 |

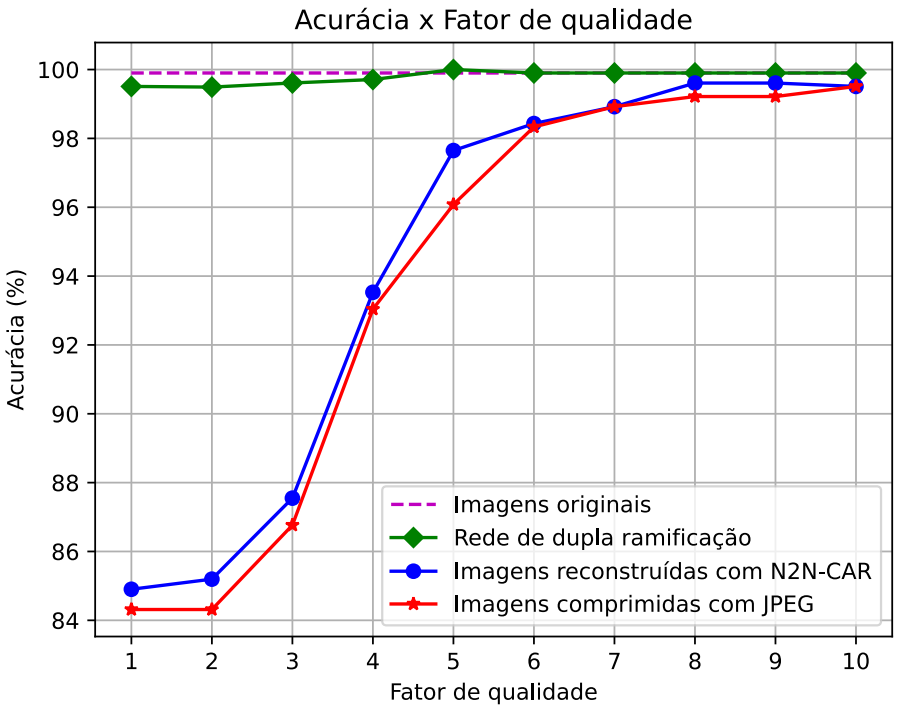



**Figura 6.2.** Gráfico da acurácia pelo fator de qualidade para a RNA de dupla ramificação com a CCT.

A Tabela 6.2 também indica que a TC não é tão expressiva como para o caso anterior, isso acontece devido à compressão prévia aplicada na base de imagens. Uma vez que as imagens originais já apresentam codificação, se a compressão não for expressiva, a tendência é de que a TC não seja tão significativa. As PSNRs para esse *dataset* se mantém em uma faixa parecida, mas sem impactar negativamente a RNA da mesma forma que no 200 Cub. Aqui também é

possível observar que, com FQ de 10, o desempenho do classificador CCT já converge para o valor original.

A nossa rede, por sua vez, ultrapassa em desempenho os outros métodos, anulando quase que completamente a influência prejudicial da compressão na classificação. A curva em verde indica que, desde o valor mínimo de FQ a rede de dupla ramificação preserva a acurácia original, chegando a 100% para FQ igual a 5. Esse resultado evidencia uma maior compatibilidade entre os ramos da RNA, fazendo com que, mesmo quando o desempenho se aproxima do original, o ramo de RAC não reduza o desempenho do ramo de classificação.

Essa boa interação entre as partes da RNA é responsável por ampliar em 5.81% e 5.29% a acurácia em relação à compressão pura e o método do N2N-CAR, respectivamente. Apesar desse aumento não ser tão grande quanto para o outro *dataset*, por conta da robustez que o modelo apresenta, é possível afirmar que o desempenho nesse caso também é positivo, pelo fato da máxima mitigação de perdas possível do classificador.

CAPÍTULO 7

# CONCLUSÃO

As imagens comprimidas fazem parte da composição de grande parte das bases de dados de IA devido à necessidade de sua aplicação em diversas situações. Por esse motivo, não é uma operação que pode ser desprezada e, em muitos casos, ela será um fator determinante para o desempenho dos modelos de aprendizado profundo. É de fundamental importância conhecer até quais limites a codificação pode ser utilizada e quais são os impactos negativos decorrentes de extrapolar a compressão. Ainda é necessário avaliar quais são as possíveis técnicas para contornar uma possível redução no desempenho dos modelos.

Esta dissertação busca resolver esse problema em uma aplicação específica de visão computacional denominada de classificação refinada de imagens. O que propomos foi uma arquitetura de RN capaz de atenuar, para os *datasets* testados, a degradação das RNs devido à compressão. O que especificamente desenvolvemos foi um modelo robusto à compressão de imagens, nos casos em que os limites de comprimir o sinal e manter o desempenho do modelo são ultrapassados.

A nossa ideia foi aplicar o treinamento fim-a-fim em uma rede de dupla ramificação, com um dos ramos focados na RAC. Os nossos resultados indicam que existe correlação entre o desempenho do nosso classificador e a redução de artefatos. Porém, outros fatores característicos do treinamento fim-a-fim também podem ser responsáveis pela melhoria apresentada. Uma hipótese é que o ramo de RAC mantém *features* importantes da imagem degradada para a classificação. Isso é indicado pela superioridade da nossa RN em comparação com a restauração do *dataset* com o N2N-CAR, o que equivale à aplicação direta do RAC nas imagens de teste.

A interação compatível entre os ramos da RN faz com que, nos dois conjuntos de imagens utilizados em duas estruturas de classificação diferentes, o aumento de desempenho fosse significativo. No *dataset* 200 Cub foi possível observar o aumento de algumas dezenas na acurácia, mesmo em comparação com a técnica do N2N-CAR. Em relação ao *dataset* 102 Flower não

houve sequer um impacto negativo na classificação devido à compressão. Em ambos os casos, houve relativo sucesso em classificar imagens comprimidas com JPEG em níveis baixos de FQ.

Apesar dos bons resultados alcançados, a nossa RN apresenta uma limitação para ser utilizada na prática: a necessidade de treinar uma RN para cada FQ. Por essa razão, um próximo trabalho seria aplicar algo parecido à FBCNN (AMARANAGESWARAO *et al.*, 2020) no ramo de RAC para flexibilizar o uso de uma única RN. Isso reduziria o consumo computacional de treinamento e dispensaria a necessidade de armazenamento de uma grande quantidade de parâmetros para cada FQ, além de proporcionar uma maior praticidade e economia de tempo. Outra proposta de trabalho futuro é a de inverter a geração de imagens pela classificação, ou seja, utilizar a classificação em um treinamento fim-a-fim para beneficiar a geração de imagens. Algo parecido ao que é realizado pela FBCNN, porém estendido a outros tipos de classes da imagem que não sejam necessariamente o FQ. Isso, possivelmente, possibilitaria a implementação de modelos de RAC mais eficientes.

# REFERÊNCIAS BIBLIOGRÁFICAS


ABBASS, H. A. Speeding up backpropagation using multiobjective evolutionary algorithms. *Neural Computation*, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 15, n. 11, p. 2705–2726, 2003. Citado na página 7.

AGGARWAL, C. C. *Neural networks and deep learning: a textbook*. [S.l.]: Springer, 2018. Citado na página 17.

AGGARWAL, R.; SOUNDERAJAH, V.; MARTIN, G.; TING, D. S.; KARTHIKESALINGAM, A.; KING, D.; ASHRAFIAN, H.; DARZI, A. Diagnostic accuracy of deep learning in medical imaging: a systematic review and meta-analysis. *NPJ digital medicine*, Nature Publishing Group UK London, v. 4, n. 1, p. 65, 2021. Citado na página 4.

AMARANAGESWARAO, G.; DEIVALAKSHMI, S.; KO, S.-B. Blind compression artifact reduction using dense parallel convolutional neural network. *Signal Processing: Image Communication*, Elsevier, v. 89, p. 116009, 2020. Citado 3 vezes nas páginas 3, 6, and 59.

AMER, H.; SHATERIAN, S.; YANG, E.-h. "deep selector-jpeg: Adaptive jpeg image compression for computer vision in image classification with human vision criteria". *arXiv preprint arXiv:2302.09560*, 2023. Citado na página 5.

ASLANI, S.; DAYAN, M.; STORELLI, L.; FILIPPI, M.; MURINO, V.; ROCCA, M. A.; SONA, D. Multi-branch convolutional neural network for multiple sclerosis lesion segmentation. *NeuroImage*, Elsevier, v. 196, p. 1–15, 2019. Citado 2 vezes nas páginas 7 and 39.

BALDI, P.; SADOWSKI, P. J. Understanding dropout. *Advances in neural information processing systems*, v. 26, 2013. Citado na página 18.

BENBARRAD, T.; ELOUTOUATE, L.; ARIOUA, M.; ELOUAAI, F.; LAANAOUI, M. D. Impact of image compression on the performance of steel surface defect classification with a cnn. *Journal of Sensor and Actuator Networks*, MDPI, v. 10, n. 4, p. 73, 2021. Citado 2 vezes nas páginas 2 and 4.

BRIGATO, L.; IOCCHI, L. A close look at deep learning with small data. In: IEEE. *2020 25th International Conference on Pattern Recognition (ICPR)*. [S.l.], 2021. p. 2490–2497. Citado na página 49.

CALVARONS, A. F. Improved noise2noise denoising with limited data. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2021. p. 796–805. Citado na página 35.

CILIMKOVIC, M. Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, v. 15, n. 1, 2015. Citado na página 18.

DONG, C.; DENG, Y.; LOY, C. C.; TANG, X. Compression artifacts reduction by a deep convolutional network. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 576–584. Citado 3 vezes nas páginas 3, 6, and 25.

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. Citado na página 22.

EBRAHIMI, F.; CHAMIK, M.; WINKLER, S. Jpeg vs. jpeg 2000: an objective comparison of image encoding quality. In: SPIE. *Applications of Digital Image Processing XXVII*. [S.l.], 2004. v. 5558, p. 300–308. Citado na página 2.

FOI, A.; KATKOVNIK, V.; EGIAZARIAN, K. "pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images". *IEEE transactions on image processing 16.5*, 2007. Citado na página 6.

GHOLAMALINEZHAD, H.; KHOSRAVI, H. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020. Citado na página 19.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016. Citado na página 20.

HASSANI, A.; WALTON, S.; SHAH, N.; ABUDUWEILI, A.; LI, J.; SHI, H. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021. Citado 4 vezes nas páginas 38, 39, 45, and 47.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado na página 7.

HUSSAIN, A. J.; AL-FAYADH, A.; RADI, N. Image compression techniques: A survey in lossless and lossy algorithms. *Neurocomputing*, Elsevier, v. 300, p. 44–69, 2018. Citado na página 1.

HUYNH-THU, Q.; GHANBARI, M. The accuracy of psnr in predicting video quality for different video scenes and frame rates. *Telecommunication Systems*, Springer, v. 49, p. 35–48, 2012. Citado na página 1.

JAKULIN, A. Baseline jpeg and jpeg2000 artifacts illustrated. *unpublished. Available: http://zeus. fri. unilj. si/˜ aleks/jpeg/artifacts. htm*, 2002. Citado na página 12.

JIANG, J.; ZHANG, K.; TIMOFTE, R. Towards flexible blind jpeg artifacts removal. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2021. p. 4997–5006. Citado 2 vezes nas páginas 6 and 28.

JO, Y.-Y.; CHOI, Y. S.; PARK, H. W.; LEE, J. H.; JUNG, H.; KIM, H.-E.; KO, K.; LEE, C. W.; CHA, H. S.; HWANGBO, Y. Impact of image compression on deep learning-based mammogram classification. *Scientific Reports*, Nature Publishing Group UK London, v. 11, n. 1, p. 7924, 2021. Citado 2 vezes nas páginas 2 and 4.

KAPLAN, S.; HANDELMAN, D.; HANDELMAN, A. Sensitivity of neural networks to corruption of image classification. *AI and Ethics*, Springer, p. 1–10, 2021. Citado na página 48.

KOONCE, B.; KOONCE, B. Resnet 50. *Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization*, Springer, p. 63–72, 2021. Citado na página 36.

LAU, W.-L.; LI, Z.-L.; LAM, K.-K. Effects of jpeg compression on image classification. *International journal of remote sensing*, Taylor & Francis, v. 24, n. 7, p. 1535–1544, 2003. Citado na página 2.

LEHTINEN, J.; MUNKBERG, J.; HASSELGREN, J.; LAINE, S.; KARRAS, T.; AITTALA, M.; AILA, T. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*, 2018. Citado 2 vezes nas páginas 33 and 34.

LI, Z.; SA, C. D.; SAMPSON, A. Optimizing jpeg quantization for classification networks. *arXiv preprint arXiv:2003.02874*, 2020. Citado na página 5.

LIST, P.; JOCH, A.; LAINEMA, J.; BJONTEGAARD, G.; KARCZEWICZ, M. Adaptive deblocking filter. *IEEE transactions on circuits and systems for video technology*, IEEE, v. 13, n. 7, p. 614–619, 2003. Citado na página 6.

LIU, S.; YANG, J.; HUANG, C.; YANG, M.-H. Multi-objective convolutional learning for face labeling. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 3451–3459. Citado 3 vezes nas páginas iv, 29, and 30.

MIANO, J. *Compressed image file formats: Jpeg, png, gif, xbm, bmp*. [S.l.]: Addison-Wesley Professional, 1999. Citado na página 1.

NILSBACK, M.-E.; ZISSERMAN, A. Automated flower classification over a large number of classes. In: *Indian Conference on Computer Vision, Graphics and Image Processing*. [S.l.: s.n.], 2008. Citado na página 45.

PENG, Y.; HE, X.; ZHAO, J. Object-part attention model for fine-grained image classification. *IEEE Transactions on Image Processing*, IEEE, v. 27, n. 3, p. 1487–1500, 2017. Citado na página 15.

POTHUGANTI, S. Review on over-fitting and under-fitting problems in machine learning and solutions. *Int. J. Adv. Res. Electr. Electron. Instrum. Eng*, v. 7, p. 3692–3695, 2018. Citado na página 48.

PRASANNA, Y. L.; TARAKARAM, Y.; MOUNIKA, Y.; SUBRAMANI, R. Comparison of different lossy image compression techniques. In: IEEE. *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*. [S.l.], 2021. p. 1–7. Citado na página 1.

RAMACHANDRAN, P.; PARMAR, N.; VASWANI, A.; BELLO, I.; LEVSKAYA, A.; SHLENS, J. Stand-alone self-attention in vision models. *Advances in neural information processing systems*, v. 32, 2019. Citado na página 24.

REIS, A. dos; HUNG, E.; SILVA, D. Deep learning-based compression artifacts reduction for jpeg image classification. In: SIMAS, E.; FERREIRA, D. D.; OLIVEIRA, L. R. (Ed.). *Anais do XVI Congresso Brasileiro de Inteligência Computacional (CBIC'2023)*. Salvador, BA: SBIC, 2023. p. 1–7. Citado 5 vezes nas páginas iv, 39, 40, 47, and 54.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. [S.l.], 2015. p. 234–241. Citado na página 32.

ROTH, S.; GEPPERTH, A.; IGEL, C. Multi-objective neural network optimization for visual object detection. *Multi-Objective Machine Learning*, Springer, p. 629–655, 2006. Citado na página 7.

SAYOOD, K. *Introduction to data compression*. [S.l.]: Morgan Kaufmann, 2017. Citado na página 12.

STOJNIC, R. *Image Classification on Flowers-102*. 2024. Disponível em: <https://paperswithcode.com/sota/image-classification-on-flowers-102>. Citado na página 45.

SUDRE, C. H.; LI, W.; VERCAUTEREN, T.; OURSELIN, S.; CARDOSO, M. J. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In: SPRINGER. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3*. [S.l.], 2017. p. 240–248. Citado na página 8.

TAUBMAN, D. S.; MARCELLIN, M. W.; RABBANI, M. Jpeg2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, Society of Photo-Optical Instrumentation Engineers, v. 11, n. 2, p. 286–287, 2002. Citado na página 9.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017. Citado na página 22.

VELIKANOV, M.; YAROTSKY, D. Explicit loss asymptotics in the gradient descent training of neural networks. *Advances in Neural Information Processing Systems*, v. 34, p. 2570–2582, 2021. Citado na página 48.

WAH, C.; BRANSON, S.; WELINDER, P.; PERONA, P.; BELONGIE, S. *The Caltech-UCSD Birds-200-2011 Dataset*. [S.l.], 2011. Citado na página 44.

WALLACE, G. K. The jpeg still picture compression standard. *Communications of the ACM*, ACM New York, NY, USA, v. 34, n. 4, p. 30–44, 1991. Citado na página 3.

WANG, C.; ZHOU, J.; LIU, S. Adaptive non-local means filter for image deblocking. *Signal Processing: Image Communication*, Elsevier, v. 28, n. 5, p. 522–530, 2013. Citado na página 6.

WANG, L.; HAN, M.; LI, X.; ZHANG, N.; CHENG, H. Review of classification methods on unbalanced data sets. *IEEE Access*, IEEE, v. 9, p. 64606–64628, 2021. Citado na página 49.

WANG, X.; GIRSHICK, R.; GUPTA, A.; HE, K. Non-local neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2018. p. 7794–7803. Citado na página 24.

WEI, X.-S.; LUO, J.-H.; WU, J.; ZHOU, Z.-H. Selective convolutional descriptor aggregation for fine-grained image retrieval. *IEEE transactions on image processing*, IEEE, v. 26, n. 6, p. 2868–2881, 2017. Citado na página 36.

WU, X.; HONG, D.; CHANUSSOT, J. Uiu-net: U-net in u-net for infrared small object detection. *IEEE Transactions on Image Processing*, IEEE, v. 32, p. 364–376, 2022. Citado na página 32.

YANG, E.-H.; AMER, H.; JIANG, Y. Compression helps deep learning in image classification. *Entropy*, MDPI, v. 23, n. 7, p. 881, 2021. Citado na página 3.

YANG, G.; LIN, L.; WU, C.; WANG, F. Dual-domain learning for jpeg artifacts removal. In: SPRINGER. *International Conference on Neural Information Processing*. [S.l.], 2023. p. 556–568. Citado na página 27.

YOON, J.; CHO, N. I. Jpeg artifact reduction based on deformable offset gating network controlled by a variational autoencoder. *IEEE Access*, IEEE, v. 11, p. 30282–30291, 2023. Citado na página 7.

ZHANG, F.; LI, M.; ZHAI, G.; LIU, Y. Multi-branch and multi-scale attention learning for fine-grained visual categorization. In: SPRINGER. *MultiMedia Modeling: 27th International Conference, MMM 2021, Prague, Czech Republic, June 22–24, 2021, Proceedings, Part I 27*. [S.l.], 2021. p. 136–147. Citado 2 vezes nas páginas 36 and 47.

ZHOU, Z.-H. Why over-parameterization of deep neural networks does not overfit? *Science China. Information Sciences*, Springer Nature BV, v. 64, n. 1, p. 116101, 2021. Citado na página 49.

# APÊNDICES

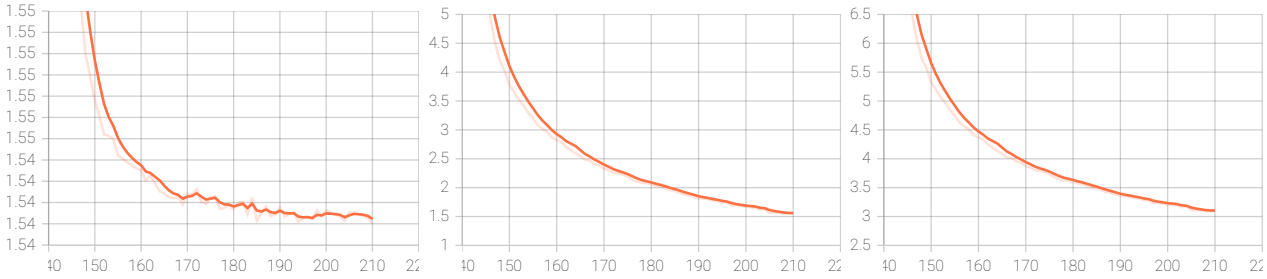
APÊNDICE A

# CURVAS DE CUSTO

A seguir estão as curvas de custo geradas durante o treinamento dos modelos apresentados no Capítulo 5. Em laranja claro, os valores nominais de custo, e em laranja escuro a suavização da curva.

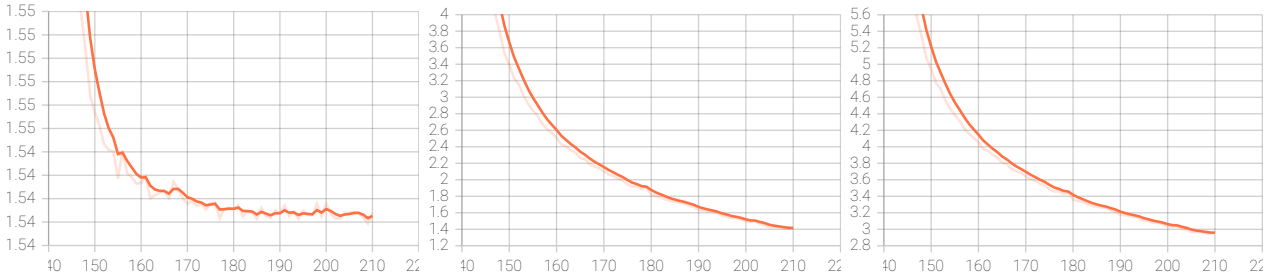
## A.1 MMAL-NET

Para a RN de dupla ramificação com o ramo de classificação sendo a MMAL-Net, as curvas de custos serão:



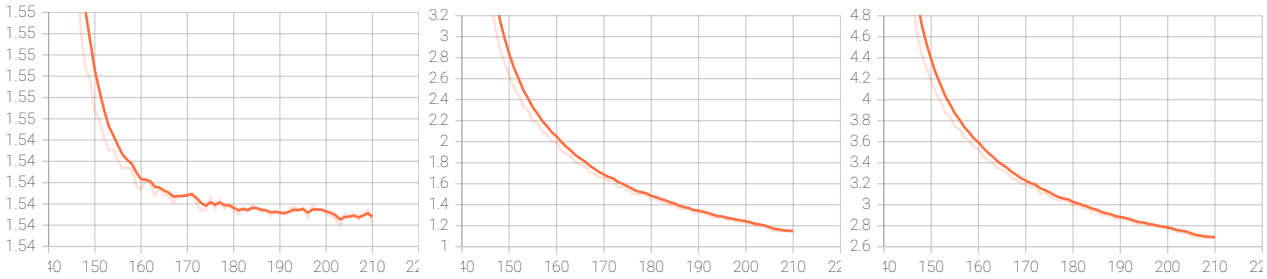
(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.1.** Treinamento com FQ igual a 2 (MMAL-Net).



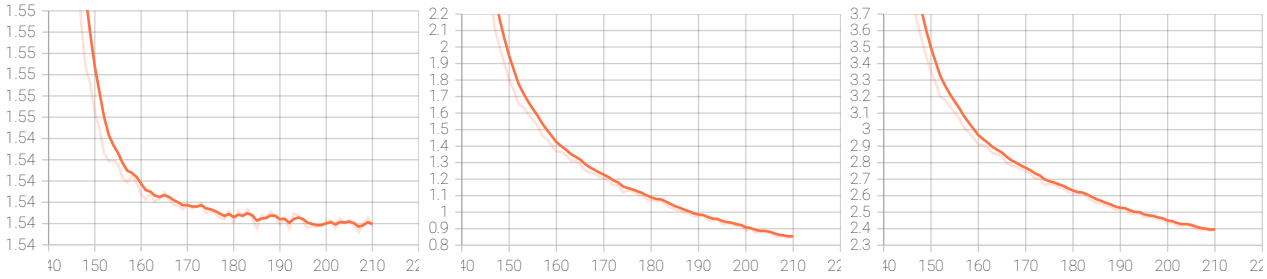
(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.2.** Treinamento com FQ igual a 3 (MMAL-Net).



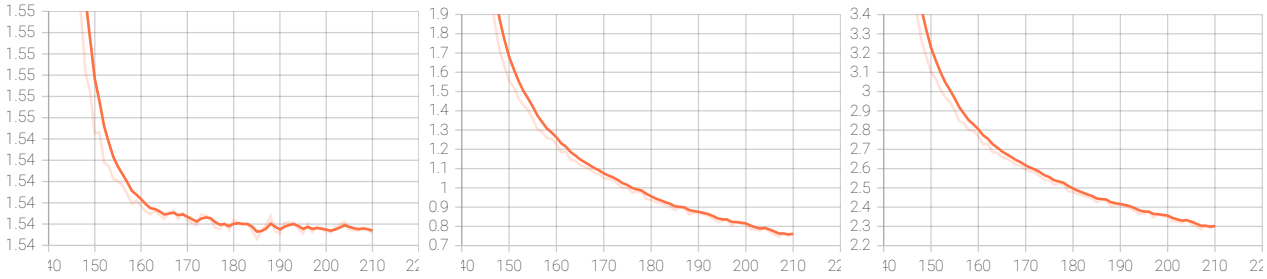
(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.3.** Treinamento com FQ igual a 4 (MMAL-Net).



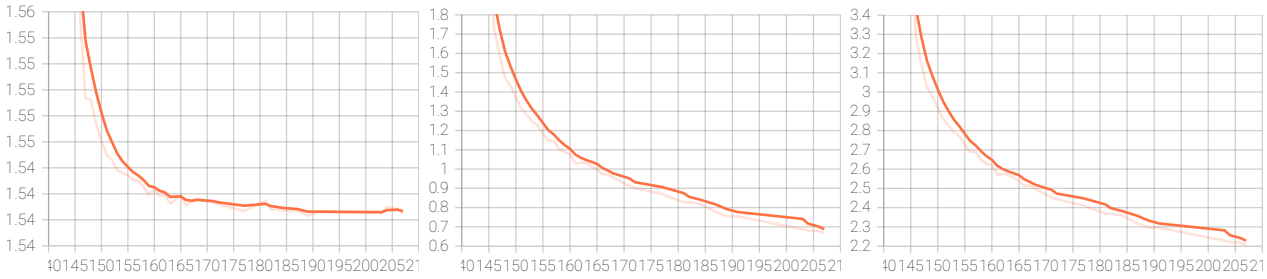
(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.4.** Treinamento com FQ igual a 6 (MMAL-Net).



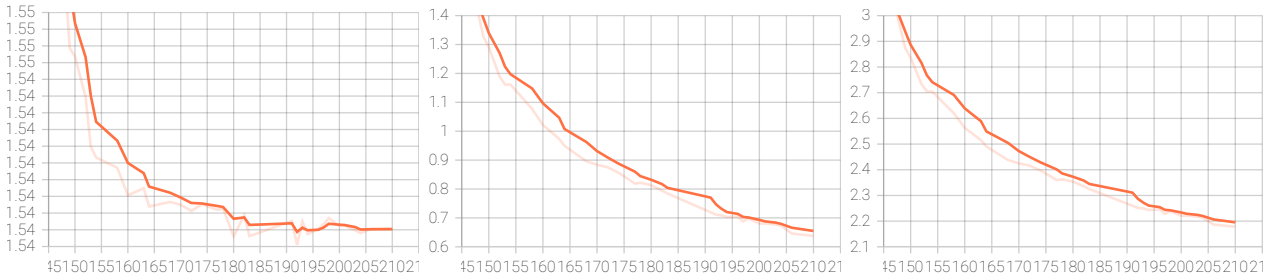
(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.5.** Treinamento com FQ igual a 7 (MMAL-Net).



(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.6.** Treinamento com FQ igual a 8 (MMAL-Net).

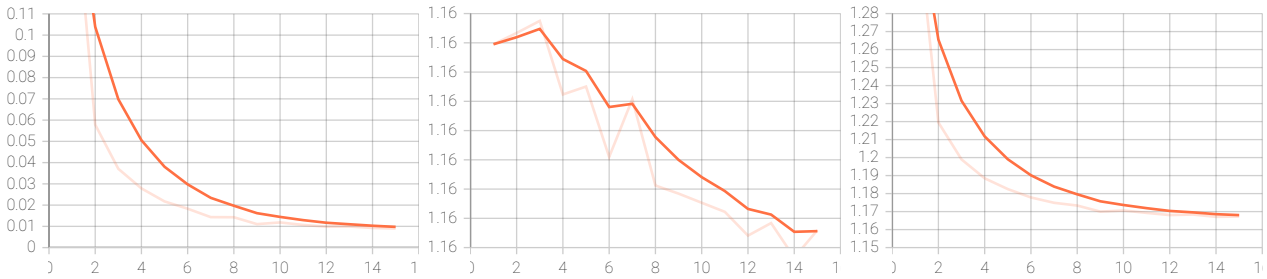


(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.7.** Treinamento com FQ igual a 9 (MMAL-Net).

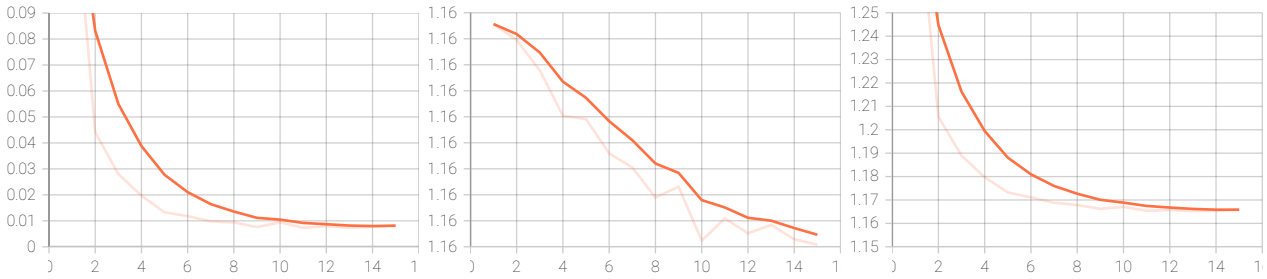
## A.2 CCT

Para a RN de dupla ramificação com o ramo de classificação sendo a CCT, as curvas de custos serão:



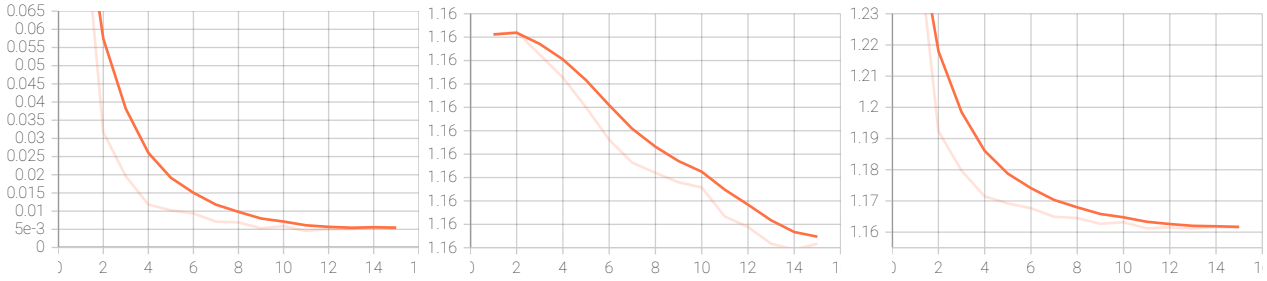
(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.8.** Treinamento com FQ igual a 2 (CCT).



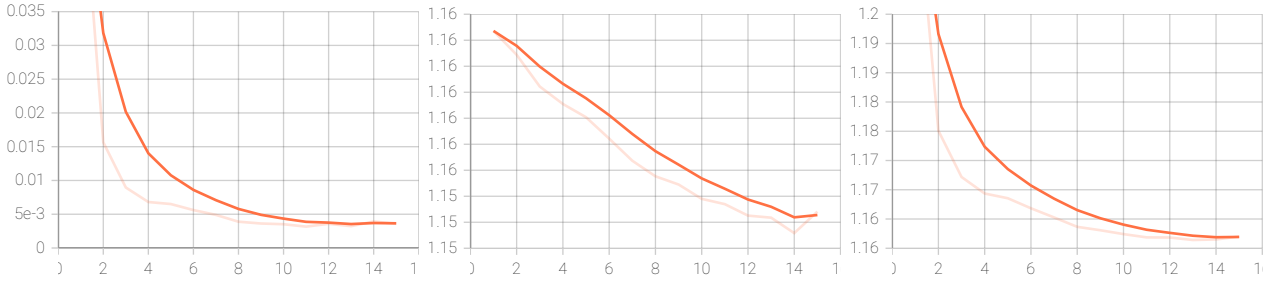
(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.9.** Treinamento com FQ igual a 3 (CCT).



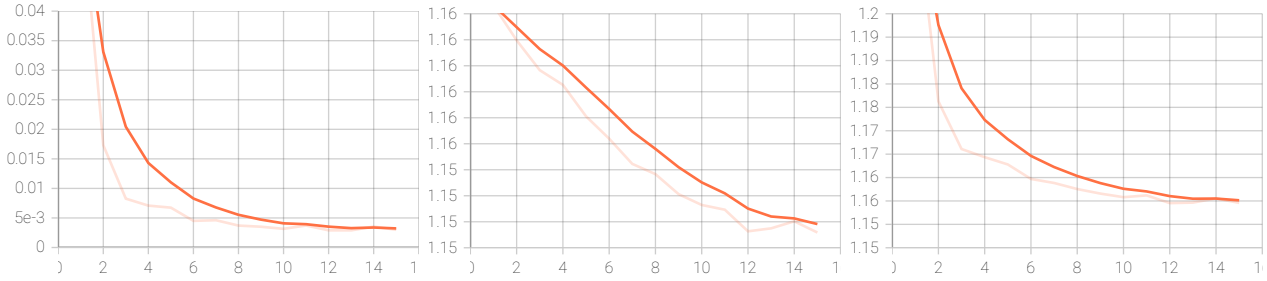
(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.10.** Treinamento com FQ igual a 4 (CCT).



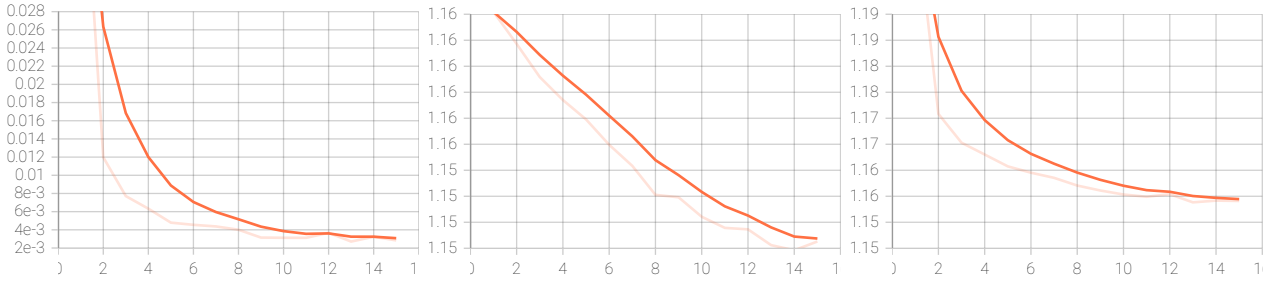
(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.11.** Treinamento com FQ igual a 6 (CCT).



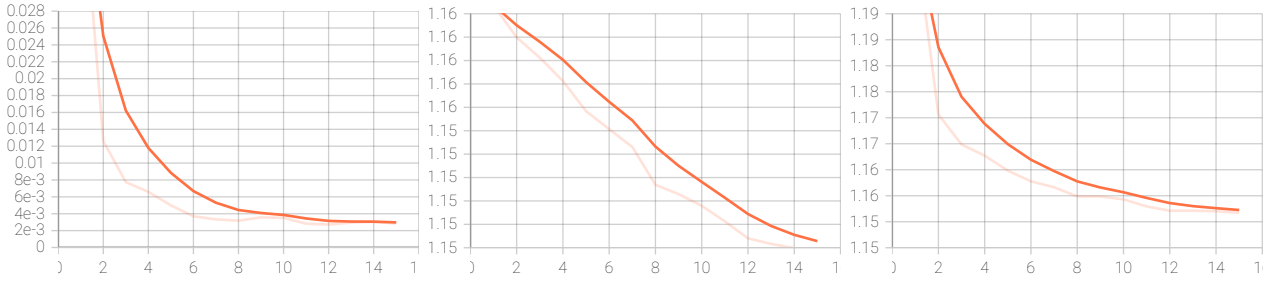
(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.12.** Treinamento com FQ igual a 7 (CCT).



(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.13.** Treinamento com FQ igual a 8 (CCT).




(a) Custo da RAC (b) Custo da Classificação (c) Custo total

**Figura A.14.** Treinamento com FQ igual a 9 (CCT).