



Programa de Pós-Graduação em Ciência da Computação

Lucas de Brito Silva

Segmentação semântica de imagens com *BPCAPooling*: uma abordagem baseada em aprendizado profundo

Rio Claro - SP

2024

UNIVERSIDADE ESTADUAL PAULISTA

“Júlio de Mesquita Filho”

Instituto de Geociências e Ciências Exatas

Câmpus de Rio Claro

LUCAS DE BRITO SILVA

**Segmentação Semântica de Imagens com *BPCAPooling*: Uma Abordagem Baseada em
Aprendizado Profundo**

Dissertação de mestrado apresentada ao Instituto de Geociências e Ciências Exatas do Campus de Rio Claro, da Universidade Estadual Paulista “Júlio de Mesquita Filho”, como parte dos requisitos para a obtenção do título de mestre no Programa de Pós-Graduação em Ciência da Computação.

Orientador: Prof. Dr. Denis Henrique Pinheiro Salvadeo

Rio Claro - SP

2024

S586s

Silva, Lucas de Brito

Segmentação semântica de imagens com BPCAPooling: uma abordagem baseada em aprendizado profundo / Lucas de Brito Silva. -- Rio Claro, 2024
194 p. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Geociências e Ciências Exatas, Rio Claro

Orientador: Denis Henrique Pinheiro Salvadeo

1. Visão computacional. 2. Segmentação de imagens. 3. Redes neurais convolucionais. 4. Preservação de espacialidade. 5. Block-based Principal Component Analysis Pooling (BPCAPooling). I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Geociências e Ciências Exatas, Rio Claro. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

“Júlio de Mesquita Filho”

Instituto de Geociências e Ciências Exatas

Câmpus de Rio Claro

LUCAS DE BRITO SILVA

**Segmentação Semântica de Imagens com *BPCAPooling*: Uma Abordagem Baseada em
Aprendizado Profundo**

Dissertação de mestrado apresentada ao Instituto de Geociências e Ciências Exatas do Campus de Rio Claro, da Universidade Estadual Paulista “Júlio de Mesquita Filho”, como parte dos requisitos para a obtenção do título de mestre no Programa de Pós-Graduação em Ciência da Computação.

Comissão Examinadora

Prof. Dr. DENIS HENRIQUE PINHEIRO SALVADEO

IGCE / UNESP / Rio Claro (SP)

Prof. Dr. ANDRE LUIS DEBIASO ROSSI

ICE / UNESP / Itapeva (SP)

Prof. Dr. ALEXANDRE LUÍS MAGALHÃES LEVADA

DC / UFSCar / São Carlos (SP)

Conceito: Aprovado.

Rio Claro (SP), 05 de Fevereiro de 2024

Agradecimentos

Gostaria de expressar minha profunda gratidão ao meu Senhor Jesus Cristo por me permitir chegar até aqui. Realizar o mestrado sempre foi um sonho, e alcançar essa conquista é, sem dúvida, resultado de Sua misericórdia e graça sobre minha vida. Esta conquista é dedicada à honra e glória d’Ele!

Dedico à Gabriela Ferreira de Brito, que no período de mestrado me apoiou como namorada, noiva e hoje é minha esposa. Muitas vezes as esperanças dela renovaram as minhas forças e esteve presente em todos os momentos. Aos meus pais, Luiz Carlos da Silva e Sônia de Brito Silva, também ao meu irmão, Gabriel de Brito Silva, por me incentivar incansavelmente e mostrar que sou capaz, incentivando-me dia após dia a prevalecer em direção aos meus sonhos. À minha filha Amélia de Brito, que me motiva a alcançar conquistas para nossa família e prover um futuro melhor para a minha posteridade.

Expresso minha sincera gratidão ao meu orientador, Denis, por confiar em mim, por me proporcionar o privilégio de aprender com ele e por investir seu tempo em minha vida, meu sonho e jornada acadêmica. Agradeço também aos amigos que fiz no grupo PRISMA (*Pattern Recognition, Image and Signal Processing, and Multimedia Applications*), com os quais compartilhei conhecimento e cresci. Gostaria de destacar Darlan Murilo Nakamura de Araújo, Uemerson Pinheiro Junior e Davi Duarte de Paula, que se tornaram amigos queridos, e o meu melhor amigo Álvaro Leandro Cavalcante Carneiro, que tem sido meu companheiro desde o ensino médio e me deu apoio nos momentos mais desafiadores. Conforme expresso por Isaac Newton, “Se enxerguei mais longe, foi por estar apoiado nos ombros de gigantes”.

Sou grato a todos os membros da comissão avaliadora por dedicarem seu tempo e por sua contribuição inestimável a este momento único em minha vida.

"O temor do SENHOR é o princípio da sabedoria; todos os que cumprem os seus preceitos
revelam bom senso. Ele será louvado para sempre!"

Salmo 111:10 - Nova Versão Internacional

Resumo

A visão computacional, especialmente no âmbito de segmentação de imagens, tem promovido contribuições significativas para análises médicas avançadas, compreensão de cenas complexas, design de sistemas autônomos, entre outros domínios de possíveis aplicações. A emergência de técnicas de aprendizado profundo tem possibilitado o avanço de arquiteturas e modelos com vistas a alcançar primeiramente o estado-da-arte na segmentação de imagens. No entanto, ainda há obstáculos significativos a serem superados, particularmente a preservação simultânea da espacialidade e a manutenção de uma visão global do mapa de atributos durante o procedimento de redução de dimensionalidade inerente às camadas de *pooling* das redes convolucionais. Frente a esse desafio, este trabalho propõe a investigação e o desenvolvimento do método Block-based Principal Component Analysis Pooling (BPCAPooling). Diferente do método convencional *Max Pooling*, o BPCAPooling é um método de *pooling* baseado no PCA que busca não apenas preservar localmente a informação espacial das amostras, mas também manter uma visão global dos mapas de atributos enquanto reduz a dimensionalidade. Para avaliar a eficácia deste método, foi aplicado primeiramente na arquitetura de redes neurais convolucionais como a Visual Geometry Group (VGG) com 16 camadas de peso, e posteriormente estendido a arquiteturas mais complexas com o desafio de segmentação de imagens, nomeadamente as U-Nets e suas variantes. Os experimentos mostraram que, embora a aplicação do BPCAPooling em modelos de classificação não tenha superado os métodos tradicionais em termos de métricas como acurácia e *loss*, observaram-se diferenças visuais na comparação com os métodos de *pooling* convencionais. Além disso, na tarefa de segmentação semântica, o método mostrou-se como uma alternativa viável, atingindo uma pontuação de 0,3333 em Mean Intersection over Union (mIoU), acurácia de 86,77% e *loss* de 0,6659.

Palavras-chave: Visão computacional; Segmentação de imagens; Redes neurais convolucionais; Preservação de espacialidade; Block-based Principal Component Analysis Pooling (BPCAPooling).

Abstract

Computer vision, especially in the field of image segmentation, has made significant contributions to advanced medical analysis, understanding complex scenes, design of autonomous systems, among other areas of possible applications. The emergence of deep learning techniques has enabled the advancement of architectures and models with a view to achieving the state-of-the-art in image segmentation. However, there are still significant obstacles to be overcome, particularly the simultaneous preservation of spatiality and maintaining a global view of the attribute map during the dimensionality reduction procedure inherent in the pooling layers of convolutional networks. Faced with this challenge, this work proposes the investigation and development of the Block-based Principal Component Analysis Pooling (BPCAPooling) method. Different from the conventional Max Pooling method, BPCAPooling is a PCA-based pooling method that seeks not only to locally preserve the spatial information of samples, but also to maintain a global view of attribute maps while reducing dimensionality. To evaluate the effectiveness of this method, it was first applied to the architecture of convolutional neural networks such as the Visual Geometry Group (VGG) with 16 weight layers, and later extended to more complex architectures with the challenge of image segmentation, namely U-Nets and its variants. The experiments showed that, although the application of BPCAPooling in classification models did not outperform traditional methods in terms of metrics such as accuracy and loss, visual differences were observed when compared with conventional pooling methods. Furthermore, in the semantic segmentation task, the method proved to be a viable alternative, reaching a score of 0.3333 in Mean Intersection over Union (mIoU), accuracy of 86.77% and loss of 0.6659.

Keywords: Computer vision; Image segmentation; Convolutional neural networks; Preservation of spatiality; Block-based Principal Component Analysis Pooling (BPCAPooling).

Lista de figuras

1.1	Segmentação médica.	17
1.2	Segmentação em sistemas autônomos.	18
1.3	Segmentação com <i>GrabCut</i>	19
1.4	Segmentação na odontologia.	20
2.1	Representação do neurônio biológico.	24
2.2	Representação gráfica de neurônio McCulloch e Pitts (1943).	25
2.3	Problemas de separabilidade.	26
2.4	Representação do modelo MLP.	27
2.5	Otimização da função de custo.	33
2.6	Parada antecipada.	36
2.7	Atualização da taxa de aprendizado.	37
2.8	Comparação de convergência de modelos no CIFAR.	39
2.9	Fenômenos de <i>underfitting</i> e <i>overfitting</i>	44
3.1	Modelo de CNN.	46
3.2	Representação do processo de convolução.	47
3.3	<i>Max pooling</i>	50
3.4	<i>Average pooling</i>	51
3.5	<i>Global Average Pooling</i>	52
3.6	<i>Global Max Pooling</i>	53
3.7	Exemplo ilustrativo do ASPP.	54
3.8	Processo de <i>dropout</i>	55
3.9	Exemplo de aumento de dados.	58
3.10	Inversão no conjunto MNIST.	59
3.11	Representação de VGGs.	60
4.1	Exemplos de segmentações de imagens.	62
4.2	Histogramas de limiarizações.	65
4.3	Limiarizações.	65

4.4	Ilustração do crescimento de região.	66
4.5	Segmentação com operador Sobel.	68
4.6	Segmentação com operador Laplaciano.	69
4.7	Segmentação com <i>watershed</i>	71
4.8	Processo de segmentação <i>watershed</i>	72
4.9	Segmentação <i>livewire</i>	74
4.10	Exemplo da segmentação com <i>superpixels</i>	75
4.11	Fluxograma do algoritmo K-means.	77
4.12	Segmentação com K-means.	77
5.1	Segmentação semântica.	81
5.2	Representação de exemplo em FCN.	83
5.3	Arquitetura U-Net.	85
5.4	Representação de IoU.	89
5.5	Representação do coeficiente Dice.	90
5.6	Segmentação com instâncias unificadas.	91
6.1	Exemplo de aplicação do BPCA.	98
6.2	Amostra com máscara de características.	102
6.3	Amostras do CIFAR-100.	104
6.4	Amostras do <i>Food-101</i>	105
6.5	Processo de <i>fine-tuning</i> da VGG-16.	107
6.6	Aumento de dados com transformações geométricas - <i>Food-101</i>	110
6.7	Aumento de dados com transformações geométricas - <i>Oxford-IIIT Pets</i>	112
6.8	Substituição das camadas de <i>pooling</i> por BPCAPooling - CIFAR-100.	113
6.9	Substituição das camadas de <i>pooling</i> por BPCAPooling - <i>Food-101</i>	113
6.10	Arquitetura U-Net-Like com aplicação de BPCAPooling.	115
6.11	Arquitetura U-Net com aplicação de BPCAPooling.	116
6.12	Exemplo de saída gerado pelo LIME em situação de erro.	125
6.13	Exemplo de saída gerado pelo LIME em situação correta.	126
6.14	Aplicação de métodos de atribuição do Xplique.	127
7.1	Comparação dos conjuntos de dados CIFAR-100 e <i>Food-101</i>	132
7.2	Evolução de Acurácia no conjunto de dados CIFAR 100.	133
7.3	Evolução de <i>Loss</i> no conjunto de dados CIFAR 100.	134

7.4	Evolução de Acurácia no conjunto de dados <i>Food-101</i>	135
7.5	Evolução da <i>Loss</i> no conjunto de dados <i>Food-101</i>	136
7.6	Camadas de <i>pooling</i> no CIFAR-100	137
7.7	Visualização de camadas de <i>Max Pooling</i> - <i>Food-101</i>	138
7.8	Visualização de camadas de BPCAPooling - <i>Food-101</i>	139
7.9	Visualização das últimas camadas de <i>pooling</i> no conjunto <i>Food-101</i>	141
7.10	LIME no CIFAR 100.	142
7.11	LIME no <i>Food-101</i>	143
7.12	Métricas com U-Net, 500 épocas, <i>Oxford-IIIT Pets</i> , acurácia.	145
7.13	Métricas com U-Net, 500 épocas, <i>Oxford-IIIT Pets</i> , mIoU	146
7.14	Métricas com U-Net, <i>Max Pooling</i> , 500 épocas, <i>Oxford-IIIT Pets</i> , mIoU	147
7.15	Aplicação de U-Net com acurácia.	148
7.16	Aplicação de U-Net com mIoU.	149
7.17	Segmentação com U-Net, <i>Max Pooling</i> , 500 épocas, <i>Oxford-IIIT Pets</i> , mIoU . .	150
7.18	Segmentação com U-Net, BPCAPooling, 500 épocas, <i>Oxford-IIIT Pets</i> , mIoU .	150
7.19	Segmentação com U-Net, <i>Max Pooling</i> , 500 épocas, <i>Oxford-IIIT Pets</i> , acurácia	150
7.20	Segmentação com U-Net, BPCAPooling, 500 épocas, <i>Oxford-IIIT Pets</i> , acurácia	151
7.21	Métricas com U-Net-Likes, 500 épocas, <i>Oxford-IIIT Pets</i> , acurácia.	152
7.22	Métricas de U-Net-Likes, BPCAPooling, 20 épocas, <i>Oxford-IIIT Pets</i> , mIoU. . .	153
7.23	Métricas com U-Net-Likes, <i>Max Pooling</i> , 20 épocas, <i>Oxford-IIIT Pets</i> , mIoU. .	154
7.24	Aplicação de U-Net-Like baseada em acurácia.	155
7.25	Aplicação de U-Net-Like baseada em mIoU.	156
7.26	Segmentação com U-Net-Like, <i>Max Pooling</i> , 500 épocas, <i>Oxford-IIIT Pets</i> , acurácia.	157
7.27	Segmentação com U-Net-Like, BPCAPooling, 500 épocas, <i>Oxford-IIIT Pets</i> , acurácia.	157
7.28	Segmentação com U-Net-Like, <i>Max Pooling</i> , 20 épocas, <i>Oxford-IIIT Pets</i> , mIoU.	157
7.29	Segmentação com U-Net-Like, BPCAPooling, 20 épocas, <i>Oxford-IIIT Pets</i> , mIoU.	158
7.30	Escores de métodos com U-Nets e acurácia.	159
7.31	Métodos destaque em U-Net com BPCAPooling e acurácia.	160
7.32	Métodos destaque em U-Net com <i>Max Pooling</i> e acurácia.	160
7.33	Escores de métodos com U-Nets e mIoU.	161

7.34	Métodos destaque em U-Net com BPCAPooling e mIoU.	162
7.35	Métodos destaque em U-Net com <i>Max Pooling</i> e mIoU.	162
7.36	Escores de métodos com U-Net- <i>Likes</i> e acurácia.	163
7.37	Métodos destaque em U-Net- <i>Likes</i> com BPCAPooling e acurácia.	164
7.38	Métodos destaque em U-Net- <i>Likes</i> com <i>Max Pooling</i> e acurácia.	164
7.39	Escores de métodos com U-Net- <i>Likes</i> e mIoU	166
7.40	Métodos destaque em U-Net- <i>Likes</i> com BPCAPooling e mIoU.	167
7.41	Métodos destaque em U-Net- <i>Like</i> com <i>Max Pooling</i> e mIoU.	167
7.42	Aplicação de BPCAPooling e BPCAUnpooling.	168

Lista de tabelas

1	Comparação entre os métodos de segmentações tradicionais.	79
2	Transformações geométricas aplicadas no conjunto de dados CIFAR 100. . . .	109
3	Transformações geométricas aplicadas no conjunto de dados <i>Food-101</i>	110
4	Transformações executadas no conjunto de dados <i>Oxford-IIIT Pets</i>	111
5	Treinamento e Teste VGG-16 + BPCAPooling (<i>Food-101</i>)	120
6	Treinamento e Teste VGG-16 + <i>Max Pooling</i> (<i>Food-101</i>)	120
7	Treinamento e Teste VGG-16 + <i>Avg Pooling</i> (<i>Food-101</i>)	120
8	Treinamento e Teste VGG-16 + BPCAPooling (CIFAR 100)	121
9	Treinamento e Teste VGG-16 + <i>Max Pooling</i> (CIFAR 100)	122
10	Treinamento e Teste VGG-16 + <i>Avg Pooling</i> (CIFAR 100)	122
11	<i>Fine-tuning</i> de VGG-16 (CIFAR 100)	130
12	<i>Fine-tuning</i> de VGG-16 (<i>Food-101</i>)	130
13	Desempenho de acurácia das U-Nets (<i>Oxford-IIIT Pets</i>)	144
14	Desempenho de mIoU das U-Nets (<i>Oxford-IIIT Pets</i>)	144
15	Desempenho de acurácia das U-Net-Likes (<i>Oxford-IIIT Pets</i>)	151
16	Desempenho de mIoU das U-Net-Likes (<i>Oxford-IIIT Pets</i>)	151

Sumário

1	INTRODUÇÃO	17
1.1	Objetivos	22
1.2	Considerações Finais do Capítulo	23
2	O BÁSICO DE REDES NEURAIS ARTIFICIAIS	24
2.1	Funções de Ativação	28
2.1.1	Função Limiar	28
2.1.2	Função Linear	29
2.1.3	Função Sigmoides	30
2.1.4	Função ReLU	30
2.1.5	Função Softmax	30
2.2	Treinamento	31
2.2.1	Função de Custo	32
2.2.2	Função de Otimização	32
2.2.2.1	Gradiente Descendente	34
2.2.2.2	Gradiente Descendente Estocástico	34
2.2.2.3	Parada antecipada	35
2.2.2.4	Taxa de Aprendizado Dinâmica	36
2.2.2.5	Adam	39
2.2.3	Retropropagação	40
2.3	Teste	41
2.3.1	Métricas de Avaliação	41
2.3.2	Validação Cruzada	42
2.3.3	<i>Overfitting</i> e <i>Underfitting</i>	43
2.4	Considerações Finais do Capítulo	44
3	REDES NEURAIS CONVOLUCIONAIS	45
3.1	Camada Convolutiva	46

3.2	Camada de <i>Pooling</i>	48
3.2.1	<i>Max Pooling</i>	49
3.2.2	<i>Average Pooling</i>	50
3.2.3	<i>Global Average Pooling</i>	52
3.2.4	<i>Global Max Pooling</i>	53
3.2.5	<i>Atrous Spatial Pyramid Pooling(ASPP)</i>	53
3.3	<i>Dropout</i>	55
3.4	Camada de Saída	55
3.5	Transferência de Aprendizado	56
3.6	Aumento de Dados	57
3.7	VGG	59
3.8	Considerações Finais do Capítulo	61
4	SEGMENTAÇÃO DE IMAGENS	62
4.1	Métodos e Técnicas Tradicionais	63
4.1.1	Segmentação Baseada em Regiões	63
4.1.2	Segmentação por Bordas	67
4.1.3	Segmentação Baseada em Grafo	69
4.1.4	Segmentação <i>Watershed</i>	70
4.1.5	Segmentação <i>Livewire</i>	73
4.1.6	Segmentação <i>Superpixels</i>	74
4.1.7	Segmentação Baseada em Agrupamentos	76
4.1.8	Segmentação com Redes Neurais	78
4.2	Considerações Finais do Capítulo	78
5	SEGMENTAÇÃO SEMÂNTICA BASEADA EM APRENDIZADO PROFUNDO	81
5.1	Arquiteturas	82
5.1.1	<i>Fully Convolutional Networks (FCN)</i>	82
5.1.2	U-Net	84
5.1.3	U-Net-Like	86
5.2	Métricas	87
5.2.1	<i>Pixel Accuracy</i>	88
5.2.2	<i>Intersection over Union (IoU)</i>	88

5.2.3	<i>Mean Intersection Over Union (MIoU)</i>	89
5.2.4	<i>F1-Score</i>	89
5.3	Considerações Finais do Capítulo	91
6	DESENVOLVIMENTO DO PROJETO	93
6.1	Materiais e Métodos	93
6.1.1	Tecnologias e <i>Hardware</i>	94
6.1.2	Extração de Características	95
6.1.3	<i>Block-based Principal Component Analysis Pooling (BPCAPooling)</i>	98
6.1.4	Conjuntos de Dados	101
6.1.4.1	<i>Oxford-IIIT Pets</i>	101
6.1.4.2	CIFAR 100	103
6.1.4.3	<i>Food-101</i>	104
6.1.5	Transferência de Aprendizado	106
6.1.6	Aumento de Dados	108
6.1.6.1	CIFAR 100	108
6.1.6.2	<i>Food-101</i>	109
6.1.6.3	<i>Oxford-IIIT Pets</i>	110
6.1.7	Alteração na Camada de <i>Pooling</i>	112
6.2	Metodologia Experimental	117
6.2.1	Escolha das Métricas	117
6.2.2	Parâmetros de Treinamento e Teste	118
6.2.2.1	VGG-16 + <i>Food-101</i>	119
6.2.2.2	VGG-16 + CIFAR 100	121
6.2.2.3	U-Net-Like + <i>Oxford-IIIT Pets</i>	122
6.2.2.4	U-Net + <i>Oxford-IIIT Pets</i>	124
6.2.3	Explicabilidade de Modelos	124
6.3	Considerações Finais do Capítulo	127
7	RESULTADOS E DISCUSSÕES	129
7.1	Resultados da Classificação de Imagens	129
7.1.1	Diferença de conjuntos de dados	131
7.1.2	LIME e Preservação de Espacialidade	136

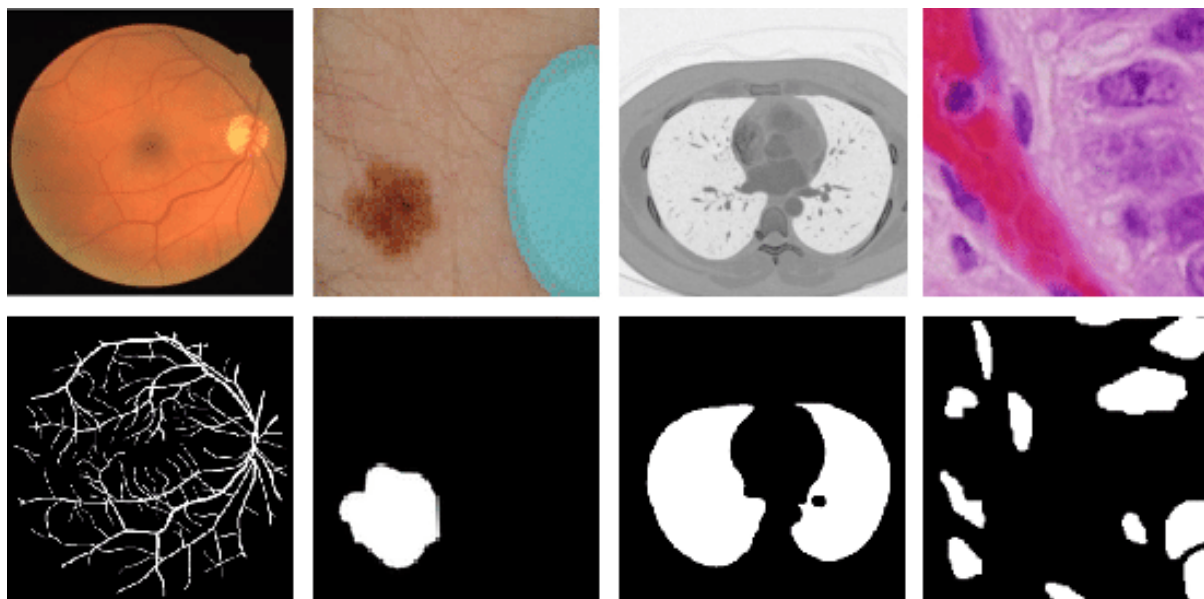
7.2	Resultados da Segmentação Semântica	143
7.3	Considerações Finais do Capítulo	169
8	CONCLUSÕES	170
8.1	Trabalhos Futuros	171
8.2	Considerações Finais do Capítulo	173
	REFERÊNCIAS BIBLIOGRÁFICAS	174

1 INTRODUÇÃO

A segmentação de imagens assume um papel fundamental no campo de visão computacional e análise de cenas, desempenhando uma função instrumental no auxílio de atividades humanas complexas. Embora a segmentação em si não desencadeie uma ação, a sua aplicação está profundamente enraizada em diversas áreas que dependem de análise visual.

Em particular, na medicina, a segmentação de imagens tem sido uma ferramenta valiosa, possibilitando avaliações analíticas mais precisas e detalhadas (Lai, 2015; Withey e Koles, 2008). É visível, por exemplo, na radiologia, onde a segmentação é aplicada para delimitar com precisão tumores (Malkanathi e Dissanayake, 2017), proporcionando informações vitais para a programação de intervenções cirúrgicas e tratamentos oncológicos. O mesmo se aplica à quantificação de volumes teciduais e ao mapeamento preciso de órgãos específicos a partir de exames de imagem (Gibson et al., 2018; Schoppe et al., 2020). Estas aplicações culminam em uma melhor compreensão da anatomia do paciente e possíveis patologias, levando a decisões de tratamento mais bem fundamentadas e precisas. A Figura 1.1 apresenta situações em que a segmentação de imagens desempenha um papel imprescindível no cenário médico.

Figura 1.1: Exemplos de aplicação da segmentação no contexto médico. Na parte superior, estão as imagens originais, enquanto na parte inferior estão suas respectivas segmentações, representando segmentação de vasos sanguíneos, câncer de pele, câncer pulmonar e núcleos celulares, respectivamente.



Fonte: Scheuer et al. (2020).

A segmentação de imagens e vídeos é igualmente crucial no desenvolvimento e implementação de sistemas autônomos (Kaymak e Ucar, 2019; Liu et al., 2020; Pan et al., 2020; Teichmann et al., 2018). Nestes sistemas, que abrangem diversas áreas - da indústria ao transporte - a segmentação é frequentemente aplicada a uma variedade de tarefas essenciais.

No setor industrial, por exemplo, máquinas autônomas frequentemente dependem da segmentação para realizar tarefas de controle de qualidade. Isso envolve a análise metódica de imagens de produtos em linhas de montagem para identificar, por exemplo, possíveis defeitos de fabricação.

De modo similar, em sistemas de transporte autônomo, como carros completamente automatizados, a segmentação de imagens é vital para garantir a segurança e a eficácia operacional. Tais sistemas utilizam a segmentação para distinguir e identificar precisamente vários elementos em um ambiente de condução - como pedestres, placas de trânsito e sinais de trânsito, conforme desenvolvido por Lee e Kim (2018) e Fleyeh (2004). Esta análise permite que o sistema reaja de maneira adequada a uma variedade de situações de condução e potenciais obstáculos (Lee e Kim, 2018; Fleyeh, 2004; Pan et al., 2020). A Figura 1.2 apresenta exemplos de tais aplicações, evidenciando o papel crucial que a segmentação de imagens e vídeos desempenha em sistemas autônomos.

Figura 1.2: Exemplos de segmentação em sistemas autônomos.



Fonte: Lee e Kim (2018).

Embora a segmentação seja um passo essencial no processo de análise de imagens, como mencionado anteriormente, ela não gera diretamente uma ação. Em vez disso, a segmentação muitas vezes faz parte de um protocolo intermediário em fluxos de trabalho que envolvem reconhecimento de imagens ou detecção de objetos.

Um exemplo distinto do papel primordial da segmentação pode ser observado no estudo conduzido por [Carneiro et al. \(2021b\)](#), onde é implementado o método *GrabCut* ([Rother et al., 2004](#)) para a segmentação de imagens. *GrabCut* é um método de segmentação baseado em grafos, uma abordagem bastante frequente no domínio da segmentação, como explorado por [Yi e Moon \(2012\)](#).

No caso do estudo de [Carneiro et al. \(2021b\)](#), o processo de segmentação é uma etapa prévia à crucial fase de detecção de doenças e pragas em folhas de café. Inicialmente, a segmentação por meio do *GrabCut* é empregada para isolar as folhas de café na imagem. Seguidamente, os resultados desta segmentação servem como base para a detecção mais precisa de possíveis enfermidades ou infestações nessas folhas. A Figura 1.3 ilustra essa aplicação.

Figura 1.3: Segmentação feita com *GrabCut*.



Fonte: [Carneiro et al. \(2021b\)](#).

Este estudo no domínio particular da segmentação de imagens exemplifica como tal técnica é instrumental na catalisação de processos complexos em sistemas de visão computacional. Permitindo o isolamento preciso das áreas de interesse, a segmentação promove a extração de informações mais detalhadas e relevantes a partir de imagens. Desta forma, impacta de forma significativa os resultados finais, consubstanciando-se como uma ferramenta insubstituível em

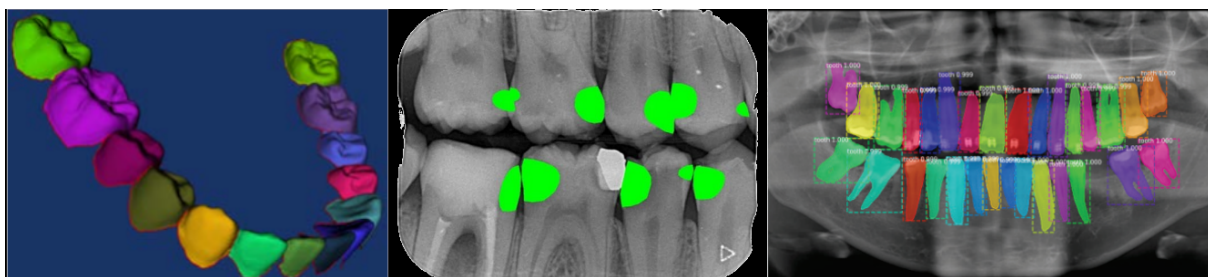
uma vasta gama de aplicações práticas e estudos acadêmicos.

A segmentação de imagens revela também a sua relevância em áreas de aplicação específicas como a odontologia, que oferece várias oportunidades e desafios únicos (Zanjani et al., 2021; Chung et al., 2020). Neste domínio, há um esforço contínuo para identificar com precisão todos os componentes da cavidade bucal do paciente. Isso engloba desde o diagnóstico de patologias, como as cáries, até a avaliação do estado geral de saúde dental.

O diagnóstico na odontologia depende em grande medida da análise de imagens clínicas, e uma interpretação acurada é crucial para determinar o curso apropriado do tratamento, incluindo a necessidade de dispositivos ortodônticos (Schwendicke et al., 2020). Neste sentido, como realçado por Priya et al. (2021); Nguyen et al. (2021) e Schwendicke et al. (2020), as abordagens baseadas em inteligência artificial estão se mostrando soluções particularmente promissoras. Ao melhorar a precisão e a velocidade do diagnóstico, tais abordagens aprimoram o desempenho humano e consequentemente otimizam os resultados de tratamento.

Uma ilustração vívida disso é a aplicação da segmentação de imagens ao reconhecimento de dentes e cáries. Por meio da utilização de algoritmos de segmentação avançados, é possível isolar e identificar exatamente essas estruturas dentárias e patologias nas imagens, facilitando a subsequente análise e diagnóstico. A Figura 1.4 apresenta alguns exemplos de aplicações da segmentação de imagens na odontologia, demonstrando o potencial desta abordagem para aprimorar a prática clínica.

Figura 1.4: Exemplo de segmentação na área odontológica. Na imagem, são mostradas segmentações de instâncias dentárias em moldes, cáries em radiografia *bitewing* e instâncias em raio-x panorâmico, respectivamente.



Fonte: retirado e adaptado de Yang et al. (2016); Bayrakdar et al. (2021); Jader et al. (2019), respectivamente.

Dentre as autênticas abordagens de segmentação, é significativo notar que muitos algoritmos foram desenvolvidos para responder à demanda contínua por técnicas eficazes de segmen-

tação. Estes incluem métodos manuais sofisticados - discutidos em detalhe no Capítulo 4 - baseados em região ou limiar (Seção 4.1.1), bordas (Seção 4.1.2), grafos (Seção 4.1.3), bem como algoritmos como *Watershed* (Seção 4.1.4), *Livewire* (Seção 4.1.5) e Superpixel (Seção 4.1.6). Também estão presentes abordagens de agrupamento (Seção 4.1.7). Além disso, dada a proliferação das redes neurais, surgiram também métodos mais complexos baseados nesta técnica (Seção 4.1.8). Começa-se, então, uma incursão nos intrincados mares da temática da segmentação de imagens, com uma primeira escala nos detalhes dessas diversas técnicas de segmentação nos capítulos seguintes.

Impulsionados por melhorias contínuas em capacidades de *hardware* e disponibilidade de dados, os avanços na área de aprendizado profundo têm gerado progressos notáveis, particularmente no domínio da segmentação de imagens. Esses avanços têm sido proporcionados não apenas por aprimoramentos em infraestrutura e dados, mas também pelo desenvolvimento de novos algoritmos e abordagens mais eficazes para resolver problemas complexos (Szegedy et al., 2015).

Enfatiza-se que os modelos de ponta que empregam aprendizado profundo demonstram notável capacidade de realizar combinações matriciais que destacam características específicas e podem até realizar segmentação precisa, atribuindo categorias a todos os *pixels* de uma imagem (Minaee et al., 2021a). Isto é evidente nas redes neurais convolucionais (Capítulo 3) e nas técnicas de segmentação semântica (Capítulo 5), respectivamente.

Com as técnicas contemporâneas de segmentação, é possível lidar com dados massivos contidos nas imagens, explorando aspectos intrincados, como a escala dos objetos na cena e a dimensionalidade preservada pelos modelos. Contudo, mesmo com os avanços das técnicas de segmentação, a identificação de objetos pequenos na imagem continua a ser um desafio significativo (Sang et al., 2023; Su et al., 2021). Além disso, tanto nos modelos que se utilizam de técnicas para reduzir a dimensionalidade das imagens quanto em abordagens mais recentes focadas na segmentação detalhada de imagens, um tema que se encontra pendente é em relação a avaliação crítica da adequação dos métodos comumente usados para a redução de dimensionalidade em relação à preservação espacial e consideração de visão global dos atributos.

Portanto, dentre os temas tratados, o presente projeto apresenta alguns algoritmos e *frameworks* reconhecidos para segmentação, mas visa, principalmente, estudar o impacto da alteração das camadas de *pooling* convencionais pelo uso do BPCAPooling em arquiteturas totalmente supervisionadas marcantes, como as U-Nets, avaliando suas vantagens, limitações e potenciais

áreas de investigação futura que poderiam informar novos experimentos e contribuir para o progresso científico no campo da segmentação de imagens, especialmente aqueles que empregam aprendizado profundo.

1.1 Objetivos

O objetivo geral deste trabalho consiste em propor um novo método de *pooling* que, além de reduzir a dimensionalidade, preserva as informações espaciais e consideração de visão global dos atributos iniciais, devido à sua capacidade de reduzir a dimensionalidade dos dados enquanto mantém as características essenciais que contribuem para a variabilidade dos dados, visando aprimorar o detalhamento das segmentações. Tal proposta se diferencia dos métodos convencionais de *pooling*, os quais não atendem a esse requisito (Liu et al., 2019). Para validar essa abordagem, foram conduzidos experimentos considerando diferentes arquiteturas, desde redes convolucionais convencionais até modelos mais avançados para a segmentação semântica, buscando observar o impacto em diferentes contextos. As principais questões de pesquisa deste trabalho são:

1. É possível conceber um método de *pooling* que reduza a dimensionalidade, tenha uma visão global dos atributos e preserve a espacialidade?
2. O método de *pooling* proposto é capaz de ser utilizado como alternativa dos métodos de *pooling* convencionais?
3. Existem diferenças significativas na aplicação do método proposto entre redes convolucionais convencionais e modelos de segmentação semântica?

A primeira questão foi explorada por meio do desenvolvimento de uma proposta de *pooling* denominada *Block-based Principal Component Analysis* (BPCAPooling), a qual será detalhada na Seção 6.1.3, visando abordar os desafios mencionados nessa questão específica.

O segundo item de pesquisa busca testar o método proposto em comparação com os métodos convencionalmente utilizados para a tarefa de *pooling*. Esta etapa está diretamente relacionada ao terceiro ponto mencionado, o qual visa realizar esses testes em diferentes arquiteturas, desde aquelas mais antigas que introduzem conceitos de *pooling* até arquiteturas mais modernas dedicadas à segmentação detalhada de imagens. Dessa forma, as contribuições desta pesquisa são:

1. O desenvolvimento de um novo método de *pooling*;
2. A avaliação da aplicabilidade do novo método de *pooling* substituindo os métodos convencionais em redes convolucionais tradicionais;
3. Visualização comparativa das saídas das camadas de *pooling* nas redes convolucionais tradicionais;
4. A análise da eficácia do novo método de *pooling* em substituir os métodos convencionais em modelos de segmentação semântica.

1.2 Considerações Finais do Capítulo

Nos próximos capítulos, serão abordados temas relacionados à compreensão de conceitos fundamentais sobre redes neurais profundas e redes neurais convolucionais no Capítulo 2. Em seguida, o Capítulo 3 trará informações mais direcionadas para as redes neurais convolucionais, fundamentais para o processamento e segmentação de imagens.

O Capítulo 4 discorrerá sobre as técnicas tradicionais empregadas em atividades de segmentação. Este será seguido por uma exploração das segmentações viabilizadas pelo surgimento das redes convolucionais e de aprendizado profundo, especialmente em relação à segmentação semântica, no Capítulo 5.

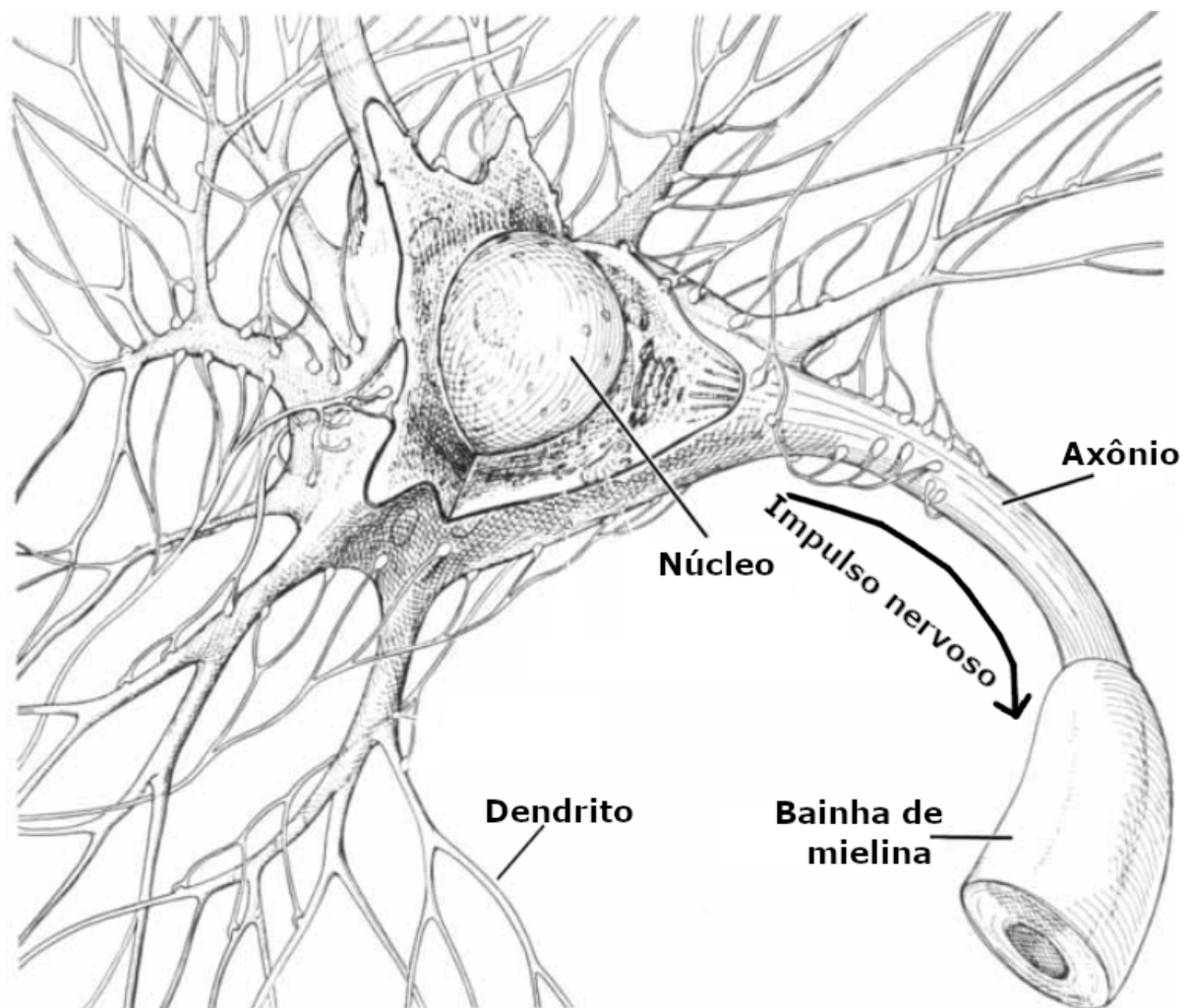
Posteriormente, o Capítulo 6 fornecerá detalhes claros sobre a metodologia proposta neste trabalho, descrevendo a estrutura do método desenvolvido e seus componentes fundamentais.

Por fim, o Capítulo 7 será dedicado à apresentação dos resultados obtidos e discussões pertinentes, enquanto o Capítulo 8 conterà as considerações finais deste estudo.

2 O BÁSICO DE REDES NEURAIS ARTIFICIAIS

Inspiradas no modelo de sistema nervoso de seres inteligentes, as redes neurais são compostas por um conjunto interconectado de neurônios de modo que, assim como o sistema humano, um neurônio, quando conectado com outros neurônios, pode comunicar impulsos nervosos por meio de sinapses. A representação do modelo humano pode ser observado na Figura 2.1.

Figura 2.1: Representação do neurônio biológico.



Fonte: traduzido e adaptado de [Stevens \(1979\)](#).

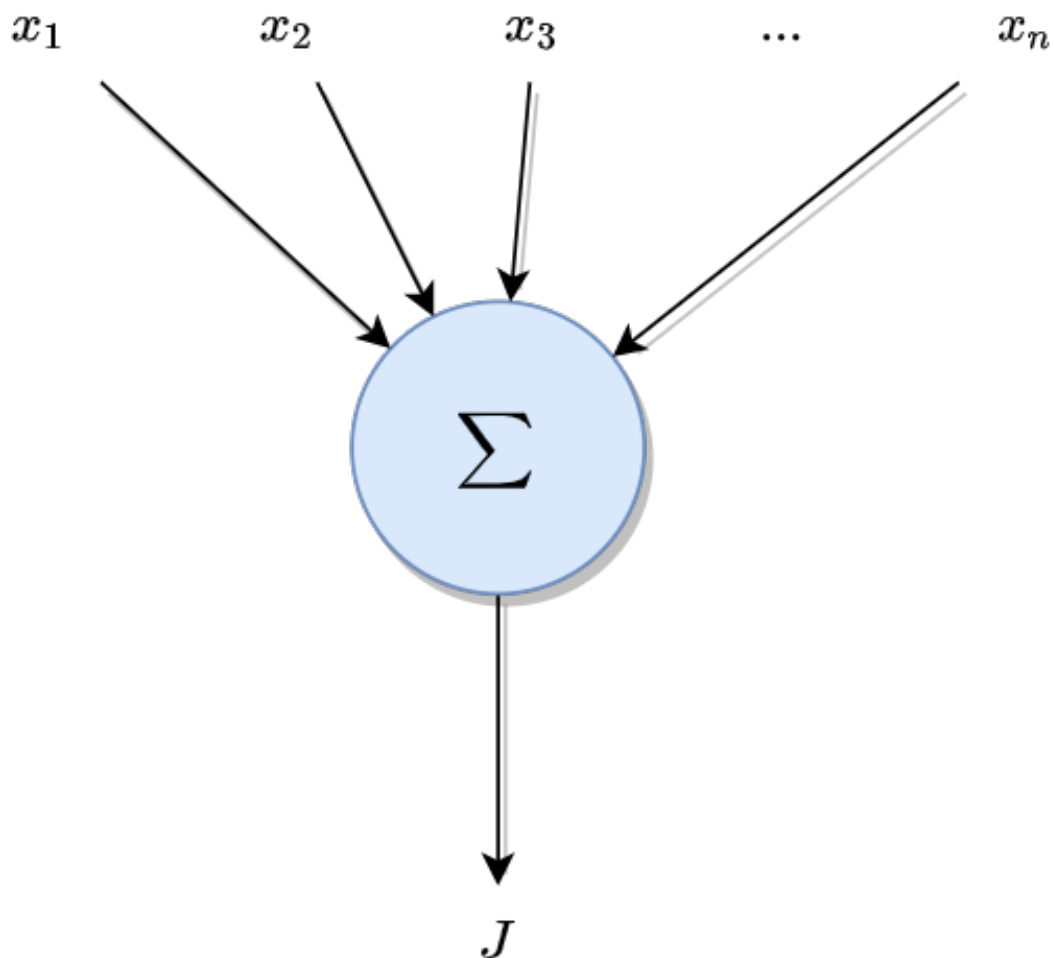
Dentre as representações matemáticas, destaca-se o modelo pioneiro desenvolvido por [McCulloch e Pitts \(1943\)](#), que baseado no neurônio biológico busca definir uma saída binária, a partir de entradas que são recebidas por meio dos dendritos, sendo assim, sua representação

matemática se dá pela equação:

$$J = \sum_{i=1}^n x_i. \quad (2.1)$$

Para a Equação 2.1 e para a Figura 2.2, é importante dizer que J representa a saída do neurônio, enquanto x_i é a entrada a partir de i . Logo, é possível observar que por meio dos dendritos ocorre a entrada dos dados que é somada (ponderadamente) no núcleo $\sum_{i=1}^n x_i$. Por fim, vale citar que $(x_1, \dots, x_n)$ é assimilado aos dendritos e, além disso, é comum ver a representação do modelo supracitado através da (Figura 2.2).

Figura 2.2: Representação gráfica de neurônio McCulloch e Pitts (1943).



Fonte: do próprio autor. Inspirado em McCulloch e Pitts (1943).

A partir do modelo de neurônio desenvolvido por McCulloch e Pitts (1943), outros trabalhos foram desenvolvidos, no qual vale ressaltar o trabalho de Rosenblatt (1958), que propôs

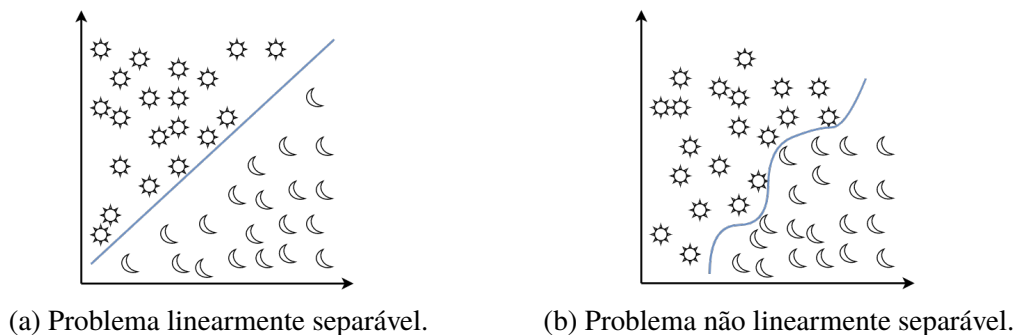
um modelo de neurônio e o nomeou de Perceptron, o qual tem como princípio o uso de uma saída binária. Logo, destaca-se que a sua função de ativação é dada de forma condicional e pode ser representada pela Equação 2.2 (Rosenblatt, 1958):

$$A(Y') = \begin{cases} 1, & \text{se } \sum_{i=1}^N w_i x_i + b \geq 0 \\ 0, & \text{caso contrario.} \end{cases} \quad (2.2)$$

Dessa forma, entende-se que b representa o viés que determina um limiar para a ativação do neurônio. O vetor $\mathbf{x}$ corresponde à entrada da rede, enquanto $\mathbf{W}$ é a matriz de pesos que deve ser inicializada aleatoriamente e é ajustada no decorrer do treinamento.

Dentre as vantagens para o modelo Perceptron, pode-se salientar a sua aptidão para resolução de alguns problemas lógicos, basicamente problemas linearmente separáveis, todavia, em meio às suas limitações, reforça-se sua imperícia em relação a problemas não linearmente separáveis como os do exemplo na Figura 2.3b.

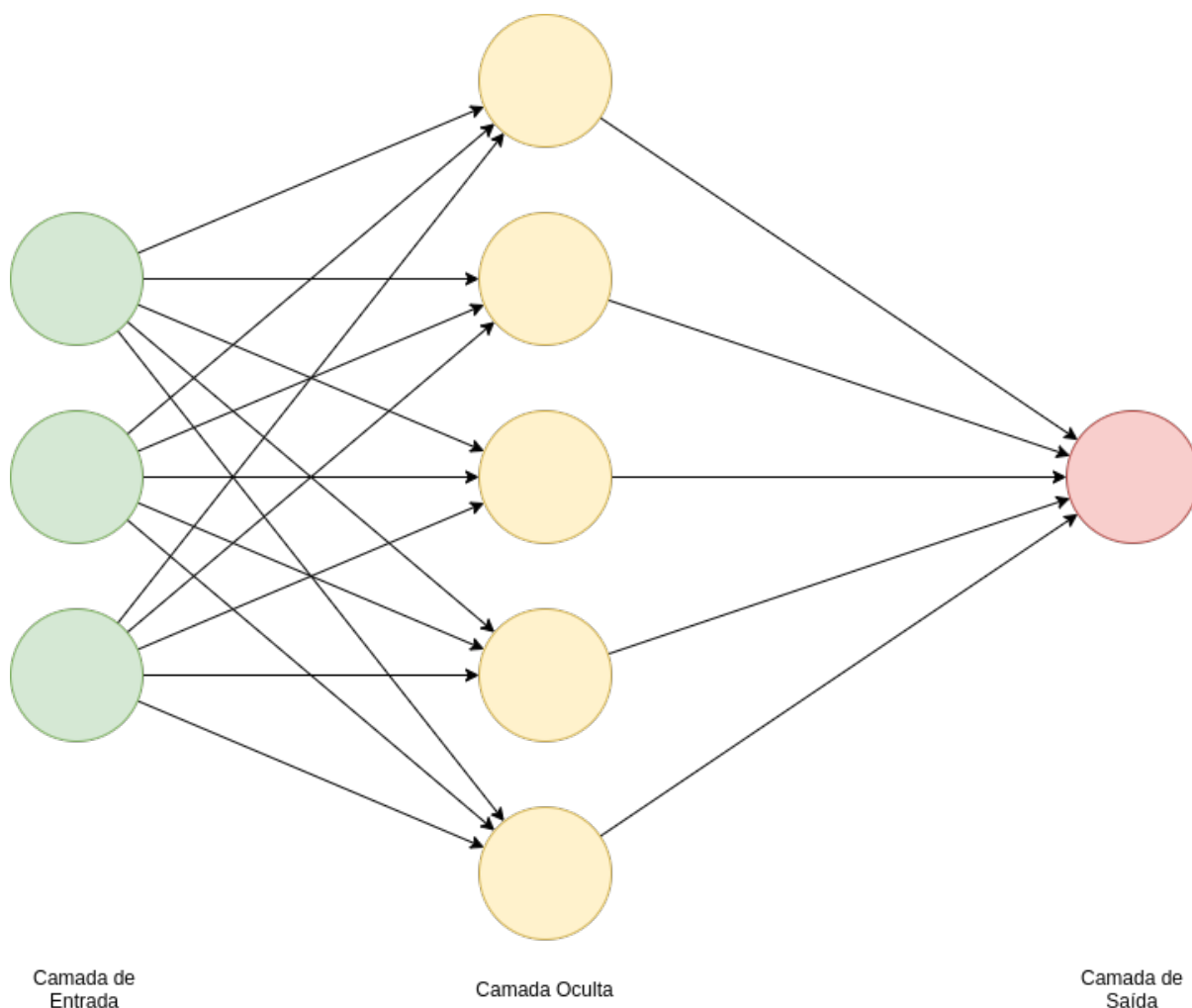
Figura 2.3: Representação de problemas linearmente e não linearmente separáveis.



Fonte: do próprio autor.

Para suprir essa necessidade de trabalhar com problemas não lineares, Werbos (1974) realizou uma proposta de utilizar várias camadas de neurônios Perceptron juntos, de modo que obrigatoriamente possuísse camadas de entrada, ocultas e de saída. Assim, esse modelo - nomeado de *Multi Layer Perceptron* (MLP) (Werbos, 1974) e representado na Figura 2.4 - ensejou o trabalho com funções de ativação não lineares e o desenvolvimento de técnicas de aprendizado, como a *backpropagation*, desenvolvida por Rumelhart et al. (1986), a qual será detalhada na Seção 2.2.3.

Figura 2.4: Representação do modelo MLP.



Fonte: do próprio autor.

Entretanto, com o uso dos modelos citados, na tentativa de resolver problemas de nível de complexidade maior e do desenvolvimento de trabalhos que contribuíssem para o estado-da-arte, percebeu-se que ao adicionar mais camadas era possível trabalhar com uma base de dados maior e mais complexa, além de obter melhores resultados e uma etapa maior de treinamento, o que deu origem às intituladas redes neurais de aprendizado profundo, ou em inglês, *deep learning* (Goodfellow et al., 2016). No que lhe concerne, essas possuem o princípio de aprender os parâmetros e representações dos dados de acordo com as suas interações (Ponti e Da Costa, 2018). Assim, permitindo que o algoritmo combine e simplifique os parâmetros antes de propagar as representações para as demais camadas (Goodfellow et al., 2016).

Além disso, vale citar que um diferencial dos modelos de *deep learning* se materializa no fato de que seus neurônios se tornam especialistas em determinados objetivos e se tornam

diferentes um dos outros, como o que ocorre com as CNNs, que têm neurônios responsáveis por extrair características, enquanto os neurônios da camada final fazem classificação (ou regressão), simulando o tráfego da informação no córtex visual humano.

Destarte, o uso e desenvolvimento de *deep learning* tem se popularizado e sido considerado a resposta para problemas de diversas áreas, como segmentações em vários contextos, desenvolvimentos de carros autônomos, segurança, otimização de modelos de busca e afins (Ghosh et al., 2019), sendo que essa popularização é propícia devido: 1) ao desenvolvimento acelerado de *hardwares* que acrescentam velocidade no treinamento dessas redes e 2) ao grande aumento de dados variados (com adventos como o *big data*) (Szegedy et al., 2015; Ponti e Da Costa, 2018).

Por conseguinte, nas demais seções deste capítulo serão apresentadas algumas variações de funções de ativação (Seção 2.1), sobre o processo de treinamento (Seção 2.2) e de testes (Seção 2.3).

2.1 Funções de Ativação

Quanto às funções de ativação, destaca-se que na seção anterior foi apresentada uma função de ativação binária, todavia, existem outras funções de ativação que foram estudadas e são utilizadas para uma melhor adequação em contextos específicos.

A função de ativação, como citado brevemente, é calculada pelo neurônio e disseminada para os próximos neurônios. Para essa Seção, o valor calculado $\sum_{i=1}^N w_i x_i + b$ (entrada da função de ativação) será representado por Y' , de modo que poderemos ver sua atuação, a seguir, nas funções de ativação Limiar, Linear, Sigmóide, ReLU e Softmax.

2.1.1 Função Limiar

A função limiar define um limite (*threshold*) γ , em que valores abaixo de γ indicam uma saída sem excitação, para valores acima de γ , há a excitação para o neurônio, utilizando assim, de medidas binárias, que normalmente são representadas por 0 e 1, respectivamente.

Embora a função limiar seja eficaz ao lidar com problemas binários, sua aplicação torna-se desafiadora em cenários multiclasse. Isso se deve ao fato de que ela não assegura a ativação exclusiva de um único neurônio na camada de saída, dificultando a representação adequada de classes distintas.

Em relação à função de ativação ($f(Y')$) desenvolvida e utilizada no Perceptron, pode-se

dizer que o valor de $\gamma = 0$. A função limiar é definida matematicamente por McCulloch e Pitts (1943), segundo a Equação 2.3:

$$f(Y') = \begin{cases} 1, & \text{se } Y' \geq \gamma \\ 0, & \text{caso contrário} \end{cases}. \quad (2.3)$$

2.1.2 Função Linear

Para suprir a problemática de trabalhar com contextos de multiclasse, saídas contínuas podem ajudar na interpretação e mitigar o problema da função de limiar. Nesse contexto, é apropriado o uso da função linear. É importante notar que os resultados produzidos por funções lineares são também lineares, tornando-as inadequadas para problemas não lineares. A função linear é expressa pela Equação 2.4, onde α representa um parâmetro ajustável, conforme definido por Rosenblatt (1958):

$$f(Y') = \alpha Y'. \quad (2.4)$$

O *Adaptive Linear Neuron* (ADALINE), proposto por Widrow e Hoff (Widrow e Lehr, 1990), é um marco histórico na evolução dos algoritmos de aprendizado de máquina. Ele representa uma significativa progressão em relação ao Perceptron, pois emprega uma função linear para calcular sua saída a partir das entradas. Enquanto o Perceptron atualiza seus pesos com base em erros de classificação binários, o ADALINE ajusta os pesos por meio de uma função de erro contínuo, utilizando a diferença entre a saída esperada e a saída calculada pela função linear. Essa abordagem mais refinada permite uma correção mais precisa dos pesos, demonstrando uma evolução significativa na capacidade de aprendizado dos modelos.

A regra de aprendizado de Widrow-Hoff, foi desenvolvida especificamente para o ADALINE. Ela se concentra em minimizar a soma dos quadrados dos erros entre as saídas desejadas e as saídas produzidas pela rede, um método que demonstra ser eficaz para ajustar os parâmetros (α) do modelo de forma a reduzir esse erro. Este princípio é aplicado em algoritmos de aprendizado profundo, enfatizando a relevância histórica e prática do ADALINE e da regra de Widrow-Hoff na fundação das redes neurais modernas e no aprendizado de máquina como um todo.

Embora o ADALINE seja restrito a problemas lineares, seu desenvolvimento foi um

passo crucial para os modelos subsequentes, que generalizam a abordagem para contextos não lineares, aprimorando as capacidades de modelagem e de aprendizado das redes.

2.1.3 Função Sigmoid

As funções sigmoide têm uma ampla aplicação além de problemas lineares, caracterizando-se como uma função de ativação não linear. Elas permitem a modelagem de probabilidade em relação à saída do neurônio.

Esta função de ativação também é conhecida como função logística. Sua expressão matemática é definida por (Glorot et al., 2011):

$$f(Y') = \frac{1}{1 + e^{Y'}}. \quad (2.5)$$

A desvantagem dessa função encontra-se em situações de variação do eixo x, em que há uma dissipação do seu gradiente, implicando em um aprendizado mais lento nos neurônios das primeiras camadas em relação aos das últimas, tendo em vista os valores muito pequenos no início (Goodfellow et al., 2016).

2.1.4 Função ReLU

A função desenvolvida por Hahnloser et al. (2000) e popularmente conhecida como ReLU (*Rectified Linear Unit*) se faz útil nas camadas ocultas da rede neural (Goodfellow et al., 2016), sendo que sua utilidade é destacada por não permitir resultados negativos (Dahl et al., 2013) e ter uma rápida convergência, principalmente quando comparada à função sigmoide, além de não passar por problemas de perda do gradiente. Sua função é definida pela Equação 2.6 (Hahnloser et al., 2000):

$$f(Y') = \max(0, Y'). \quad (2.6)$$

2.1.5 Função Softmax

Por fim, quando se trata da função Softmax, dois processos são essenciais para a convergência do modelo em questão, sendo ele: 1) realizar a exponenciação dos valores de cada neurônio e 2) realizar a normalização de cada valor pela soma de todos os valores exponenciado no passo anterior, assim, definido uma saída no domínio $[0, 1]$, fazendo com que a soma total

seja sempre 1 (Kotu e Deshpande, 2018) e que a saída da função seja a distribuição de probabilidade da classe. Esse comportamento garante que a função Softmax seja recomendada para problemas de multiclases, além de determinar as probabilidades de cada uma das classes.

Em um problema de k-classes, a função Softmax pode ser expressa pela Equação 2.7 (Kotu e Deshpande, 2018):

$$f(Y = \mathbf{k}) = \frac{e^{z_k}}{\sum_{i=1}^k e^{Y_i}}, \quad (2.7)$$

tendo z_k como cada entrada do vetor $\mathbf{k}$.

2.2 Treinamento

Definida como uma fase crucial por Ponti e Da Costa (2018), a fase de treinamento está relacionada com a minimização da função de custo (que será abordada na Seção 2.2.1), de modo que a rede ajuste os seus parâmetros iterativamente. Essas interações são explicadas por cada laço de repetição que ocorre com cada exemplo, sendo esperado que o erro propagado na rede seja minimizado. Assim, quando as interações ocorrem por todo o conjunto de dados de treinamento, é dito que aconteceu a ocorrência de uma época.

Normalmente, os passos realizados em uma época são os seguintes: 1) inicialização; 2) escolha de uma amostra (normalmente de modo aleatório); 3) cálculo de saída através de função de ativação; 4) cálculo do erro através de função de custo e *backpropagation*; 5) modificação dos pesos de cada camada da rede e; 6) uma nova interação, até que tenha se passado por todas as instâncias do conjunto de treinamento.

Além da seleção aleatória, vale citar que normalmente os pesos são iniciados de modo aleatório nas camadas, assim como os hiperparâmetros, que são definidos pelo usuário na etapa de inicialização, por exemplo, o valor da taxa de aprendizado.

Nesta seção, será brevemente percorrido sobre as funções de custo (Seção 2.2.1), que indicam o erro da rede em relação ao conjunto de treinamento, sobre as funções de otimização (Seção 2.2.2), que auxiliam na convergência do modelo e, por fim, sobre o algoritmo *backpropagation* (Seção 2.2.3), que propaga o erro para as camadas internas da rede.

2.2.1 Função de Custo

As funções de custo (do inglês, *loss functions*) são responsáveis por realizar a avaliação do modelo durante a fase de treinamento, de certa maneira, avaliando quão perto o aprendizado do modelo está da resposta correta.

Em meio às avaliações utilizadas como métrica para o cálculo da função de custo em relação a problemas como o de regressão, destaca-se o erro quadrático médio (ou *Mean Squared Error*, MSE)(Wang et al., 2004), em que para o âmbito de visão computacional e trabalho com imagens, comumente ocorre da saída da rede neural ser uma imagem, a partir de onde calcula-se o MSE dos *pixels* existentes entre a imagem predita $\mathbf{y}'$ e a imagem original $\mathbf{y}$. Assim, supondo que M e N sejam as dimensões da imagem, o MSE é calculado pela equação:

$$MSE(\mathbf{y}, \mathbf{y}') = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (\mathbf{y}_{i,j} - \mathbf{y}'_{i,j})^2. \quad (2.8)$$

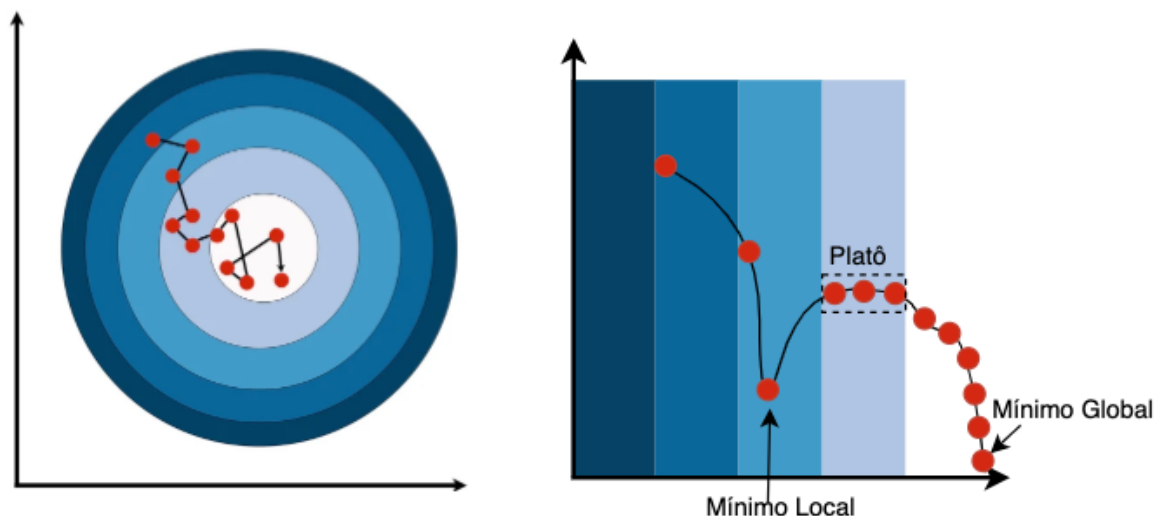
Ainda quanto às métricas para a realização de avaliação em relação a imagens, existem várias delas descritas na literatura, todavia, cada uma delas possui seu uso especificado para o problema que deseja resolver, por exemplo o caso das métricas para segmentações semânticas (Seção 5.2), que também poderia ser empregadas como função de custo para o treinamento.

2.2.2 Função de Otimização

O grande objetivo das funções de otimização no contexto de aprendizado profundo, de certa forma, podem ser resumidos pela Equação 2.9 e visualizada na Figura 2.5, em que almeja-se uma minimização global da função de custo $f(\mathbf{x})$ através de pesos $\boldsymbol{\theta}$ e de entradas $\mathbf{x}$:

$$\boldsymbol{\theta}_* = \operatorname{argmin}(f(\mathbf{x}; \boldsymbol{\theta})). \quad (2.9)$$

Figura 2.5: Representação da otimização da função de custo em um espaço de características.



Fonte: do próprio autor. Inspirado em Montesinos López et al. (2022) e Géron (2017).

A Figura 2.5 proporciona uma visualização de alguns desafios para as funções de otimização, sendo 1) mínimos locais e 2) platô (*plateau*, em inglês). Um mínimo local representa um ponto que é o menor em comparação com outros pontos em sua vizinhança próxima. Entretanto, é importante notar que um mínimo local nem sempre corresponde ao menor valor em todo o espaço de características. Dessa forma, a função de custo pode apresentar vários mínimos locais, o que frequentemente dificulta a busca pelo mínimo global desejado na otimização (Goodfellow et al., 2016). Já os pontos de platô indicam áreas onde o valor da função de custo permanece estagnado. Nestes locais, os gradientes são muito semelhantes, o que torna desafiadora a otimização do modelo (Goodfellow et al., 2016).

Em meio às diversas funções disponíveis, considera-se que o método mais utilizado é o de gradiente descendente estocástico, ou em inglês, *stochastic gradient descent* (SGD), desenvolvido inicialmente por Cauchy (1847), assim como suas variações (Goodfellow et al., 2016), sendo que o gradiente descendente estocástico é considerado como uma adaptação do gradiente descendente e que causa a aceleração do treinamento na maioria das vezes por selecionar aleatoriamente apenas uma parcela do conjunto de treinamento. Todavia, segundo Goodfellow et al. (2016), há casos em que o SGD não tem bom desempenho para convergir e por conta disso são estudadas suas variações.

De qualquer modo é interessante ter consciência que até então não há um consenso na

escolha do melhor método de otimização para problemas em gerais, segundo aborda Goodfellow et al. (2016).

2.2.2.1 Gradiente Descendente

O método do gradiente descendente representa uma técnica de otimização com a capacidade de encontrar soluções ideais em uma variedade de contextos, operando de maneira iterativa (Géron, 2017). Seu processo inicia-se com o cálculo dos gradientes locais resultados da função de custo $f(\theta)$, bem como com a atualização do valor dos pesos θ com base na magnitude do gradiente e na direção indicada pelo cálculo das derivadas. A derivada da função $f'(\theta)$, determina a inclinação da função $f(\theta)$ no ponto θ , permitindo a análise de como a variação nos valores de entrada x da função se reflete em mudanças correspondentes nos valores de saída (Goodfellow et al., 2016), sendo possível minimizar a função movendo os valores de θ em pequenos incrementos no sentido oposto ao da derivada.

Em situações envolvendo funções com múltiplas variáveis de entrada, as derivadas parciais $\frac{\partial}{\partial \theta_i} f(\theta)$ são levadas em consideração para mensurar o quanto a função f varia quando a entrada θ_i é alterada no ponto, θ . Assim, o gradiente passa a ser representado por um vetor que encapsula os valores de todas as derivadas parciais (Goodfellow et al., 2016), sendo expresso por: $\nabla_{\theta} f(\theta)$. Por fim, a atualização dos parâmetros ocorre conforme a Equação 2.10 descrita por Goodfellow et al. (2016):

$$\theta' = \theta - \epsilon \nabla_{\theta} f(\theta). \quad (2.10)$$

Dessa forma, o valor de entrada θ é ajustado por meio da multiplicação das derivadas parciais pela taxa de aprendizado ϵ , conforme mostrado na Figura 2.5, onde os círculos vermelhos indicam a evolução da taxa de aprendizado.

2.2.2.2 Gradiente Descendente Estocástico

O SGD é amplamente adotado devido à sua eficiência em modelos de aprendizado profundo e conjuntos de dados em larga escala (Goodfellow et al., 2016). No entanto, é importante mencionar suas limitações. O SGD não assegura a convergência e é sensível aos valores iniciais do modelo. No entanto, essas desvantagens não excluem sua utilidade, especialmente quando combinado com técnicas como normalização e o uso de *momentum* (Goodfellow et al., 2016).

A fórmula do SGD emprega a derivada da função $f(\mathbf{x}; \boldsymbol{\theta})$ em relação aos parâmetros $\boldsymbol{\theta}$ para otimizar a saída $\mathbf{y}' = f(\mathbf{x})$. Esta atualização dos parâmetros, impulsionada pela derivada, visa melhorar a saída prevista $\mathbf{y}'$. A representação matemática do SGD é demonstrada na Equação 2.11, conforme proposto por Goodfellow et al. (2016):

$$\mathbf{g} = \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)}) \right), \quad (2.11)$$

onde $L(x, y)$ é a função de custo, $\mathbf{x}^{(i)}$ representa a entrada, $\mathbf{y}^{(i)}$ indica a saída esperada, m é o número de exemplos no conjunto de dados, e $f(\mathbf{x}^{(i)}; \boldsymbol{\theta})$ é a saída da rede para o i -ésimo exemplo $\mathbf{x}^{(i)}$ com o conjunto de pesos $\boldsymbol{\theta}$.

2.2.2.3 Parada antecipada

Ao configurar os hiperparâmetros de um modelo, é crucial considerar a quantidade de épocas pelas quais o conjunto de dados deve passar durante o treinamento. Essa escolha tem um impacto significativo nos fenômenos discutidos na Seção 2.3.3. Optar por um número insuficiente de épocas frequentemente resulta em *underfitting*, enquanto um número excessivo pode levar ao *overfitting*. Surge, então, a pergunta fundamental: “Qual é o número ideal de épocas para cada problema?”

Para abordar essa questão, diversos estudos têm sido realizados com o intuito de encontrar critérios de parada no treinamento que minimizem ou previnam esses problemas (Prechelt, 1998a). Além disso, pesquisas recentes tentam antecipar situações de *overfitting* e *underfitting*, interrompendo o treinamento antes que afete o desempenho do modelo (Vilares Ferro et al., 2023). Existem múltiplos métodos disponíveis para essa tarefa, sendo a parada antecipada (do inglês, *early stopping*) um dos mais populares, podendo ser considerado uma forma de regularização, tal como o decaimento de pesos (Vilares Ferro et al., 2023).

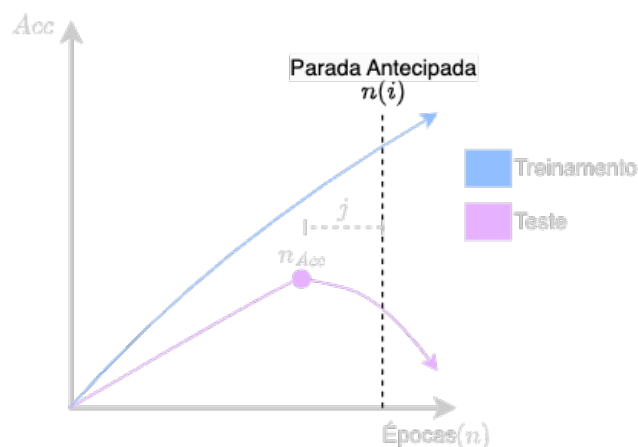
A aplicação da Parada Antecipada implica calcular a precisão da classificação nos dados de validação ao final de cada época (Prechelt, 1998a). Quando a precisão para de melhorar, encerramos o treinamento. Isso simplifica a definição do número de épocas, pois não é necessário explicitamente determinar como ele depende de outros hiperparâmetros - essa adaptação é feita automaticamente. Ademais, a Parada Antecipada atua como um mecanismo automático de prevenção ao *overfitting*. Embora, em estágios iniciais de experimentação, possa ser útil desativar temporariamente a Parada Antecipada para identificar sinais de *overfitting* e, assim, ajustar a

abordagem de regularização.

A implementação efetiva da parada antecipada exige uma definição mais precisa do que significa “parar de melhorar” em termos de precisão da classificação. Como observado anteriormente, a precisão pode oscilar mesmo durante uma tendência geral de melhora. Parar imediatamente quando há uma diminuição na precisão pode interromper o processo quando melhorias adicionais ainda são possíveis (Prechelt, 1998b). Uma regra mais sensata para a parada antecipada $n(i)$ no treinamento é se a melhor precisão de classificação não melhorar por um determinado número de épocas consecutivas j a partir da época de melhor acurácia n_{Acc} , como representado pela Equação 2.12 e também pela Figura 2.6:

$$n(i) = n_{Acc} + j. \quad (2.12)$$

Figura 2.6: Representação do momento ideal para parada antecipada.



Fonte: do próprio autor.

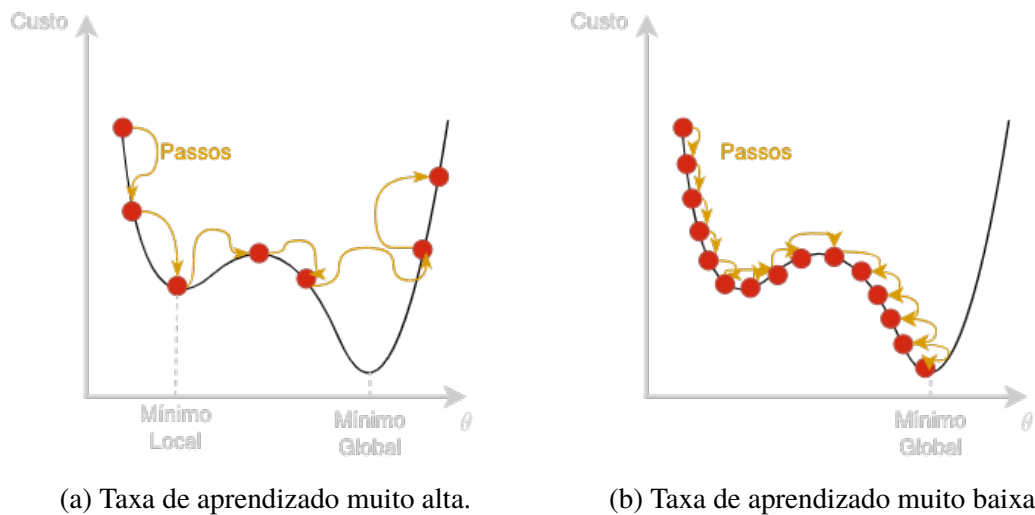
2.2.2.4 Taxa de Aprendizado Dinâmica

A taxa de aprendizado (*learning rate*, em inglês), como demonstrado anteriormente, é um dos hiperparâmetros fundamentais utilizados na etapa inicial do treinamento de modelos. Sua representação como ϵ pode ser observada na Equação 2.10 e desempenha um papel crucial na determinação do tamanho dos passos das iterações para a otimização de gradientes (Smith, 2017).

Definir a taxa de aprendizado, assim como abordado na situação de parada antecipada mencionada na Seção 2.2.2.3, muitas vezes torna-se um desafio sem que tenha um veredito de-

finido como estado da arte (Goodfellow et al., 2016). Isso se deve ao fato de que quanto maior o valor definido, mais acentuada é a atualização dos parâmetros, possibilitando uma exploração mais rápida do espaço de características, como ilustrado na Figura 2.7a. No entanto, essa abordagem frequentemente resulta na perda do valor ótimo (Tang et al., 2021). Por outro lado, ao diminuir o valor da taxa de aprendizado, a atualização dos parâmetros torna-se mais gradual, o que ajuda a evitar a perda de mínimos globais, como apresentado na Figura 2.7b. Contudo, é importante salientar que essa estratégia consome consideravelmente mais recursos computacionais e, muitas vezes, não atinge o valor mínimo global devido ao grande número de iterações necessárias para a convergência (Tang et al., 2021).

Figura 2.7: Representação de atualização da taxa de aprendizado.



Fonte: do próprio autor.

Essas representações visuais na Figura 2.7 evidenciam os impactos da variação da taxa de aprendizado na otimização do modelo. A escolha adequada desse hiperparâmetro é crucial para encontrar um equilíbrio entre a velocidade de convergência e a busca pelo valor ótimo da função de custo.

O aumento no número de estudos, como os de Tang et al. (2021); Senior et al. (2013) e Smith (2017), que visam aprimorar técnicas para encontrar valores ótimos de taxas de aprendizado tem se tornado uma tendência crescente. Esses estudos se concentram na busca por equilibrar os parâmetros ou otimizar a definição desses valores. Muitos desses estudos adotam uma abordagem baseada na programação de taxas de aprendizado dinâmicas, através do uso de um agendador de taxa de aprendizado (*learning rate scheduler*) - um método que define critérios para aumentar ou diminuir o valor da taxa de aprendizado ϵ . Normalmente, esse valor é

programado para diminuir linearmente, resultando em atualizações de pesos maiores no início da otimização e menores à medida que a otimização avança e o ponto mínimo se aproxima. A redução gradual da taxa de aprendizado pode ser descrita pela Equação 2.13, proposta por Goodfellow et al. (2016):

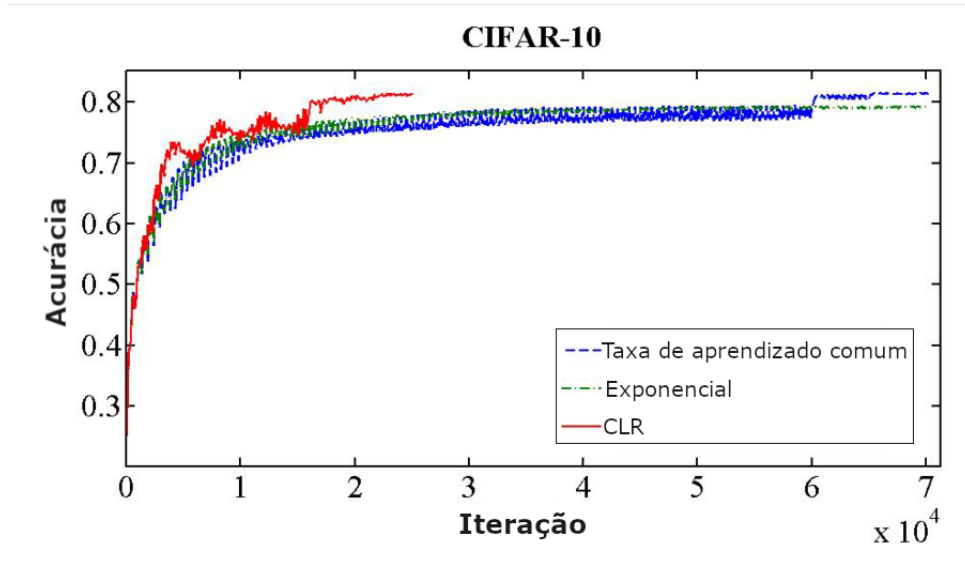
$$\epsilon_i = (1 - \alpha)\epsilon_0 + \alpha\epsilon_\tau, \quad (2.13)$$

sendo que nesta equação, τ representa o número de iterações em que ϵ será reduzido para uma constante, i é a iteração atual, ϵ_0 é a taxa de aprendizado inicial e α é determinado por $\frac{i}{\tau}$. Além disso, é importante destacar que o valor de τ é definido como o número de iterações necessárias para completar as épocas do conjunto de treinamento, enquanto ϵ_τ é aproximadamente 1% do valor de ϵ_0 , conforme indicado por Goodfellow et al. (2016).

É relevante mencionar que o valor de ϵ_0 pode variar conforme o problema em questão. Em problemas que envolvem um grande número de imagens, como frequentemente ocorre em redes neurais profundas citadas no Capítulo 2, valores típicos de ϵ_0 variam entre $1e^{-5}$ e $1e^{-6}$. Esse ajuste é necessário para garantir a eficiência do treinamento e a convergência do modelo.

Por fim, vale mencionar que propostas de decaimento da taxa de aprendizado, como o *Cyclical Learning Rates* (CLR), desenvolvido por Smith (2017), buscam reduzir o tempo de treinamento e manter a estabilidade dos modelos com a aplicação de variações geométricas. Por exemplo, o modelo triangular busca iniciar o treinamento com baixos valores de taxa de aprendizado e aumentar gradualmente até certo limite, proporcionando uma aceleração de convergência e acompanhado de um processo de diminuição para não perder os mínimos globais em um processo cíclico. Essa proposta sugere uma aceleração da convergência do modelo (Smith, 2017), o que pode ser observado através da Figura 2.8.

Figura 2.8: Comparação de convergência de modelos no *dataset* CIFAR (Krizhevsky et al., 2014) com uso de taxa de aprendizado cíclica.



Fonte: retirado e adaptado de Smith (2017).

2.2.2.5 Adam

O algoritmo *Adaptive Moment Estimation*, conhecido como Adam, foi proposto por Kingma e Ba (2014) com o objetivo de otimizar adaptativamente a taxa de aprendizado, integrando conceitos dos algoritmos RMSProp (Hinton et al., 2012) e Momentum (Polyak, 1964). A atualização dos parâmetros ocorre por meio do controle da média móvel exponencialmente ponderada de cada algoritmo mencionado, utilizando os parâmetros ρ_1 e ρ_2 em cada iteração para calcular a estimativa de primeira ordem s , conforme a Equação 2.14 (Goodfellow et al., 2016):

$$s \leftarrow \rho_1 s + (1 - \rho_1)g, \quad (2.14)$$

tendo g como gradientes da iteração. E a estimativa de segunda ordem r do gradiente, como expresso na Equação 2.15 (Goodfellow et al., 2016):

$$r \leftarrow \rho_2 r + (1 - \rho_2)g \odot g. \quad (2.15)$$

Além disso, é importante mencionar que é recomendado aplicar correção de viés às estimativas obtidas, o que pode ser realizado pelas Equações 2.16 e 2.17, conforme proposto por Goodfellow et al. (2016):

$$\hat{s} \leftarrow \frac{s}{1 - \rho_1} \quad (2.16)$$

e

$$\hat{r} \leftarrow \frac{r}{1 - \rho_2}. \quad (2.17)$$

Dessa forma, ao empregar o algoritmo Adam na otimização dos parâmetros, podemos representar a atualização dos parâmetros conforme demonstrado na Equação 2.18:

$$\theta \leftarrow \theta + \left(-\epsilon \frac{\hat{s}}{\sqrt{\hat{r}} + \delta} \right), \quad (2.18)$$

onde δ denota uma constante para estabilizar a divisão.

2.2.3 Retropropagação

Por meio do desenvolvimento de [Rumelhart et al. \(1986\)](#), o algoritmo de retropropagação (ou em inglês, *Backpropagation*) foi capaz de contribuir diretamente no desenvolvimento dos modelos de redes neurais profundas, de modo que esse algoritmo conseguiu realizar o reajuste de pesos em camadas ocultas dos modelos. Com o auxílio de uma função de custo (Seção 2.2.1), o erro é calculado na saída da rede (y) e há a propagação do mesmo, iniciando da camada de saída em direção para a camada de entrada. Isso é realizado através do trabalho conjunto do algoritmo *backpropagation* e de um algoritmo de otimização (Seção 2.2.2), em que o *backpropagation* realiza um cálculo rápido de gradientes utilizados pelos algoritmos de otimização.

Dessa forma, a explicação dessa seção se resume em processos em que, dado o erro E que é obtido a partir da saída da rede (y) em relação a um *ground truth* (g), a primeira etapa é realizada com o cálculo da derivada parcial de E em relação a cada neurônio de saída, sendo formada por $\frac{\partial E}{\partial y_i}$ tendo que $i \in \mathbb{K}$ como a representação do tamanho de cada saída. Depois, com a aplicação da regra da cadeia de E em relação às entradas x_j , é possível obter a seguinte expressão ([Rumelhart et al., 1986](#)):

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_j}. \quad (2.19)$$

A partir da expressão $\frac{\partial E}{\partial x_j}$ é possível visualizar que o erro será afetado na saída (y) a medida que a entrada (x) for alterada, segundo comentado por [Rumelhart et al. \(1986\)](#), que também exemplifica essa função na Equação 2.20, ao considerar a entrada como função linear,

as conexões do neurônio i , assim como os pesos dos neurônios i e j :

$$\frac{\partial E}{\partial y_j} = \sum_j \frac{\partial E}{\partial x_j \cdot w_{ji}}. \quad (2.20)$$

2.3 Teste

A etapa de teste está relacionada com a avaliação do modelo treinado nas condições descritas na Seção 2.2. Todavia, para o teste deve-se considerar que o novo conjunto de dados seja desconhecido pela rede neural, ou seja, que não haja intersecção com o conjunto de treinamento, visto que o objetivo do treino está em validar a capacidade de abstração em relação à generalização do modelo.

Entendido isso, habitualmente, é encontrado um conjunto de dados com menos exemplos quando se trata do conjunto de testes (Goodfellow et al., 2016), sendo necessário que ambos os conjuntos de dados, tanto o de teste quanto o de treinamento, possuam uma mesma distribuição dos dados.

O processo de teste pode ser realizado entre as épocas do treinamento, assim, avaliando o desempenho do modelo em um conjunto desconhecido de dados e sendo possível avaliar se a rede está ou não melhorando a capacidade de generalização de acordo com o decorrer das épocas.

Assim, nesta seção são descritos alguns dos elementos utilizados para auxiliar nos teste, como as métricas de avaliação (Seção 2.3.1) e o método *cross validation* (Seção 2.3.2), além de fenômenos que obstruem o desenvolvimento, como *overfitting* e *underfitting* (Seção 2.3.3).

2.3.1 Métricas de Avaliação

Outro ponto adjutório à avaliação dos modelos, tanto para o conjunto de treinamento quanto para o conjunto de teste, é crucial para entender o desempenho e a eficácia do modelo ao longo do treinamento. Uma métrica fundamental nesse contexto é a acurácia, frequentemente representada como Acc , que mede a proporção de exemplos classificados corretamente pelo modelo. Matematicamente, a acurácia pode ser expressa pela Equação 2.21:

$$Acc = \frac{\text{Quantidade de predições corretas}}{\text{Quantidade de predições realizadas}}. \quad (2.21)$$

A acurácia, como demonstrado na Equação 2.21, basicamente se resume entre a razão

de acertos pela quantidade total de exemplos no conjunto e é adequada para situações em que se tem classes balanceadas. Mas, quando se tratam de classes desbalanceadas, o modelo em questão pode apresentar fenômenos de *underfitting* ou *overfitting*, como será explicado na Seção 2.3.3. Esses fenômenos podem ser identificados ao observar o desempenho da acurácia tanto no conjunto de treinamento quanto no conjunto de teste.

Dessa forma, é desejado obter $Acc_{treino} \approx Acc_{teste}$, e com acurácias altas. Sendo que as situações de *overfitting* possuem a característica de casos em que Acc_{treino} é exacerbadamente mais alta que Acc_{teste} . Por outro lado, tendo Acc_{treino} muito baixa, é comum em casos de *underfitting*.

Em situações de problemas relacionadas à regressão, é comum utilizar a métrica de Erro Quadrático Médio (ou MSE), conforme demonstrado na Equação 2.8. Essa métrica é amplamente empregada na avaliação de modelos de regressão devido à sua capacidade de quantificar a diferença entre os valores previstos pelo modelo e os valores reais observados nos dados de teste.

2.3.2 Validação Cruzada

Quando se trata na divisão entre os *datasets* de treino e teste, habitualmente é comentado sobre a configuração de 20% para teste e 80% para treino. Todavia, o fato de utilizar um conjunto fixo implica em uma incerteza estatística sobre a média estimada de erro do teste, visto que pode ocorrer um desbalanceamento imprevisto (Goodfellow et al., 2016).

Dessa forma, a validação cruzada, conhecida como *cross validation* em inglês, opera com o princípio de dividir o conjunto de dados em subconjuntos que compõem o conjunto original, permitindo cálculos iterativos por meio de seleções amostrais. Entre as técnicas mais comuns de *cross validation* para essa divisão, destacam-se os métodos *k-fold* e *leave-one-out*. Ambos mantêm a premissa de dividir aleatoriamente o conjunto de dados em subconjuntos de treinamento e teste, possibilitando a avaliação do algoritmo quanto à sua capacidade de generalização. O método *k-fold* divide o conjunto de dados em k grupos (ou dobras), onde cada grupo é utilizado uma vez como conjunto de teste enquanto os outros $k - 1$ grupos são usados como conjunto de treinamento. Já o método *leave-one-out* é uma forma extrema de *k-fold*, onde cada amostra é usada uma vez como conjunto de teste, enquanto todas as outras amostras formam o conjunto de treinamento. Essas técnicas são fundamentais para garantir uma avaliação robusta dos modelos de aprendizado de máquina, ajudando a estimar sua capacidade de generalização

para dados não vistos.

Dessa forma, o erro (ε) é computado no teste (T) pela média dos erros individuais (ε_i) em relação ao número de interações (N), como demonstrado na Equação 2.22:

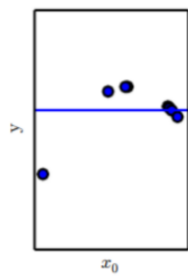
$$\varepsilon_T = \frac{1}{N} \sum_{i=1}^N \varepsilon_i. \quad (2.22)$$

2.3.3 *Overfitting e Underfitting*

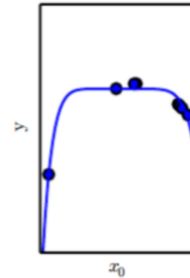
As situações de *overfitting* e *underfitting* são frequentemente observadas quando o modelo apresenta alta variância. No caso do *overfitting*, o modelo tende a memorizar os dados do conjunto de treinamento, perdendo a capacidade de generalização para novos dados, como os do conjunto de teste. Por outro lado, o *underfitting* ocorre quando o modelo não é capaz de aprender os padrões presentes nos dados de treinamento, resultando em um desempenho abaixo do esperado e uma baixa acurácia, que geralmente é definida pelo desenvolvedor como um critério mínimo de desempenho aceitável. Esses problemas podem prejudicar a eficácia do modelo em lidar com novos dados e comprometer sua utilidade em aplicações do mundo real.

A partir da representação na Figura 2.9 é possível exemplificar os fenômenos supracitados, em que na Figura 2.9a se visualiza uma situação característica de *underfitting* ao tentar utilizar um modelo linear para um tipo de problema não linear. Já na imagem 2.9b se encontra uma situação em que há uma função que é utilizada adequadamente para a situação, tendo esta convergido e com uma boa adaptação para demais situações com contextos similares. E, por fim, há a representação de uma situação com *overfitting* na Figura 2.9c, em que o conjunto está decorado e o modelo se ajustou demais ao conjunto de dados - situação que acontece devido ao uso de funções complexas para situações simples.

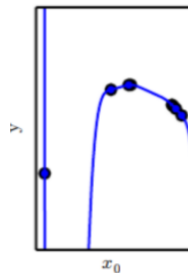
Figura 2.9: Fenômenos de *underfitting* e *overfitting*.



(a) Fenômeno de *underfitting*.



(b) Representação de função adequada.



(c) Fenômeno de *overfitting*.

Fonte: retirado e adaptado de [Goodfellow et al. \(2016\)](#).

Alerta-se que em situações em que a função de custo de treinamento continua decrescente e a função de custo de teste começa a aumentar, é provável que o modelo esteja iniciando uma circunstância de *overfitting*, já que casos de *underfitting* ocorrem quando ambas as funções de custo permanecem altas mesmo após muitas épocas.

Sendo assim, *underfitting* e *overfitting*, quando presentes, devem ser corrigidos, embora os ajustes dos métodos à complexidade da situação ainda sejam feitos a partir de tentativa e erro e com o auxílio de gráficos ([Goodfellow et al., 2016](#)).

2.4 Considerações Finais do Capítulo

Neste Capítulo 2, foram abordados temas desde a concepção inicial das redes neurais, apresentando as ideias propostas por [McCulloch e Pitts \(1943\)](#), até detalhes gerais do funcionamento das redes neurais. No entanto, é importante ressaltar que, após o surgimento das *deep learnings*, algumas redes foram direcionadas para resolver problemas específicos. Um exemplo é o uso generalizado das Convolutional Neural Networks (CNNs) no contexto de processamento de imagens e informações espaciais, tópico que será discutido no próximo capítulo.

3 REDES NEURAIS CONVOLUCIONAIS

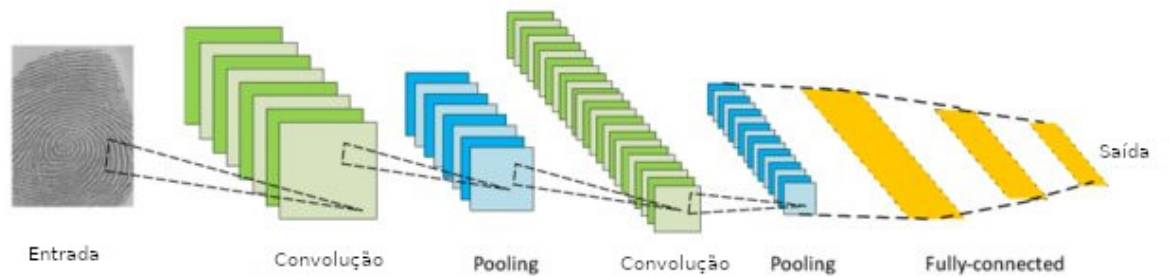
Dentre os diversos tipos de redes de aprendizado profundo, as mais amplamente utilizadas atualmente para lidar com visão computacional, processamento de imagens e possuindo um valor notável para o trabalho com dados espaciais (Goodfellow et al., 2016; Ponti e Da Costa, 2018; Ghosh et al., 2019), destacam-se as *Convolutional Neural Networks* (CNNs) (LeCun et al., 1999). As CNNs recebem esse nome por empregarem camadas convolucionais, sendo especialmente proeminentes e eficazes nesses contextos.

É inegável que as CNNs são consideradas o estado-da-arte em várias competições (Parkhi et al., 2015), além de serem modelos bioinspirados de sucesso (Goodfellow et al., 2016). Elas são capazes de simular pontos cerebrais responsáveis pela interpretação de impulsos visuais, reproduzindo propriedades do córtex visual. Além disso, essas redes têm a capacidade de extrair diferentes características, utilizando a combinação das camadas de convolução (Seção 3.1) juntamente com operações de *pooling* que tem o papel de compressão e serão detalhadas na Seção 3.2.

Entre a vasta gama de arquiteturas disponíveis para as CNNs, destacam-se algumas das mais influentes: AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan e Zisserman, 2015), ResNet (He et al., 2016), GoogLeNet (Szegedy et al., 2015), MobileNet (Howard et al., 2017) e DenseNet (Huang et al., 2017). Essas arquiteturas possuem contribuições significativas e têm sido aplicadas em uma variedade de contextos, cada uma com suas características e aplicações específicas.

Por meio da Figura 3.1 é representada uma arquitetura de um modelo indicando uma entrada, as camadas convolucionais, responsáveis pela extração de características dos dados de entrada, bem como o sistema de *pooling* e a camada de saída (ou camada densa).

Figura 3.1: Modelo de CNN.



Fonte: retirada e adaptado de [Minaee et al. \(2021b\)](#).

Dessa forma, nas seções seguintes, os elementos fundamentais que compõem uma rede neural convolucional serão minuciosamente explorados. Estes incluem a camada convolucional (Seção 3.1), a camada de *pooling* (Seção 3.2), o componente *dropout* (Seção 3.3) e a camada de saída (Seção 3.4) da rede. Adicionalmente, serão abordados temas complementares que impactam significativamente no desempenho das CNNs, tais como transferência de aprendizado (Seção 3.5) e aumento de dados (Seção 3.6). Por fim, será explorada uma das arquiteturas que exerceu considerável influência na evolução desse campo, a VGG (Seção 3.7).

3.1 Camada Convolucional

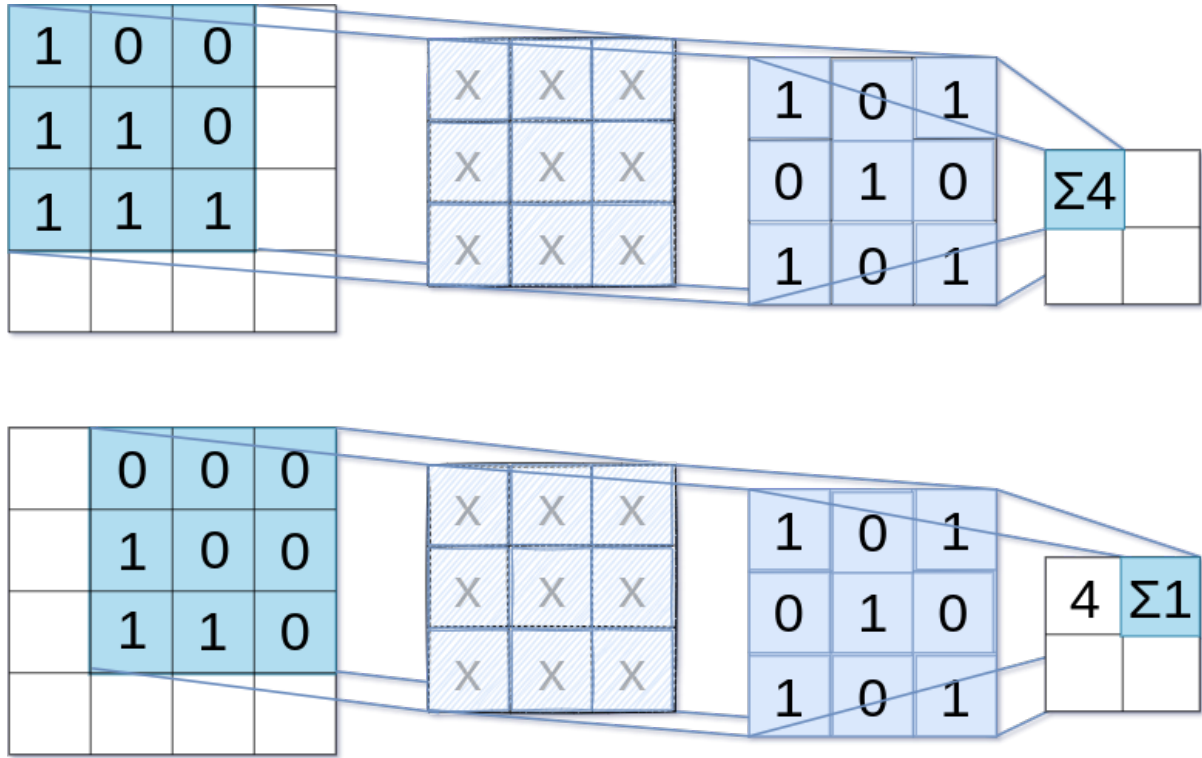
Na arquitetura das CNNs, a modelagem de cada neurônio presente na camada convolucional envolve a aplicação de um filtro na imagem em análise ([Ponti e Da Costa, 2018](#)). Esses filtros são compostos por pesos e a operação de convolução é categorizada como uma operação linear ([Goodfellow et al., 2016](#)), resultando nos chamados mapas de características (*feature maps*, em inglês).

É fundamental destacar que entre as camadas convolucionais existem N filtros, cujos parâmetros são definidos de acordo com a natureza do problema em questão, utilizando seus resultados para formar tensores, conforme comentado por [Ponti e Da Costa \(2018\)](#).

A operação de convolução pode ser ilustrada pela Figura 3.2, onde uma imagem de entrada é representada pela matriz de maior escala. Nesse processo, utiliza-se uma janela, conhecida como *kernel*, composta por pesos, que convolui com a matriz de entrada para extrair características específicas da imagem, mantendo a integridade das informações sobre sua disposição espacial.

Essas operações são fundamentais nas CNNs, pois permitem a extração eficiente de características importantes das imagens, o que é crucial para a eficácia e a precisão do processo de aprendizado da rede neural convolucional.

Figura 3.2: Representação do processo de convolução.



Fonte: do próprio autor.

Ainda na Figura 3.2, é possível observar o processo de deslizamento do *kernel* durante a operação de convolução, resultando em um dos valores presentes na saída. Esse processo é repetido até que toda a imagem seja percorrida, de maneira sequencial da esquerda para a direita e de cima para baixo, movimentando-se através das linhas e colunas.

O processo iterativo de aplicação da máscara sobre a imagem pode ser representado matematicamente pela Equação 3.1 (Gonzalez e Woods, 2009):

$$w(x, y) * f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t), \quad (3.1)$$

onde x e y representam coordenadas espaciais da matriz de entrada, $w(x, y)$ é o valor do *pixel* na posição (x, y) da janela e $f(x, y)$ é o valor do *pixel* na posição (x, y) da matriz de entrada. Além disso, vale citar que s e t , representam variáveis de iteração usadas para percorrer a máscara e,

m e n representam as dimensões da máscara, utilizadas para definir os limites de iteração nas somas duplas a e b , conforme especificado pelas Equações 3.2 e 3.3 (Gonzalez e Woods, 2009):

$$a = (m - 1)/2, \quad (3.2)$$

$$b = (n - 1)/2. \quad (3.3)$$

É crucial ressaltar que a convolução é um processo de média móvel, isto é, implica no cálculo da média ponderada de uma janela específica na imagem, correspondente ao tamanho do *kernel* utilizado. Nesse contexto, o *kernel* define os pesos aplicados aos pontos da janela. Consequentemente, à medida que o *kernel* desliza pela imagem, novas janelas são definidas e suas médias ponderadas são computadas.

Essa abordagem de convolução, com seu deslizamento e cálculo ponderado, é essencial para a extração de informações significativas da imagem, contribuindo para a identificação de padrões relevantes ao longo do processo de aprendizado da rede neural convolucional.

3.2 Camada de *Pooling*

Dentro do contexto das camadas de *pooling*, é relevante mencionar que essas camadas são amplamente empregadas nas CNNs com o propósito de reduzir o tempo de treinamento, já que sua função primária é diminuir a dimensionalidade dos mapas de características entre as iterações da rede.

É digno de nota que a adaptação e evolução desses métodos de *pooling* têm recebido crescente atenção na literatura recente (Sabri et al., 2020; Zafar et al., 2022). Cada modificação busca contribuir para a solução de problemas específicos, evidenciando o constante desenvolvimento dessas técnicas na área de visão computacional e aprendizado profundo.

Ademais, vale ressaltar que na literatura existem diversos modelos de *pooling* que se destacam no uso em redes de aprendizado profundo. Entre eles, destacam-se o *max pooling*, *average pooling*, *median pooling* e *weighted average* (Goodfellow et al., 2016). Há também outros modelos comumente encontrados em redes mais modernas, como o *global average pooling*, *global max pooling* e *astrous spatial pyramid pooling*, principalmente em aplicações de segmentação ou com propostas alternativas. Alguns desses modelos serão discutidos a seguir.

3.2.1 Max Pooling

O *max pooling* é uma das técnicas mais conhecidas e amplamente utilizadas para realizar o *pooling* em CNNs (Zafar et al., 2022; Paul e Chaki, 2019). Esta técnica é preferida devido à sua simplicidade, não exigindo ajuste de parâmetros, além de atender de forma eficiente à necessidade de reduzir a dimensionalidade das entradas (Boureau et al., 2010).

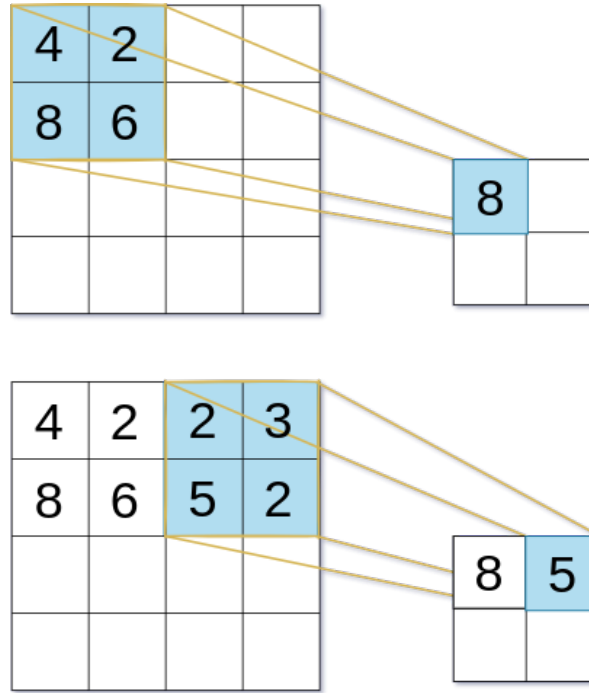
O funcionamento do *max pooling*, denotado por $f_{\max}(\mathbf{X})$, pode ser representado pela Equação 3.4:

$$f_{\max}(\mathbf{X})_{i,j,k} = \max_{m,n} X_{i \cdot s_x + m, j \cdot s_y + n, k}. \quad (3.4)$$

Nesta equação, $\mathbf{X}$ representa o valor de entrada, (i, j) são os índices do valor de saída e k é a quantidade de canais (ou camadas) associadas ao valor de entrada. Além disso, s_x e s_y representam os tamanhos de deslocamento (ou *stride*) horizontal e vertical, respectivamente. Finalmente, m e n representam as posições dentro da janela de *pooling* ao longo das dimensões horizontal e vertical, respectivamente, sendo que essas variáveis variam de acordo com o tamanho da janela de *pooling* e o passo (*stride*) ao percorrer a matriz de entrada para realizar a operação de *max pooling*.

O resultado da operação de *max pooling* é obtido utilizando um *kernel* como ilustrado na Figura 3.3 que representa a execução de deslizamento de um *kernel* 2×2 com um *stride* igual a 2 mostrando que a técnica captura apenas os maiores valores do mapa de características.

Figura 3.3: *Max pooling*.



Fonte: do próprio autor.

É relevante destacar que a aplicação da técnica de *max pooling* adiciona uma leve invariância por translação ao processo. No entanto, de modo geral, isso não afeta de maneira significativa os valores de saída da camada de *pooling* (Boureau et al., 2010).

A técnica de *max pooling* é conhecida por preservar características importantes da entrada, ao mesmo tempo em que reduz a dimensionalidade dos mapas de características, tornando-os mais computacionalmente eficientes e facilitando a detecção de padrões por camadas subsequentes da rede neural convolucional. Todavia, essa técnica não se demonstra adequada quando se trata da preservação de informação da entradas (Liu et al., 2019), visto que se realiza a redução de dimensionalidade de forma local.

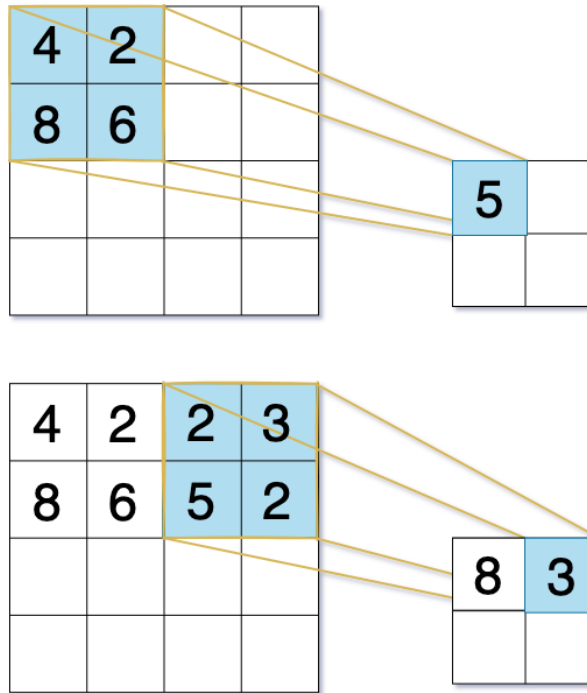
3.2.2 *Average Pooling*

A técnica de *average pooling*, semelhante à técnica de *max pooling* descrita na Seção 3.2.1, é considerada uma técnica estabelecida e tradicional para aplicação em CNNs (Zafar et al., 2022; Paul e Chaki, 2019). O funcionamento de ambas as técnicas é bastante semelhante, diferenciando-se principalmente pela aplicação da média em vez da função $\max$ para obter o resultado do *Average Pooling*, denotado por $f_{avg}(\mathbf{X})$, conforme ilustrado na Equação 3.5:

$$f_{avg}(\mathbf{X})_{i,j,k} = \frac{1}{f_x \cdot f_y} \sum_{m,n} X_{i \cdot s_x + m, j \cdot s_y + n, k}. \quad (3.5)$$

Nesta equação, $\mathbf{X}$ representa o valor de entrada, (i, j) são os índices do valor de saída e k é o número de camadas. Os parâmetros s_x e s_y representam os valores de *stride* vertical e horizontal, respectivamente, enquanto os tamanhos dos filtros f_x e f_y determinam a janela de *pooling*, centrada nos índices de saída (i, j) , tendo m e n referentes às posições locais dentro da janela de *pooling*, assim como na Equação 3.4. Uma representação visual dessa fórmula é ilustrada na Figura 3.4 que demonstra semelhanças quando comparada à Figura 3.3, principalmente, no que diz ao funcionamento com o deslizamento de uma janela 2×2 com *stride* igual a 2.

Figura 3.4: *Average pooling*.



Fonte: do próprio autor.

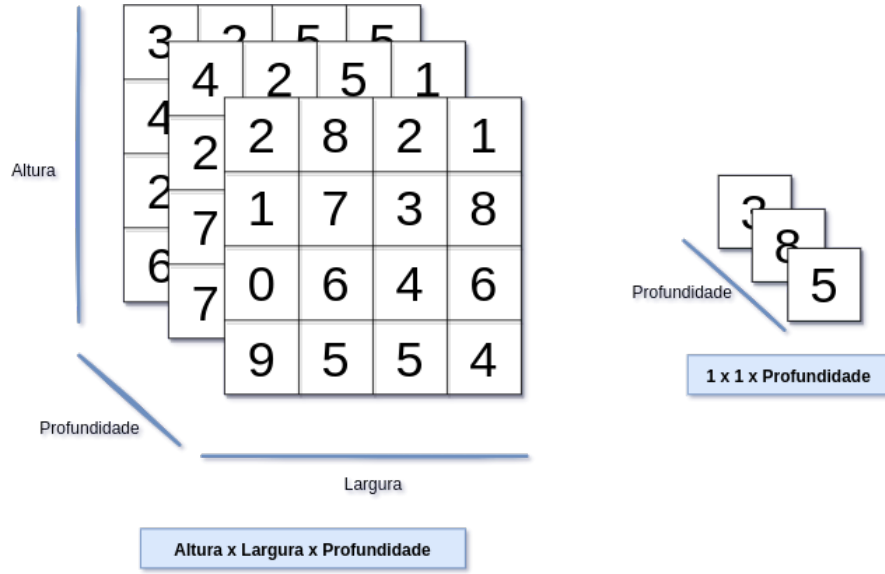
Por fim, é relevante mencionar que o resultado visual obtido pelo *average pooling* tende a extrair características de maneira mais suave do que o *max pooling*. Enquanto o *max pooling* enfatiza características mais proeminentes, como bordas, o método de *average pooling* tende a extrair características mais suaves, como informações relacionadas às cores. Essa técnica foi utilizada na primeira rede neural profunda baseada em convolução (LeCun et al., 1998). Contudo, observam-se desafios semelhantes em relação à preservação da espacialidade das entradas (Liu

et al., 2019), como ocorre também no algoritmo de *max pooling*.

3.2.3 Global Average Pooling

A técnica de *Global Average Pooling*, por sua vez, permite a redução dos mapas de características a um único valor por camada, como ilustrado na Figura 3.5.

Figura 3.5: *Global Average Pooling*.



Fonte: do próprio autor.

Neste caso, percebe-se que essa técnica é uma variação do *Average Pooling*. Em vez de usar um *kernel* e *strides*, o *Global Average Pooling* aplica a média em todos os valores de entrada. Isso pode ser descrito pela Equação 3.6:

$$f_{\text{global avg}}(\mathbf{X})_k = \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n \mathbf{X}_{i,j,k}, \quad (3.6)$$

onde k representa o índice das camadas, $\mathbf{X}$ denota o valor de entrada, m é o tamanho do eixo x e n é o tamanho do eixo y do mapa de características.

É importante ressaltar que o *Global Average Pooling* também pode ser empregado como uma operação para substituir as camadas totalmente conectadas (em inglês, *fully connected layers*) em CNNs. Nessa abordagem, o método utiliza a média de cada mapa de características para alimentar uma camada de saída com Softmax, que será discutida na Seção 3.4, como foi realizado no trabalho de Kumar et al. (2021).

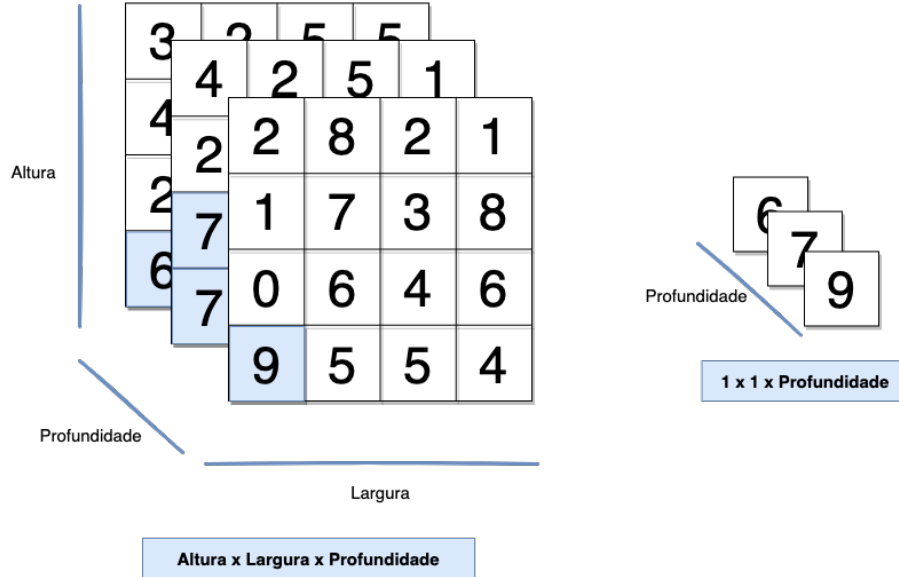
3.2.4 Global Max Pooling

O método de *Global Max Pooling* possui semelhanças em comparação com os métodos de *Max Pooling* e *Global Average Pooling*. Este método aplica a função de maximização ao longo de todo o mapa de características de cada camada dos valores de entrada, como pode ser visualizado na Equação 3.7:

$$f_{\text{global max}}(\mathbf{X})_k = \max_{i,j} \mathbf{X}_{i,j,k}, \quad (3.7)$$

onde k representa o índice das camadas e $\mathbf{X}$ denota o valor de entrada. Essa representação também pode ser exemplificada por meio da Figura 3.6.

Figura 3.6: *Global Max Pooling*.



Fonte: do próprio autor.

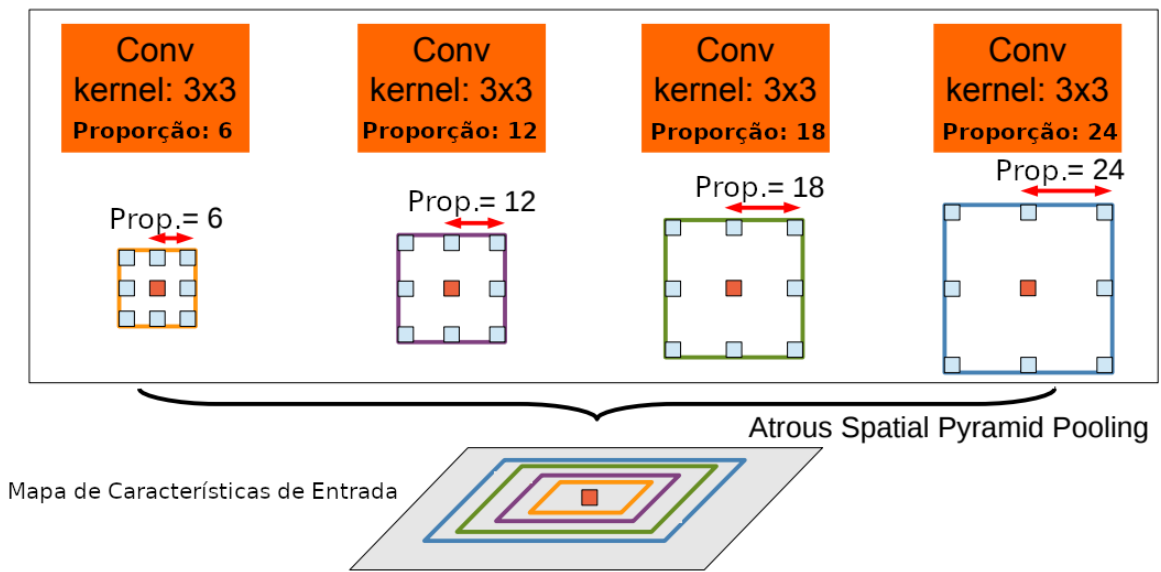
Entretanto, é crucial ressaltar que os métodos convencionais não preservam informações espaciais. Isso se aplica ainda mais aos métodos globais, os quais resumem toda a representação da entrada em um único valor (Christlein et al., 2019). Essa perda de informações espaciais pode ser limitante em algumas tarefas, especialmente em contextos onde a disposição e a relação entre os elementos na entrada têm um papel significativo na análise ou no reconhecimento de padrões.

3.2.5 Atrous Spatial Pyramid Pooling (ASPP)

Além da proposta de preservação de escala, embora ainda sem manter a espacialidade, foi introduzido o método *Atrous Spatial Pyramid Pooling* (ASPP) por Chen et al. (2018). Este

tipo de operação de *pooling* está intrinsicamente ligado às segmentações semânticas (Mohan e Valada, 2020) (que serão discutidas no Capítulo 5). A principal vantagem desse algoritmo é a capacidade de capturar objetos e contexto relevantes da imagem em várias escalas distintas. Essa funcionalidade está associada à reamostragem utilizando convoluções com diferentes taxas de dilatação para cada *kernel*, permitindo explorar um mapa de características com múltiplos campos de visão antes mesmo do processo de convolução (Chen et al., 2018). Um exemplo desse módulo pode ser visualizado na Figura 3.7.

Figura 3.7: Exemplo ilustrativo do ASPP.



Fonte: Adaptado de Chen et al. (2018).

O funcionamento do ASPP, que realiza ajustes no mapa de características para calcular informações em múltiplas escalas, pode ser expresso pela Equação 3.8, proposta por Chen et al. (2018):

$$y[i] = \sum_k x[i + r \cdot k]w[k], \quad (3.8)$$

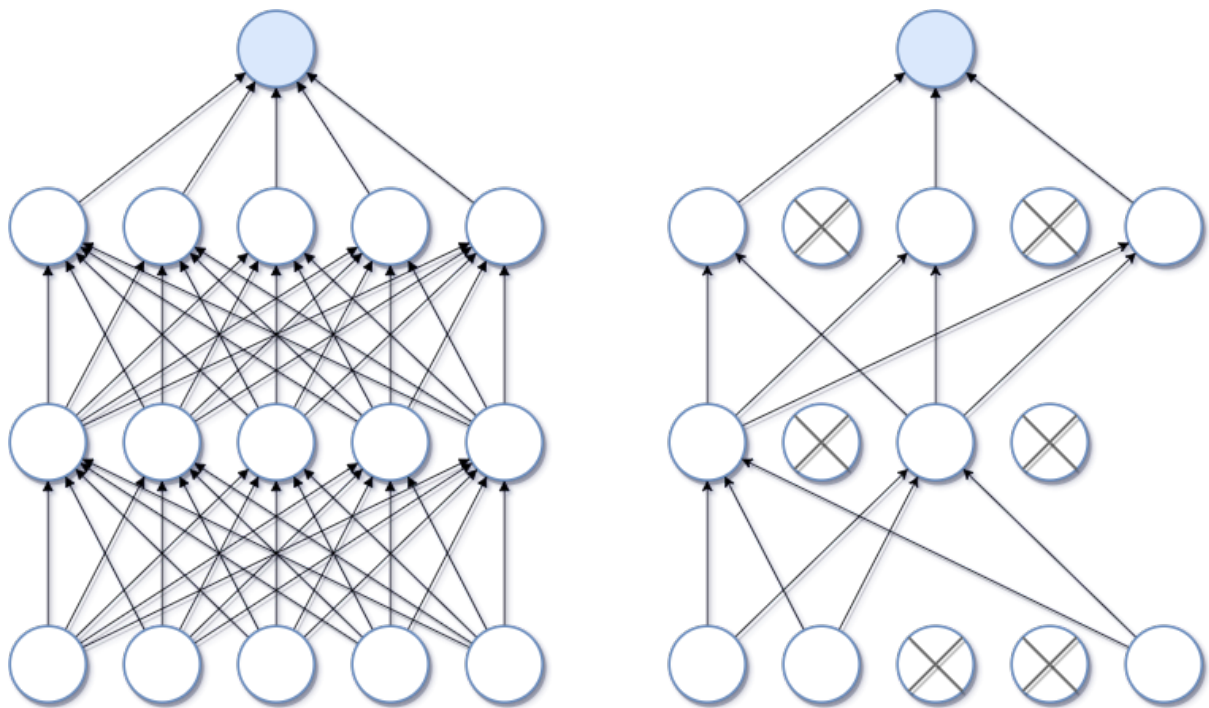
onde r representa uma taxa de dilatação que determina as *strides* para amostrar o mapa de características de entrada x , tendo que para cada *pixel* i na saída y e com o filtro w , a convolução dilatada (em inglês, *atrous convolution*) sendo aplicada.

3.3 Dropout

Dentre as estratégias para mitigar o *overfitting*, conforme discutido na Seção 2.3.3, destaca-se a técnica de *dropout* (Goodfellow et al., 2016). Esta técnica, embora não seja aplicada durante a fase de teste, consiste em desligar aleatoriamente neurônios em camadas ocultas de uma rede neural convolucional (CNN). O objetivo do *dropout* é reduzir o viés dos neurônios e aumentar a importância dos neurônios restantes.

O processo de *dropout* é ilustrado na Figura 3.8, que mostra apenas as conexões entre os neurônios restantes, destacados no lado direito da Figura 3.8.

Figura 3.8: Processo de *dropout*.



Fonte: do próprio autor.

A técnica de *dropout* é uma estratégia eficaz para regularização em redes neurais, pois ajuda a evitar que a rede se ajuste excessivamente aos dados de treinamento, tornando-a mais capaz de generalizar para dados novos e não vistos durante o treinamento.

3.4 Camada de Saída

Após a composição de várias camadas de convolução, utilizando filtros, e a aplicação de camadas de *pooling*, os modelos de CNN geralmente incluem uma camada de saída. Esta

camada é fundamental para a determinação das classes identificadas na imagem. Vale destacar que essa camada não tem um impacto significativo no desempenho em relação ao tempo de convergência do modelo, nem interfere no progresso e evolução do processo de treinamento, sendo que o processo de inferência é muito similar ao usado pelas redes neurais no Capítulo 2.

Por fim, para gerar a saída final, é comum utilizar funções de ativação como a Softmax (abordada na Seção 2.1.5). A aplicação da Softmax permite a identificação da classe de um objeto específico em uma imagem, por exemplo. Essa função de ativação é utilizada para a tarefa de classificação, fornecendo as probabilidades de pertencimento a cada classe no conjunto de dados.

3.5 Transferência de Aprendizado

A transferência de aprendizado é uma técnica valiosa no campo da aprendizagem profunda, que sustenta a ideia de usar conhecimento adquirido ao resolver um problema para ajudar a resolver um problema diferente, mas relacionado. Particularmente na aprendizagem profunda, a transferência de aprendizado tem se mostrado benéfica, especialmente, quando se lida com conjuntos de dados com limitações (Pan e Yang, 2010).

A mais comum estratégia de aplicação da transferência de aprendizado em CNNs é o uso de modelos pré-treinados, onde uma rede treinada em uma tarefa grande e bem definida, como a classificação de imagens em ImageNet (Deng et al., 2009), é reutilizada em outra tarefa específica. Isso é feito substituindo as últimas camadas do modelo pré-treinado para se ajustar à tarefa específica e treinando o modelo para ajustar esses novos parâmetros (Yosinski et al., 2014). Com essa mudança, as camadas ou pesos reaproveitados veêm inicialmente de modo “congelado”, de modo que não há atualização dos erros dessa camada no processo de retropropagação.

Essa abordagem é altamente eficaz, uma vez que durante a fase inicial do treinamento, conhecida como *warmup*, os modelos aprendem características de baixo nível nas camadas iniciais e gradualmente capturam características mais abstratas nas camadas posteriores. A premissa é que as camadas iniciais do modelo extraem recursos genéricos, como bordas e texturas, que são fundamentais para uma ampla variedade de tarefas, enquanto as camadas finais são mais especializadas na tarefa específica para a qual o modelo foi originalmente treinado (Yosinski et al., 2014).

É importante ressaltar que a aplicação da transferência de aprendizado também contribui significativamente para mitigar o *overfitting* (Géron, 2017). Após a fase de *warmup*, ocorre

o gradual *descongelamento* das camadas, iniciando o processo de ajuste fino (ou em inglês, *fine-tuning*), permitindo que as camadas se adaptem melhor ao problema atual. Quanto mais dados estiverem disponíveis no conjunto, mais camadas ou blocos podem ser descongelados para ajustes finos (Géron, 2017).

Utilizando transferência de aprendizado, a comunidade pode economizar tempo e recursos computacionais, evitando treinar uma rede complexa do zero para cada nova tarefa. Além disso, modelos pré-treinados podem ajudar a melhorar o desempenho do aprendizado, principalmente quando o conjunto de dados da nova tarefa é pequeno ou semelhante ao conjunto de dados inicial do modelo pré-treinado (Pan e Yang, 2010).

Apesar dessas vantagens, aplicar transferência de aprendizado em CNNs não é uma tarefa trivial. A seleção e ajuste das camadas para a tarefa pretendida, bem como a seleção das políticas adequadas de otimização e a taxa de aprendizado, são desafios que podem afetar a eficácia da transferência de aprendizado (Yosinski et al., 2014).

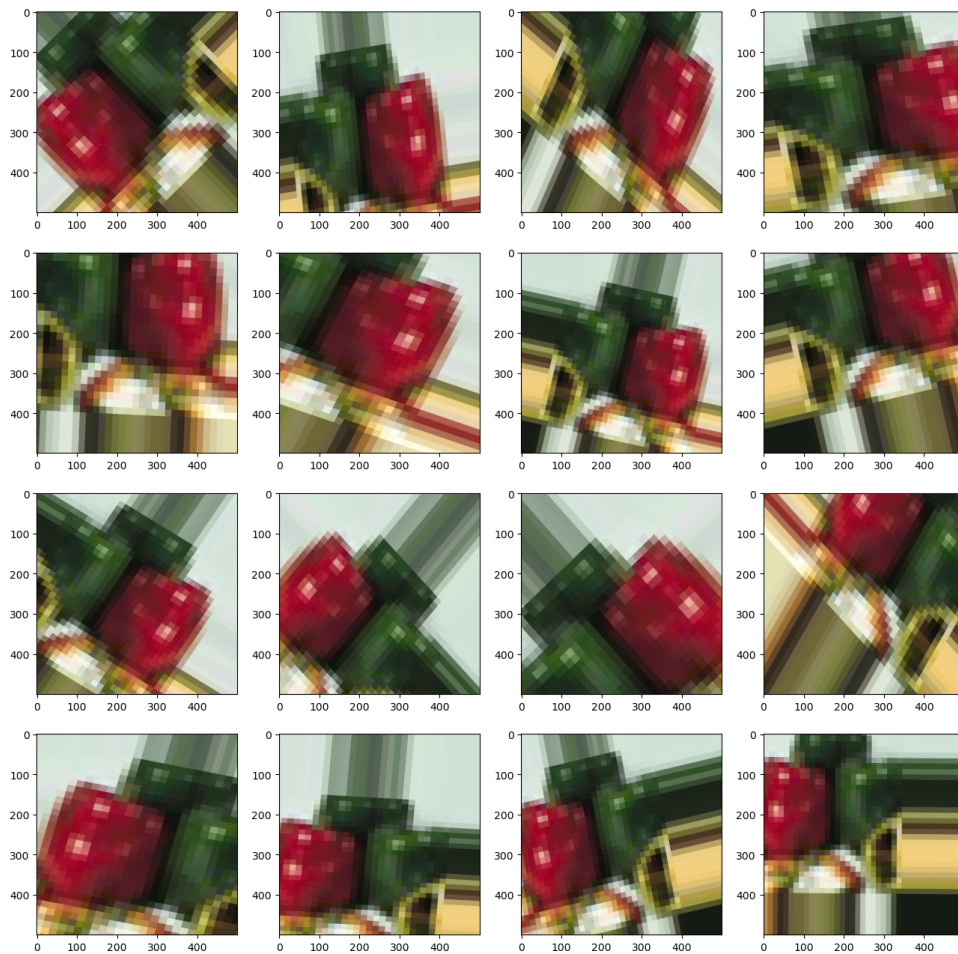
Não obstante, a transferência de aprendizado continua sendo uma estratégia amplamente vencedora na atual prática de aprendizagem profunda e sua aplicação está em crescimento contínuo, especialmente em tarefas como análise de sentimentos, detecção de objetos, reconhecimento facial, dentre outros (Pan e Yang, 2010).

3.6 Aumento de Dados

A técnica conhecida como aumento de dados, ou *data augmentation* em inglês, é uma estratégia que visa mitigar o *overfitting* e aumentar a robustez do modelo. Essa abordagem consiste em aplicar transformações geométricas e de intensidade nas imagens durante o processo de treinamento, introduzindo variações nas instâncias para enriquecer o conjunto de dados (Géron, 2017).

A utilização desse método apresenta a vantagem de aumentar a invariância do modelo diante de alterações nas condições de captura das imagens. Por meio dessas transformações, é possível criar diversas variações em termos de escala, posição, cor, luminosidade, saturação e contraste dos objetos, promovendo uma maior capacidade de generalização do modelo. Isso assegura que o reconhecimento das figuras ocorra em diferentes contextos adversos. A Figura 3.6 ilustra exemplos de algumas dessas técnicas de aumento de dados aplicadas a uma imagem.

Figura 3.9: Exemplo de aumento de dados com transformações geométricas em um exemplo de pimentão.



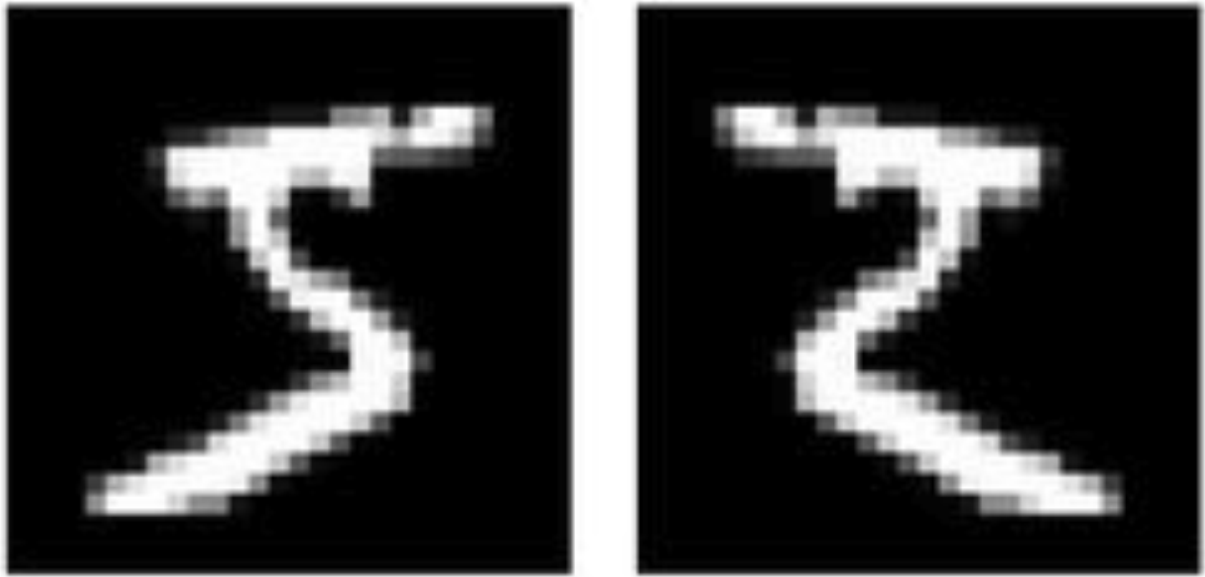
Fonte: do próprio autor, utilizando o conjunto de dados CIFAR 100(Bossard et al., 2014).

Como mencionado anteriormente, é importante destacar que o aumento de dados geralmente envolve o uso de transformações geométricas e de intensidade. As transformações geométricas são responsáveis por aplicar mapeamentos e alterações nos *pixels* das imagens, incluindo rotação, cisalhamento, translação, redimensionamento e outras técnicas comuns usadas em processos de registro de imagens (Pedrini e Schwartz, 2008). Por outro lado, as transformações de intensidade manipulam as cores e os valores dos *pixels*, modificando-os por meio de operações matemáticas como raiz quadrada, potenciação, logaritmos e outras técnicas similares (Pedrini e Schwartz, 2008).

No entanto, é crucial salientar que a seleção das transformações deve ser feita de maneira personalizada para cada problema específico a ser abordado. Isso ocorre porque, além de aumentar a carga computacional, muitas vezes as transformações podem levar à degradação do desempenho do modelo (Carneiro et al., 2021a; Hyttinen et al., 2020). Essas transformações

podem modificar o rótulo original do exemplo utilizado ou gerar casos que resultem em falsos positivos ou negativos, como no caso da inversão aplicada ao conjunto de dados MNIST (LeCun et al., 2010), ilustrado na Figura 3.10, e o uso de *Generative Adversarial Networks* (GANs) em áreas sensíveis, como na medicina.

Figura 3.10: Exemplo de inversão no conjunto de dados MNIST, em que o dígito “5” é alterado para “2”.



Fonte: do próprio autor, utilizando o conjunto de dados LeCun et al. (2010).

Por fim, é importante destacar que o processo de aumento de dados geralmente ocorre durante o treinamento do modelo, garantindo que os exemplos utilizados para o aprendizado nunca sejam idênticos, contribuindo para a robustez do modelo. Outra abordagem é aplicar o aumento de dados antes do treinamento, mas essa prática pode resultar em um maior consumo de memória e armazenamento.

3.7 VGG

A Rede Neural Convolutiva *Visual Geometry Group* (VGG), desenvolvida pelos pesquisadores do grupo VGG na Universidade de Oxford (Simonyan e Zisserman, 2015), tem sido uma das arquiteturas mais influentes no campo da Visão Computacional. A VGG proporcionou avanços significativos no entendimento das redes neurais convolutivas por sua simplicidade arquitetônica e eficiência para lidar com tarefas de aprendizado profundo (Simonyan e Zisserman, 2015).

de generalização da rede (Simonyan e Zisserman, 2015).

No entanto, a VGG também tem suas limitações. Devido à sua arquitetura profunda, a VGG demanda alto poder computacional e detém grande uso de memória. Além disso, a ausência de camadas totalmente conectadas reduziu a capacidade de modelar dependências de alto nível entre os recursos (Simonyan e Zisserman, 2015).

Apesar dessas limitações, a VGG continua sendo amplamente usada na comunidade de aprendizado profundo, principalmente como ponto de partida para a transferência de aprendizado, onde os pesos pré-treinados da VGG são usados como inicialização de outras tarefas de aprendizado. Além disso, a VGG forneceu um ponto de partida sólido para o desenvolvimento de arquiteturas mais profundas e eficientes, como a ResNet (He et al., 2016). Ao introduzir a ideia de profundidade nas redes neurais convolutivas, a VGG marcou um passo importante na evolução das redes de aprendizado profundo.

3.8 Considerações Finais do Capítulo

Neste capítulo é evidenciada a importância das CNNs na abordagem de problemas espaciais, especialmente no domínio de imagens. Foram abordados detalhes sobre como as características são extraídas por meio de filtros e camadas convolucionais (Seção 3.1), além da forma como as técnicas de *pooling* (Seção 3.2) são empregadas para reduzir a dimensionalidade dos dados gerados pelas convoluções. Também se destaca a relevância das camadas de saída (Seção 3.4) na definição das classes presentes na imagem, assim como a aplicação de *dropout* (Seção 3.3) para mitigar problemas de *overfitting* (Seção 2.3.3), formando assim, arquiteturas profundas, como é o caso das VGGs (Seção 3.7).

No entanto, o próximo capítulo se concentrará na segmentação de imagens, uma tarefa mais específica que não requer necessariamente o uso de redes profundas. Muitos métodos podem ser explorados de forma mais artesanal e menos complexa, como será discutido na Seção 4.1.

4 SEGMENTAÇÃO DE IMAGENS

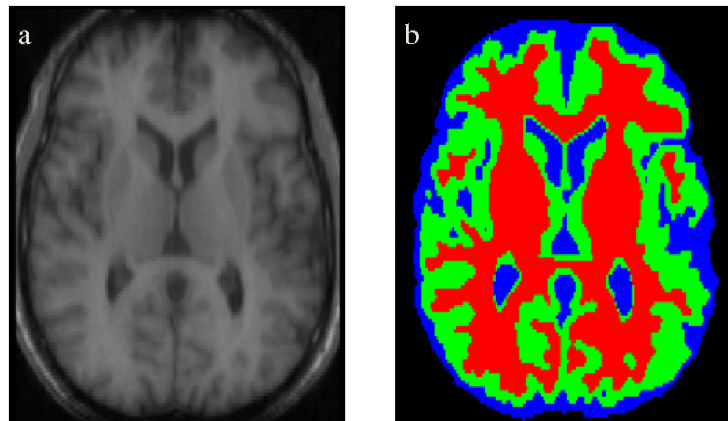
A segmentação de imagens é uma área que se enquadra nas disciplinas de visão computacional e processamento digital de imagens, visando principalmente a distinção de áreas ou propriedades específicas em uma imagem (Haralick e Shapiro, 1985; Yuheng e Hao, 2017; Ghosh et al., 2019). Este processo possibilita análises mais precisas ou a separação de áreas de interesse de acordo com os objetivos da análise.

A segmentação de imagens desempenha um papel fundamental em várias aplicações, incluindo reconhecimento facial (Yuheng e Hao, 2017). Além disso, áreas como medicina, veículos autônomos, sensoriamento remoto e outras também se beneficiam significativamente dessas técnicas. Para ilustrar, as Figuras 4.1a e 4.1b representam segmentações faciais e médicas, respectivamente.

Figura 4.1: Exemplos de segmentações de imagens.



(a) Segmentação de imagens faciais.



(b) Segmentação de imagem médica.

Fonte: Nirkin et al. (2018) e Withey e Koles (2008), respectivamente.

Para demonstrar o funcionamento das segmentações de maneira matemática, consideramos um conjunto de imagens I definido em um domínio $\Omega \subset \mathbb{R}^d$, onde $d \geq 2$. A imagem

observada I_0 e os predicados lógicos $P_1, \dots, P_n$ são utilizados para verificar n afirmações expressas com base em características da imagem, como bordas, suavidade, textura ou cor. O predicado $P_k(A)$ é verdadeiro se todos os pontos de $A \subseteq \Omega$ satisfizerem a k -ésima afirmação. Essa abordagem é inspirada em trabalhos anteriores, como mencionado por [Antonelli et al. \(2022\)](#).

Por exemplo, para realizar uma segmentação de duas regiões em uma imagem em tons de cinza normalizada I_0 , pode-se definir $n = 2$ afirmações simples. Essas afirmações estão relacionadas ao nível de cinza da intensidade da luz e são utilizadas para distinguir entre o plano de fundo e o primeiro plano, como demonstrado na Equação 4.1:

$$P_1(A) = \{I^*(x) < \alpha, \forall x \in A\}, \quad P_2(A) = \{I^*(x) \geq \alpha, \forall x \in A\}, \quad (4.1)$$

em que $I^*(x)$ representa uma aproximação adequada de I_0 e $\alpha \in (0, 1)$ é um valor escolhido de forma apropriada.

Compreendido o conceito de segmentação de imagens, esta área compreende uma ampla gama de técnicas, com a Seção 4.1 abordando os métodos de segmentação tradicionais, que continuam sendo amplamente estudados e aplicados até os dias atuais.

4.1 Métodos e Técnicas Tradicionais

Quando se trata de segmentações, algumas técnicas são muito conhecidas e utilizadas, sendo que algumas são calculadas utilizando apenas convoluções matriciais simples ([Yuheng e Hao, 2017](#)).

Sendo assim, nas próximas seções algumas técnicas de segmentação serão discutidas, das quais pode-se citar as segmentações baseadas em regiões (Seção 4.1.1), bordas (Seção 4.1.2), baseada em grafos (Seção 4.1.3), *watershed* (Seção 4.1.4), *liveware* (Seção 4.1.5), *superpixel* (Seção 4.1.6), agrupamentos (Seção 4.1.7) e até mesmo as que fazem uso de redes neurais (Seção 4.1.8) com a aplicação de alguns conceitos que foram comentados no Capítulo 2.

4.1.1 Segmentação Baseada em Regiões

Alguns tipos de segmentação comumente trabalham com os valores dos *pixels*, de modo que seja possível segregar áreas e regiões de interesse de regiões que não são possuem relevância. Dentre as diversas técnicas, vale destacar a técnica de *Threshold Segmentation*, ou em português, segmentação limiar, a qual é desenvolvida e aplicada em diversos trabalhos ([Yanowitz](#)

e Bruckstein, 1989).

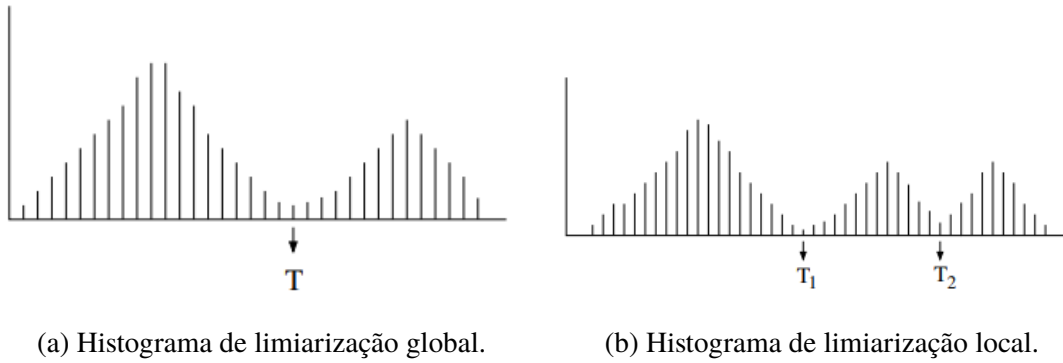
De um modo geral, a segmentação por meio da determinação de um limiar acontece em imagens que utilizam a escala de cinza ou algum sistema de cor que possua um canal voltado para intensidade, como o sistema HSV (Schneider et al., 2003), visto que quando no sistema correto, há a determinação de um valor entre o fundo e a área de interesse da imagem. Dessa forma, os *pixels* acima do valor médio determinado são categorizados como área de intensidade e os inferiores são categorizados como fundo (Pedrini e Schwartz, 2008), como é possível observar na Equação 4.2, em que T representa o limiar determinado, $f(x, y)$ é um *pixel* da imagem f e $g(x, y)$ é este *pixel* limiarizado:

$$g(x, y) = \begin{cases} 0, & \text{se } f(x, y) \leq T \\ 1, & \text{se } f(x, y) > T \end{cases}. \quad (4.2)$$

Sobre *threshold segmentation*, vale citar que há uma série de técnicas para determinar o valor de limiarização como é trabalhado por Atta et al. (2018), mas habitualmente utiliza-se da média (Atta et al., 2018; Yanowitz e Bruckstein, 1989; Yuheng e Hao, 2017) ou de técnicas adaptativas que são citadas em trabalhos, como Yanowitz e Bruckstein (1989), mas que ganham repercussões abrangentes e é evidenciada na técnica Otsu (Otsu, 1979), principalmente quando se trata de segmentações com um limiar global como demonstra o histograma na Figura 4.2a e o exemplo prático na Figura 4.3c. Todavia, quando as regiões de interesse estão divididas, faz-se necessário a divisão de limiares locais, como pode ser observado no histograma da Figura 4.2b, Equação 4.3 ou na Figura 4.3d:

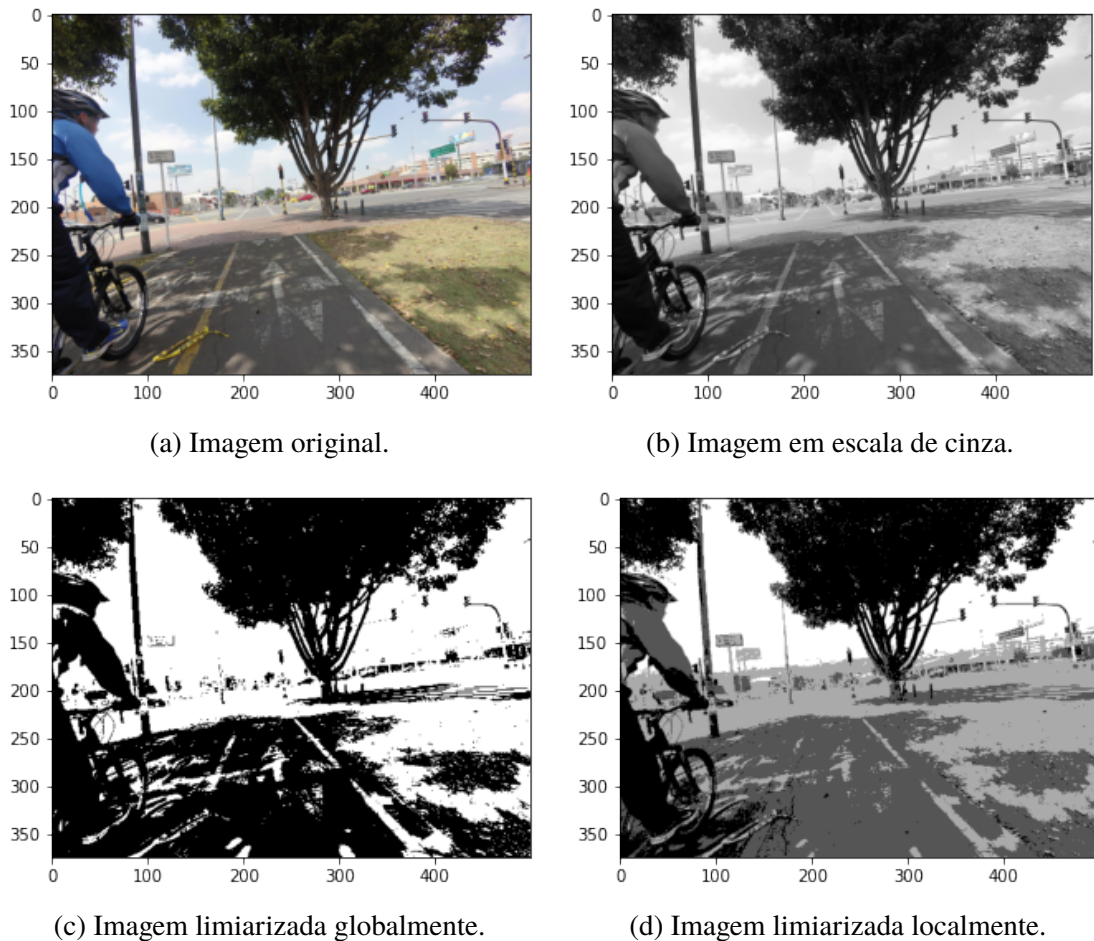
$$g(x, y) = \begin{cases} v1, & \text{se } f(x, y) \leq T \\ v2, & \text{se } T_1 < f(x, y) \leq T_2 \\ v3, & \text{se } f(x, y) > T_2 \end{cases}. \quad (4.3)$$

Figura 4.2: Histogramas de limiarizações.



Fonte: [Pedrini e Schwartz \(2008\)](#).

Figura 4.3: Limiarizações.



Fonte: retirado e adaptado de [Neuhold et al. \(2017\)](#).

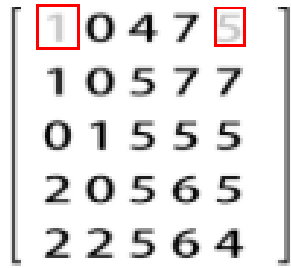
Por fim, no contexto da segmentação baseada em regiões, destaca-se a técnica de crescimento de regiões, também conhecida como *Region Growing*. Essa abordagem consiste em definir um limiar e distribuir sementes na imagem, de forma que os *pixels* vizinhos sejam incorporados à região se as diferenças entre eles e as sementes forem inferiores ao limiar estabelecido.

O efeito desse procedimento pode ser visualizado na Figura 4.4 ou representado matematicamente pela Equação 4.4, onde $P(R)$ é o predicado da região, $f(r, s)$ é o *pixel* semente, e $f(x, y)$ representa um *pixel* vizinho ao da semente na vizinhança-8:

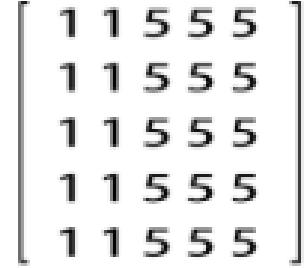
$$P(R) = \begin{cases} 1, & \text{se } |f(x, y) - f(r, s)| \leq T \\ 0, & \text{caso contrário} \end{cases} \quad (4.4)$$

Esta técnica é ilustrada na Figura 4.4, onde a Figura 4.4a mostra a imagem original com os *pixels* sementes em vermelho ($f(r, s)$). As Figuras 4.4b e 4.4c representam, respectivamente, o crescimento da região com limiares $T = 3$ e $T = 6$.

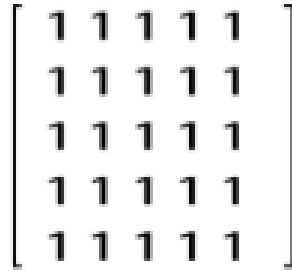
Figura 4.4: Ilustração do crescimento de região.



(a) Imagem original, sendo em vermelho os *pixels* sementes ($f(r, s)$).



(b) Crescimento de região com $T = 3$.



(c) Crescimento de região com $T = 6$.

Fonte: retirado e adaptado de Yuheng e Hao (2017).

Esta técnica é utilizada para agrupar *pixels* que possuem similaridades entre si, formando regiões coesas na imagem.

Por fim, ainda sobre o aspecto de crescimento de região, salienta-se que é de extrema importância a determinação das sementes e do valor limiar (Yuheng e Hao, 2017; Pedrini e Schwartz, 2008), além de ser necessário ter as condições para determinação de sistema de cores e canais semelhantes às de *threshold segmentation*.

4.1.2 Segmentação por Bordas

Em meio às categorias de segmentação tradicional estão as segmentações por descontinuidade, com subcategorias de segmentação por pontos, retas, junções e, finalmente, por bordas (Pedrini e Schwartz, 2008). Correntemente, é a partir dessas segmentações que são construídos alguns dos filtros convolucionais que foram explorados no Capítulo 3.

Sendo assim, é de referir que as bordas são definidas como limites que possuem regiões com contrastes discrepantes em relação ao seus níveis de cinza ou que separam duas regiões homogêneas (Pedrini e Schwartz, 2008; Yuheng e Hao, 2017), algo que não é tão perceptível em imagens reais, considerando o fato de que os dispositivos de captura costumeiramente realizam a suavização e o borrimento das bordas.

Para as questões de detecção dessas bordas são utilizados cálculos de derivadas, assim como a representação por meio de matrizes de filtros se fazem comuns (Pedrini e Schwartz, 2008; Muthukrishnan e Radha, 2011), citando que um dos mais conhecidos filtros de primeira ordem é o filtro Sobel e para segunda ordem, cita-se o filtro laplaciano.

O operador Sobel, em suma, pode ser introduzido ao assimilar com uma operação de média local, em que este pode calcular bordas horizontais e verticais ou realizar a combinação de ambas. Tendo que diferenças horizontais entre os pontos são utilizadas para detecção de bordas verticais e que diferenças verticais entre os pontos são usadas para detecção de bordas horizontais (Pedrini e Schwartz, 2008).

Essas operações de Sobel podem ser feitas com o auxílio de convoluções e das seguintes matrizes:

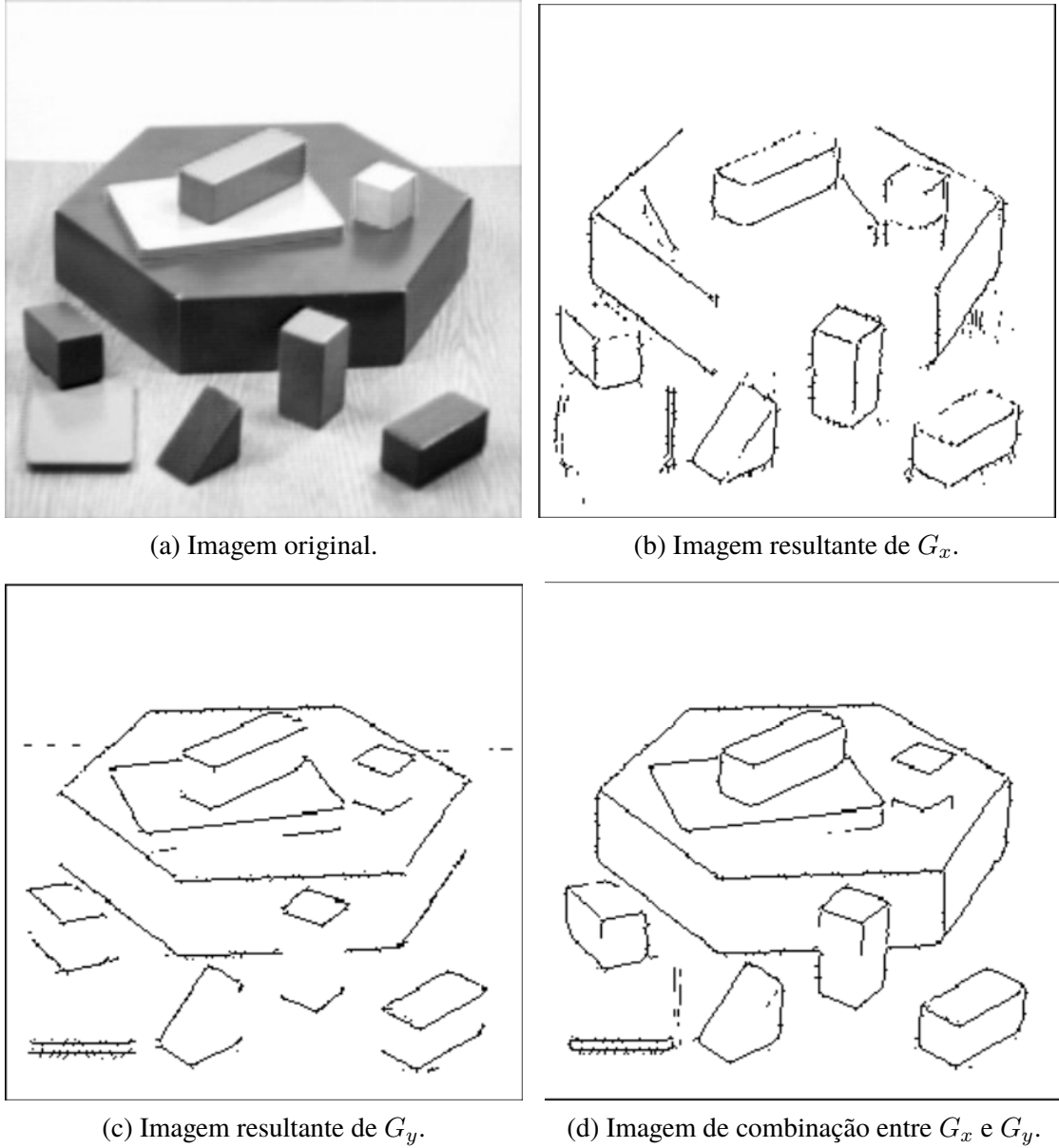
$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (4.5)$$

e

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \quad (4.6)$$

sendo G_x uma matriz para o encontro de bordas verticais e G_y para o encontro de bordas horizontais (Pedrini e Schwartz, 2008), tendo que sua combinação pode resultar em imagens como a da Figura 4.5.

Figura 4.5: Segmentação com operador Sobel.



Fonte: [Pedrini e Schwartz \(2008\)](#).

Já em relação ao operador laplaciano, destaca-se que este faz uso de uma derivada segunda que pode ser representada a partir da Equação 4.7:

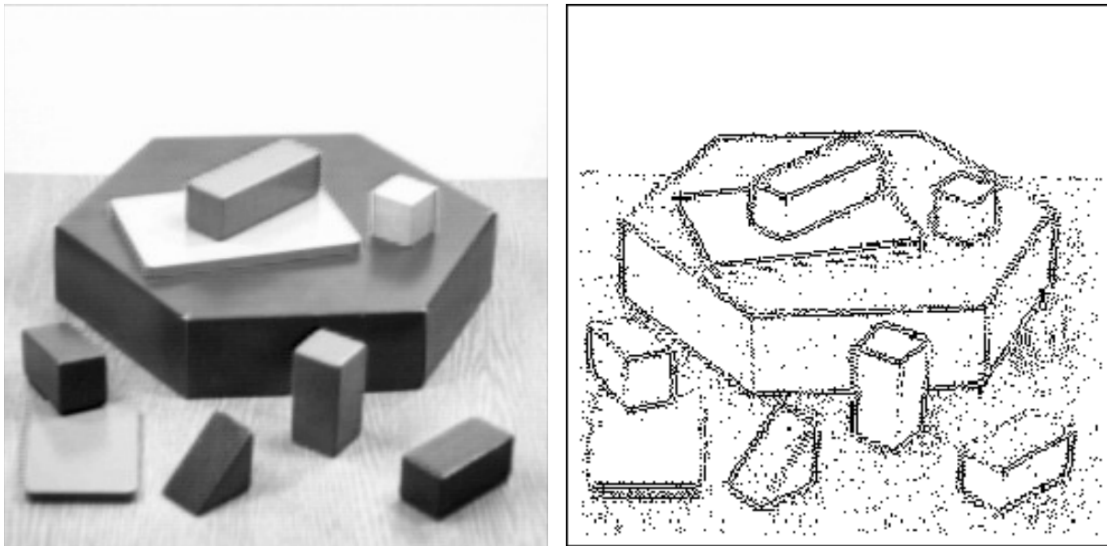
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}, \quad (4.7)$$

tendo que $f(x, y)$ é uma função bidimensional contínua e sendo recomendada para situações em que a imagem não tenha ruído e a posição da borda sejam mais relevantes ([Yuheng e Hao, 2017](#)). Isso se dá, visto que por contar com uma derivada de segunda ordem, torna-se muito sensível aos ruídos ([Pedrini e Schwartz, 2008](#)).

Ainda sobre o operador laplaciano, pode-se dizer que há uma necessidade de que o coeficiente do *pixel* central resulte em um valor positivo e que os seus *pixels* externos sejam negativos (Pedrini e Schwartz, 2008). Assim, considerando uma matriz de vizinhança-8, pode-se exemplificar uma implementação do operador Laplaciano que considera as derivadas segundas em linhas e colunas tendo por base a matriz H_1 representada na Equação 4.8, resultando em aplicações como as visualizadas na Figura 4.6, sendo:

$$H_1 = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \quad (4.8)$$

Figura 4.6: Segmentação com operador Laplaciano.



(a) Imagem original.

(b) Mapa de bordas resultante.

Fonte: Pedrini e Schwartz (2008).

4.1.3 Segmentação Baseada em Grafo

A segmentação baseada em grafo é uma técnica robusta e versátil utilizada para dividir uma imagem em segmentos ou regiões. Esta abordagem consiste em modelar uma imagem como um grafo, onde cada nó representa um *pixel* (ou um *superpixel*, em certas implementações) e as arestas representam a relação entre os *pixels* vizinhos. Tipicamente, essa relação é baseada na similaridade entre os nós, ou seja, *pixels* com características visuais semelhantes, tais como intensidade, cor e textura, tendem a ser conectados por uma aresta (Felzenszwalb e Huttenlocher, 2004).

A segmentação então envolve a subdivisão do grafo em subgrafos (ou regiões na imagem) que são internamente homogêneos e externamente heterogêneos. Existem diversos algoritmos para realizar essa tarefa, muitos deles baseados em técnicas de otimização de grafo, tais como o corte mínimo (*min-cut*) e a expansão máxima de fluxo (*max-flow*) (Boykov et al., 2001).

Um algoritmo amplamente reconhecido na segmentação baseada em grafos é o algoritmo de Felzenszwalb e Huttenlocher (Felzenszwalb e Huttenlocher, 2004), que integra-se com algoritmos de *superpixels*. Este método aborda a segmentação como um problema de agrupamento hierárquico e emprega o critério de dissimilaridade, conhecido como "Evidência de Borda" (em inglês, *Evidence for a Boundary*), para determinar a fusão ou separação de dois nós (ou segmentos). Matematicamente, esta condição é expressa pela Equação 4.9:

$$D(C_i, C_j) = \frac{|B_{ij}|}{\min(|C_i|, |C_j|)} \leq k, \quad (4.9)$$

onde $D(C_i, C_j)$ é a dissimilaridade entre os segmentos C_i e C_j , $|B_{ij}|$ representa a diferença de intensidade entre as bordas dos segmentos C_i e C_j , $|C_i|$ e $|C_j|$ são os tamanhos dos segmentos C_i e C_j , respectivamente, e k é um limiar que determina o equilíbrio entre a evidência de borda e a expansão dos componentes. Esta equação assegura que $D(\cdot)$ seja um predicado que geralmente determina se a borda entre os componentes é significativa o suficiente para separá-los (quando a borda é evidente), ou se os componentes devem ser fundidos (quando a borda não é clara).

Embora o algoritmo produza segmentações de alta qualidade, a seleção adequada dos parâmetros pode representar um desafio.

O trabalho Falcao et al. (2004) traz uma inovação com a Transformada de Floresta de Imagens (IFT), uma metodologia que modela a imagem como um grafo, abrindo caminho para operações de processamento de imagens baseadas na identificação de caminhos de custo mínimo. Este procedimento, proposto por Falcao et al. (2004), permite uma segmentação eficaz ao conectar cada *pixel* a um caminho mínimo, derivado de várias sementes, destacando-se pela sua adaptabilidade e eficiência. A IFT apoia a versatilidade da segmentação baseada em grafo ao simplificar a análise de imagens, fornecendo uma abordagem unificada para a segmentação e análise de contorno.

4.1.4 Segmentação *Watershed*

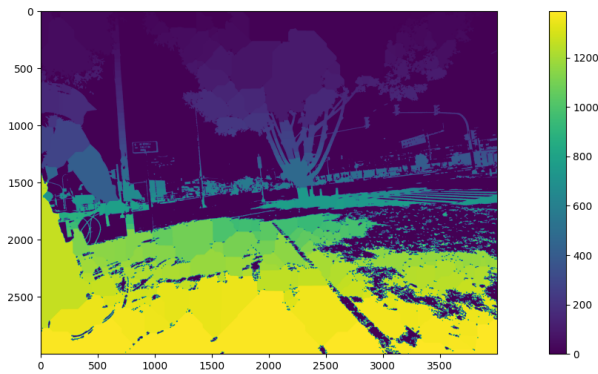
O algoritmo *watershed* (ou bacia hidrográfica, em português) é uma técnica amplamente utilizada para segmentação de imagens. Seu nome se baseia na divisão das superfícies topográ-

ficas (Pedrini e Schwartz, 2008), onde os valores dos *pixels* formam determinadas “montanhas” e uma variação de duas intensidades altas forma um “vale”. O algoritmo *watershed* tem como desafio encontrar um limiar para preencher esses vales a partir dos mínimos regionais, formando bacias de retenção e destacando algumas “montanhas” que consistem em segmentos de linhas de contorno dos objetos da imagem (Pedrini e Schwartz, 2008; Seal et al., 2015). É possível visualizar a imagem original e a imagem segmentada por meio do algoritmo *watershed* a partir da Figura 4.7.

Figura 4.7: Exemplos de segmentação com a aplicação de *watershed*.



(a) Imagem original.



(b) Imagem segmentada por *watershed* com níveis do valor dos *pixels* à direita.

Fonte: retirado e adaptado de Neuhold et al. (2017).

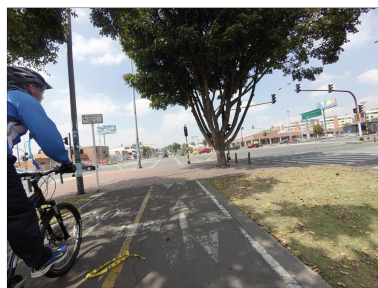
Em resumo esse algoritmo pode ser realizado a partir do seguinte passo-a-passo:

1. Transformação da imagem a ser segmentada em uma imagem de gradiente;
2. Definição dos mínimos locais da imagem de gradiente como marcadores iniciais;

3. Marcação dos *pixels* vizinhos aos marcadores como pertencentes às suas respectivas bacias de retenção;
4. Iterativamente, propagar as bacias de retenção até que elas se fundam em um “nível de água” comum;
5. Extração das linhas de contorno das bacias de retenção para formar os objetos segmentados.

Além disso, por meio da Figura 4.8 é possível observar alguns resultados do processo de obtenção do resultado final, que é a imagem segmentada com a técnica *watershed*.

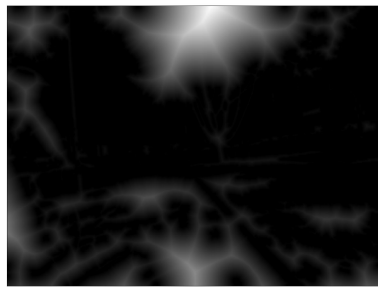
Figura 4.8: Processo de segmentação *watershed*.



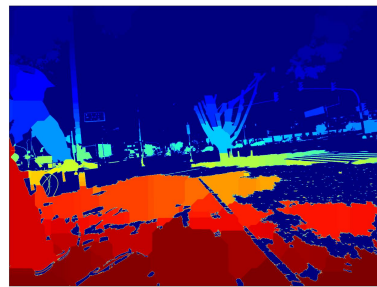
(a) Imagem original.



(b) Limiarização.



(c) Transformação das distâncias.



(d) Objetos segmentados.

Fonte: retirado e adaptado de Neuhold et al. (2017).

A deficiência desse algoritmo se dá pela sua sensibilidade em relação aos ruídos que são expressados por valores máximos ou mínimos na imagem, gerando assim, o aumento de regiões de segmentação (Pedrini e Schwartz, 2008). Além disso, a descontinuidade das bordas de algumas imagens pode ocasionar em que algumas bacias cubram determinados contornos e impedindo a segmentação dos mesmos.

4.1.5 Segmentação *Livewire*

A segmentação *livewire*, também conhecida como segmentação interativa ou segmentação assistida pelo usuário, é uma técnica na qual o usuário define um ponto inicial e um ponto final na imagem. A partir desses pontos, o processo da segmentação realiza automaticamente a identificação dos caminhos de menor custo entre o ponto inicial e final. Este custo geralmente é definido com base em características de borda, tal como o gradiente de intensidade, a direção de borda, e entre outros (Falcão, 1997).

O método *livewire*, proposto por Barrett e Mortensen (1997), é um dos mais conhecidos para a segmentação interativa. A técnica emprega uma estrutura de grafo para representar a imagem, na qual cada *pixel* é considerado como um nó e as arestas são definidas pelas distâncias euclidianas entre os *pixels* vizinhos. O custo de cada aresta é determinado com base nas características da borda, onde:

- $f_{gradiente}(i, j)$ é uma função de custo baseada no gradiente de intensidade dos *pixels* em i e j ,
- $f_{direcao}(i, j)$ é uma função de custo baseada na direção de borda dos *pixels* em i e j ,
- $f_{tamanho}(i, j)$ é uma função de custo baseada na distância euclidiana (ou comprimento) dos *pixels* em i e j .

Essas características são combinadas para formar o custo total de uma aresta na estrutura de grafo, como expresso matematicamente pela Equação 4.10:

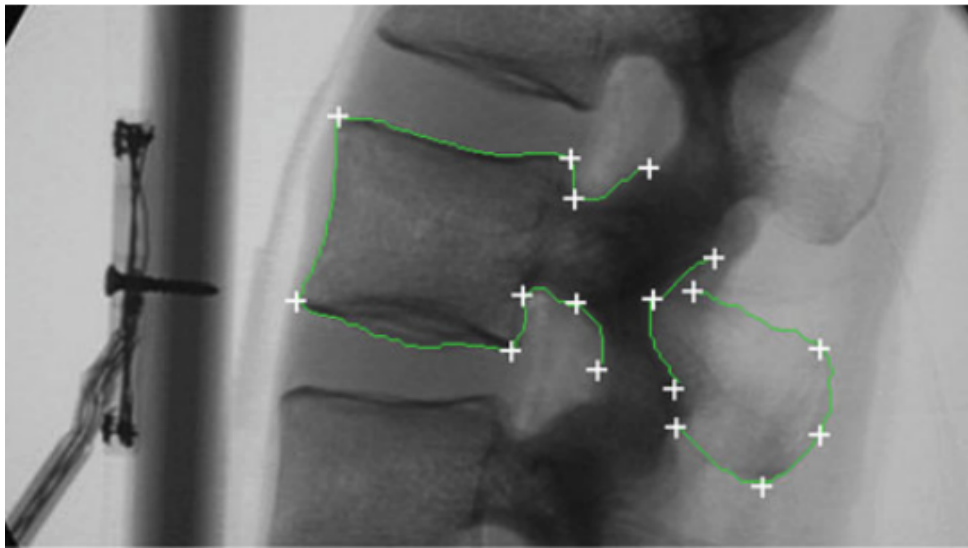
$$c_{(i,j)} = f_{gradiente}(i, j) + f_{direcao}(i, j) + f_{tamanho}(i, j), \quad (4.10)$$

onde $c_{(i,j)}$ é o custo total de uma aresta entre os nós i e j .

Essas funções de custo garantem que as bordas bem definidas na imagem realcem-se e que as bordas contínuas e mais longas tenham um custo menor, direcionando o caminho preferencial para estas.

Vale citar que geralmente, a técnica *livewire* é muito eficiente em imagens médicas onde as bordas dos objetos de interesse são claras e consistentes. Contudo, ela pode ter desempenho insatisfatório em imagens com ruído, baixo contraste ou descontinuidade nas bordas (Falcão, 1997).

Figura 4.9: Exemplo da segmentação *livewire* para extração dos contornos da imagem.



Fonte: retirado e adaptado de [Zheng et al. \(2011\)](#).

Como ilustrado na Figura 4.9, o usuário pode iterativamente definir vários pontos iniciais e finais, e a técnica *livewire* irá automaticamente conectar esses pontos formando a borda do objeto de interesse. Assim, a segmentação *livewire* é um método semi-automático que combina a intuição do usuário com a eficiência da computação, fornecendo resultados de segmentação bastante precisos.

4.1.6 Segmentação *Superpixels*

A noção de *superpixels* foi introduzida por [Ren e Malik \(2003\)](#) como uma forma de agrupar *pixels* que compartilham características comuns, como cor, textura, e localização espacial, para formar uma unidade maior e mais coesa chamada de *superpixel*. Essa abordagem de segmentação de imagens é frequentemente utilizada como um estágio preliminar para tarefas de compreensão de imagens e análise de cenas, pois pode reduzir significativamente a complexidade dos dados ao diminuir o número de unidades de processamento ([Achanta et al., 2012](#)).

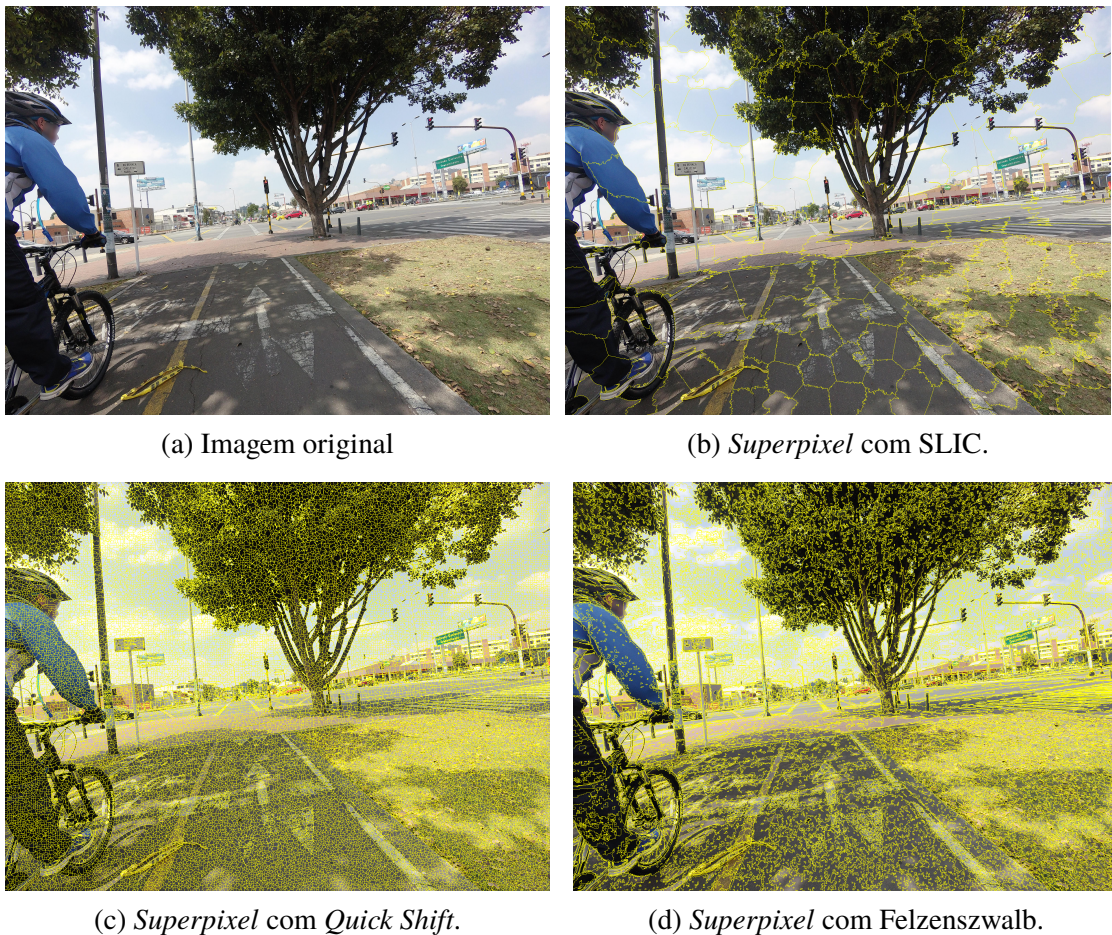
Diversos algoritmos foram propostos para a geração de *superpixels*, tais como: *Simple Linear Iterative Clustering* (SLIC) ([Achanta et al., 2012](#)), *Quick Shift* ([Vedaldi e Soatto, 2008](#)), *Felzenszwalb Efficient Graph-Based Image Segmentation* ([Felzenszwalb e Huttenlocher, 2004](#)) e outros.

O algoritmo SLIC tem sido largamente utilizado devido à sua eficácia e eficiência. Esse método baseia-se no algoritmo K-means (que será discutido na Seção 4.1.7) para clusterização de *pixels*, mas, ao contrário do K-means, leva em conta tanto a cor (ou intensidade para imagens em

escala de cinza) quanto a proximidade espacial dos *pixels* (Achanta et al., 2012). Os *superpixels* gerados com o algoritmo SLIC são compactos e pode-se controlar a quantidade desejada de *superpixels*.

A Figura 4.10 ilustra a segmentação de *superpixels* utilizando o algoritmo SLIC, *Quick Shift* e Felzenszwalb em uma imagem. Cada segmento de *superpixel* distinto na Figura 4.10b representa uma separação de cor diferente. Como pode ser observado, os *superpixels* aderem bem às bordas dos objetos na cena, o que os torna uma unidade de processamento útil para várias tarefas de análise de imagem no campo de visão computacional. Além disso, reduzir a imagem aos *superpixels* em alguns casos, pode permitir diminuir o tempo de processamento para regiões de interesse, sem perder muita informação relevante.

Figura 4.10: Exemplo da segmentação com *superpixels*.



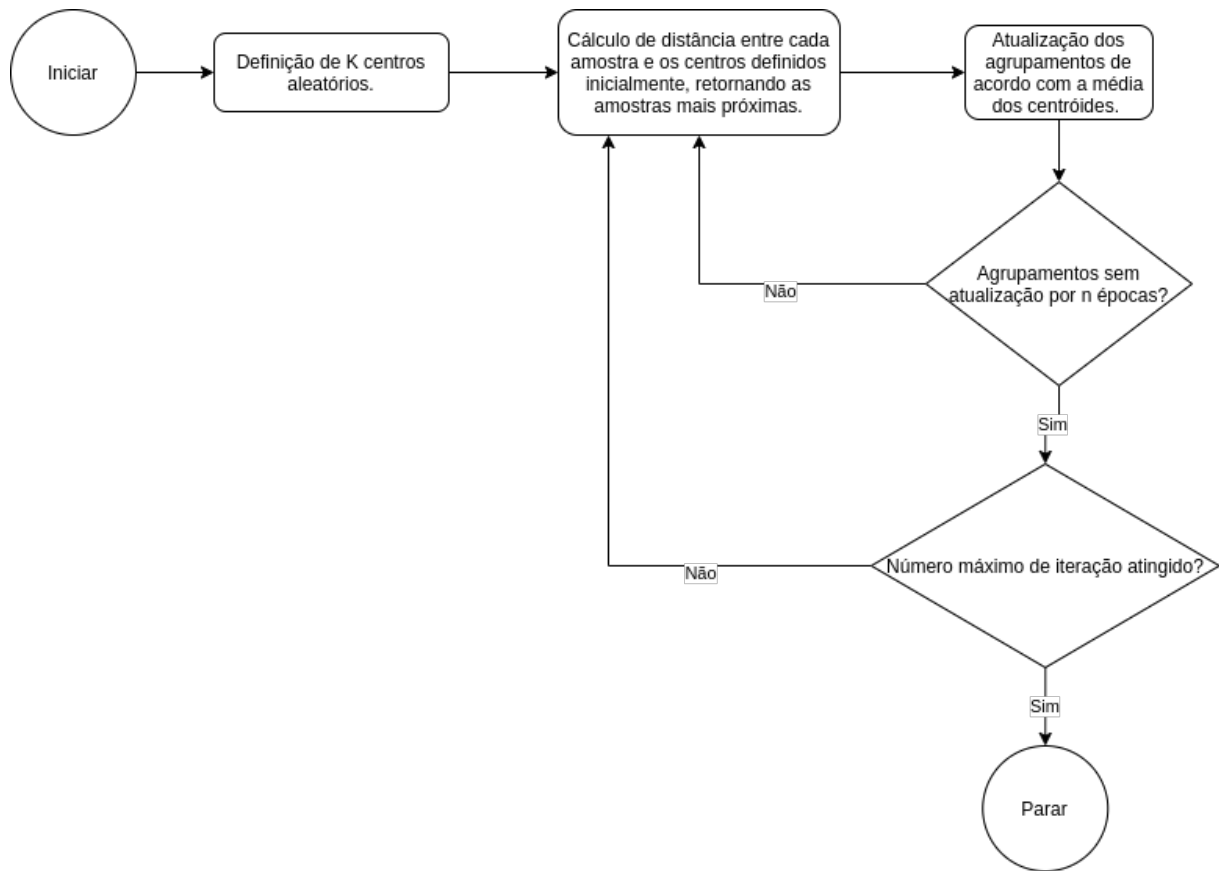
Fonte: retirado e adaptado de Neuhold et al. (2017).

4.1.7 Segmentação Baseada em Agrupamentos

Como o nome propõe, as segmentações baseadas em agrupamentos na esfera de visão computacional e processamento de imagens são compostas pela agregação de *pixels* que possuem características similares em determinado espaço de características (Yuheng e Hao, 2017). No entanto, a definição de agrupamento (ou *clustering*) ainda não é concreta (Yuheng e Hao, 2017), haja vista que não é categorizada como segmentação por alguns autores, pois os agrupamentos são realizados no espaço de medidas, enquanto a segmentação ocorre no domínio espacial da imagem (Haralick e Shapiro, 1985).

Posto isso, destaca-se o uso do K-means (MacQueen, 1967; Hans-Hermann, 2008), o qual é um método não supervisionado que realiza agrupamento de dados de um *dataset* de acordo com as suas similaridades e em quantidades desejadas (k). Ou seja, o algoritmo gera um ponto inicial randômico como centroide para um determinado *cluster* e depois tenta atribuir os demais dados do *dataset* aos *clusters* de acordo com a sua proximidade. Para isso, o algoritmo usa a geometria (ou distância) euclidiana (Mahmud et al., 2012). Vale ressaltar que os pontos centroides são atualizados conforme as classificações que são feitas no algoritmo, e os mesmos param de atualizar quando o algoritmo cumpre seu objetivo de convergência, há poucas interações dos dados com diferentes *clusters* ou através de um número fixo de iterações (Dunham, 2006), conforme representado no fluxograma da Figura 4.11, tendo que $n \in \mathbb{Z}$.

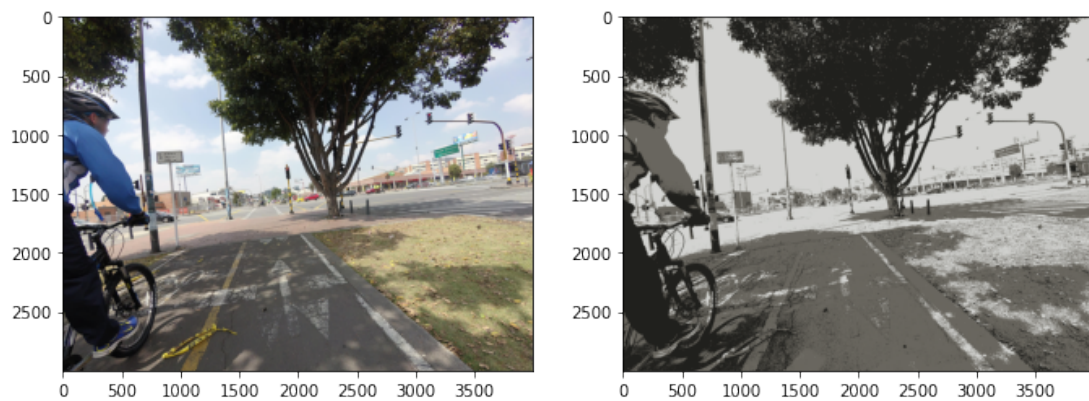
Figura 4.11: Fluxograma do algoritmo K-means.



Fonte: do próprio autor.

Na Figura 4.12, é possível observar resultado do uso do algoritmo K-means para agrupar as regiões da imagem, tendo que foi definido um valor de $k = 3$.

Figura 4.12: Segmentação com K-means.



(a) Imagem original.

(b) Segmentação com $k = 3$.

Fonte: retirado e adaptado de Neuhold et al. (2017).

4.1.8 Segmentação com Redes Neurais

A segmentação de imagens por meio de redes neurais (Capítulo 2) ou redes convolucionais (Capítulo 3), é frequentemente associada à área de detecção de objetos devido à capacidade de demarcação e ao uso de caixas delimitadoras (em inglês, *bounding boxes*) (Ghosh et al., 2019). No entanto, é importante salientar que, para as segmentações de imagens, as arquiteturas CNNs são amplamente utilizadas (Gonçalves et al., 2021). Essas redes conseguem extrair as características mais relevantes de uma imagem e organizá-las, sendo eficazes não apenas em tarefas de detecção, localização e rastreamento de objetos, mas também na segmentação de imagens (Ghosh et al., 2019).

Um método mais recente que aborda essa tarefa e segue essa linha de raciocínio será detalhado no Capítulo 5, que trata das segmentações semânticas.

4.2 Considerações Finais do Capítulo

Na Tabela 1, é possível visualizar as técnicas comentadas no decorrer do capítulo, assim como uma breve descrição de seus usos, suas vantagens e desvantagens.

Tabela 1: Comparação entre os métodos de segmentações tradicionais.

Técnica	Descrição	Vantagens	Desvantagens
Segmentação Baseada em Regiões	Concentra-se em encontrar e agrupar valores semelhantes sendo separados por um limiar.	- Possui baixa complexidade e alto desempenho; - Não requer pré-processamento; - Recomendado para imagens com alto contraste entre fundo e objeto de interesse.	- Omissão de detalhes quando não há um contraste significativo.
Segmentação por Bordas	Realizado com base na detecção de descontinuidade, detectando bordas e definindo um limite do objeto.	- Recomendável para imagens com bons contrastes entre os objetos.	- Perda de detalhes para imagens com muitas bordas, ruidosas ou com baixo contraste entre os objetos.
Segmentação Baseada em Agrupamentos	Divide os <i>pixels</i> da imagem em <i>k clusters</i> homogêneos e exclusivos obtendo, assim, as segmentações.	- Conceitos já estabelecidos; - Recomendado para aplicações em tempo real; - Bom desempenho para pequenos <i>datasets</i>; - Não precisa de tempo de treinamento.	- Pode demandar muito tempo para encontrar a minimização da função de custo; - Não é adequado para agrupar <i>clusters</i> não convexos.
Segmentação Baseada em Grafos	Representa uma imagem como um grafo, onde cada nó é um <i>pixel</i> e as arestas são a relação entre os <i>pixels</i> vizinhos.	- Robusto e preciso; - Flexível com variados tipos de imagens; - Compatibilidade com outros métodos de segmentação.	- Parâmetros podem ser difíceis de determinar; - Sensível aos parâmetros; - Podem ocorrer segmentações em excesso; especialmente em imagens ruidosas.
Segmentação <i>Supapixel</i>	Agrupar <i>pixels</i> que compartilham características comuns para formar uma maior unidade coesa denominada <i>superpixel</i> .	- Reduz a complexidade dos dados; - Adere bem às bordas do objeto; - Maior velocidade de processamento.	- Pode perder detalhes finos das imagens; - Parâmetros podem ser difíceis de determinar.
Segmentação <i>Watershed</i>	Trata a imagem como um relevo topográfico, inundando todas as depressões a partir dos mínimos locais.	- Recomendável para imagens com bordas claras e bem definidas; - Baixo custo computacional.	- Sensível a ruídos; - Pode produzir segmentos excessivos.
Segmentação <i>Livewire</i>	Abordagem interativa que elege um caminho de menor custo entre dois pontos escolhidos pelo usuário.	- Proporciona resultados de segmentação precisos; - Aproveita a intuição do usuário; - Eficiência em imagens médicas com bordas claras e consistentes.	- Desempenho insatisfatório em imagens com ruído, baixo contraste ou descontinuidade nas bordas.
Segmentação com Redes Neurais	Normalmente relacionada a algoritmos de aprendizado profundo e CNNs.	- Conceito já estabelecido; - Abordagem simples e flexível.	- O treinamento do modelo pode consumir muito tempo e recursos.

Fonte: do próprio autor.

Na Seção 4.1, abordamos os algoritmos comumente utilizados para segmentações, indo desde técnicas mais simples, como o método de *threshold segmentation* (Seção 4.1.1), até abordagens de maior complexidade baseadas em redes neurais artificiais (Seção 4.1.8). Contudo, como discutido por Ghosh et al. (2019) e Minaee et al. (2021a), atualmente existem abordagens de segmentação que empregam *deep learning* e apresentam excelentes desempenhos, sendo capazes de realizar segmentações de imagens e áreas de interesse em nível de *pixel*. Uma dessas abordagens será explorada no próximo capítulo.

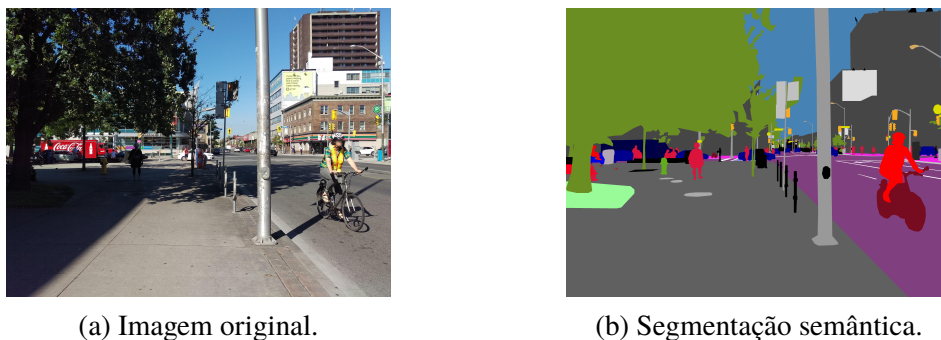
5 SEGMENTAÇÃO SEMÂNTICA BASEADA EM APRENDIZADO PROFUNDO

No contexto das segmentações baseadas em técnicas de *deep learning* com uma vertente mais específica das CNNs, a segmentação semântica se destaca por atribuir uma classe ou rótulo específico a cada pixel em uma cena (Wang et al., 2017; Ghosh et al., 2019; Long et al., 2015; Arbelaez et al., 2012; Zhang et al., 2018), constituindo, assim, um trabalho em nível de pixel.

O emprego de técnicas de segmentação semântica tem demonstrado utilidade em cenários onde o contexto é desconhecido, permitindo compreender a categoria dos objetos presentes na cena, além de suas propriedades de localização, espacialidade, dimensionalidade e formatos (Zhang et al., 2018). Essas abordagens são de extrema importância em áreas como imagens médicas, sistemas autônomos e campos correlatos.

Normalmente, as saídas das redes de segmentação semântica consistem em máscaras de *pixels* organizadas como mapas de características, mapeando todas as classes na imagem original. Um exemplo visual dessa saída pode ser observado especificamente na Figura 5.1b, tendo a Figura 5.1a como referência da imagem original.

Figura 5.1: Segmentação semântica.



Fonte: Neuhold et al. (2017).

Portanto, neste capítulo, serão abordados tópicos relacionados às técnicas estado-da-arte no âmbito das arquiteturas de segmentação semântica, conforme detalhado na Seção 5.1. Três variantes proeminentes serão discutidas nessa seção, buscando oferecer uma visão aprofundada dessas estratégias. Ademais, as métricas frequentemente utilizadas para avaliar a qualidade das segmentações semânticas serão minuciosamente detalhadas na Seção 5.2, fornecendo uma compreensão abrangente dos critérios de avaliação relevantes nesse tipo de abordagem.

5.1 Arquiteturas

Quando abordamos as arquiteturas para os modelos de segmentação semântica, é evidente o crescente desenvolvimento de oportunidades para resolver a problemática de separar classes de objetos a nível de *pixel*, como mencionado por [Guo et al. \(2018\)](#). No contexto das arquiteturas de segmentação semântica, existe uma divisão relevante relacionada ao nível de supervisão da rede, que inclui abordagens totalmente supervisionadas, fracamente supervisionadas e semi-supervisionadas, conforme detalhado por [Hao et al. \(2020\)](#). No entanto, este trabalho se concentra exclusivamente nas arquiteturas consideradas estado-da-arte na parte totalmente supervisionada, pressupondo a disponibilidade de dados de treinamento rotulados, incluindo imagens originais juntamente com suas respectivas máscaras anotadas ([Hao et al., 2020](#)).

Para esmiuçar mais sobre essas arquiteturas, é importante ressaltar a diferença em relação às CNNs convencionais, que tendem a perder detalhes espaciais devido aos processos de *pooling* e convoluções. As redes de segmentação semântica, ao contrário, são mais atentas a esses detalhes. Nesse contexto, a colaboração adequada entre esses dois tipos de características possui o potencial de aprimorar significativamente o desempenho da segmentação semântica, seguindo a estratégia denominada de “aprimoramento de recursos”, a qual será explorada adiante.

5.1.1 Fully Convolutional Networks (FCN)

Quando aborda-se o tema das segmentações semânticas, vale dizer que arquiteturas e *frameworks* foram desenvolvidos visando o avanço em termos de performance e capacidade. No entanto, o primeiro *framework* a alcançar o status de estado-da-arte para esta tarefa de segmentação baseia-se na arquitetura Fully Convolutional Networks (FCN) ([Long et al., 2015](#)). Contrariamente às redes tradicionais que possuem camadas densas no final, a FCN é totalmente composta por camadas convolucionais, até mesmo na sua última camada ([Hesamian et al., 2019](#)).

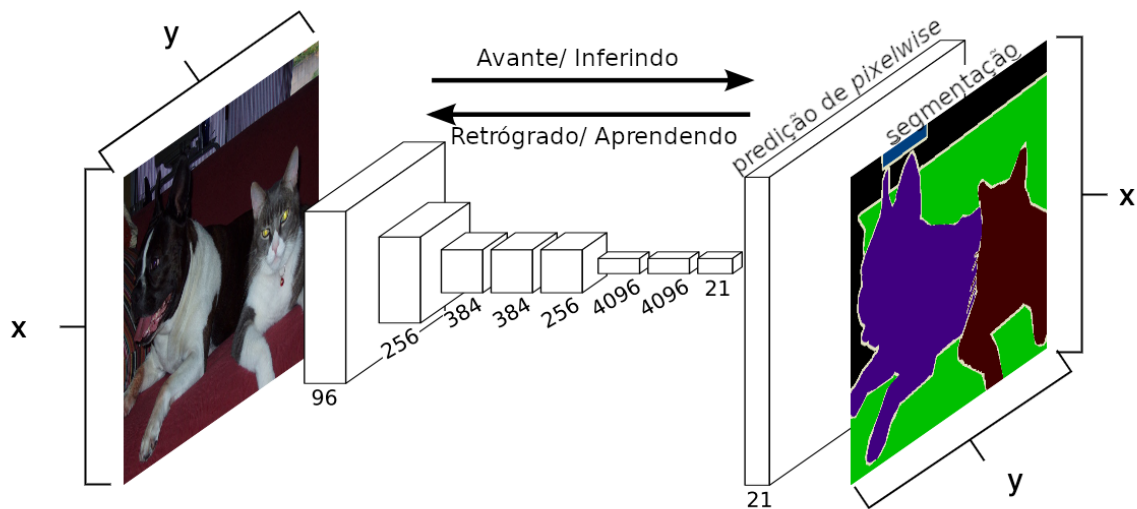
Esta característica permite à FCN realizar uma análise *pixelwise*, onde cada *pixel* da imagem é subamostrado. Em seguida, através de processos de convolução e *upsampling*, um mapa de segmentação semântica é construído. Este mapa de segmentação possui dimensões idênticas à imagem de entrada, o que permite a atribuição de uma classe semântica para cada *pixel* ([Minaee et al., 2021a](#); [Zhang et al., 2018](#); [Hesamian et al., 2019](#)).

A aplicação do FCN nos *datasets* PASCAL VOC 2011 ([Everingham et al., 2010](#)), SIFT Flow ([Liu et al., 2011](#)) e NYUDv2 ([Nathan Silberman Derek Hoiem e Fergus, 2012](#)), resultou

numa precisão de *pixel* (PA) de 90,3% e uma mIoU de 62,7% no primeiro *dataset* mencionado (Ghosh et al., 2019). O FCN, portanto, estabeleceu um novo patamar de referência quanto ao estado-da-arte em segmentação semântica (Minaee et al., 2021a) que antes era ocupado pelo modelo DeepLab (Chen et al., 2017).

A arquitetura da FCN é baseada nas arquiteturas das *Convolutional Neural Networks* (CNNs) VGG-16 (explorada na Seção 3.7) e GoogLeNet (Szegedy et al., 2015). Esta adaptação permitiu que a FCN aceitasse entradas de tamanhos variáveis e produzisse saídas da mesma dimensão (Minaee et al., 2021a), como pode ser visualizado na Figura 5.2.

Figura 5.2: Representação de exemplo em FCN.



Fonte: retirada e adaptado de Long et al. (2015).

Essas saídas, que são mapas de características do mesmo tamanho da imagem de entrada, podem ser utilizadas para várias aplicações. Por exemplo, em uma tarefa de detecção de objetos, cada *pixel* da saída será classificado com a etiqueta do objeto à qual pertence, fornecendo portanto uma localização precisa dos objetos na imagem. Em uma aplicação de condução autônoma, a saída pode ser usada para identificar as zonas navegáveis ou obstáculos presentes na cena.

É relevante notar que a eficácia da FCN também está ligada ao seu mecanismo de subamostragem e amostragem ascendente, também conhecido como *upsampling*. As camadas inferiores da rede capturam características locais, como bordas e texturas, enquanto as camadas

superiores conseguem captar características de nível superior, como classes de objetos e suas relações espaciais (Long et al., 2015).

O desafio da subamostragem é que, enquanto ela contribui para a identificação de recursos em várias escalas e aumenta a eficiência computacional, também leva a uma perda de detalhes espaciais devido à redução da resolução da imagem. Para mitigar isso, a FCN implementa operações de *upsampling*, que expandem a saída de baixa resolução para o tamanho original da imagem. Isso é realizado por meio de camadas de deconvolução ou interpolação, permitindo que a FCN produza uma segmentação densa em tempo de execução em uma única passagem para frente pela rede (Long et al., 2015).

Além disso, a FCN integra previsões de várias camadas em vários níveis de resolução, conhecido como saltos de conexões (ou em inglês, *skip connections*), para refinar ainda mais a precisão da segmentação (Minaee et al., 2021a). Com isso, a FCN consegue garantir que a segmentação semântica final esteja em conformidade tanto com as características de nível inferior quanto de alto nível, proporcionando resultados de segmentação precisos e detalhados.

Por fim, quanto às suas limitações, é informado que as configurações de FCN são ineficientes em trabalhos de segmentação 3D e para o uso em tempo real, visto que leva cerca de 100ms - em hardware não especificado pelo autor - para trabalhar com imagens de baixa resolução (Minaee et al., 2021a), mas novas arquiteturas baseadas nesse modelo e na arquitetura ParseNet (Liu et al., 2015) vêm evoluindo para suprir essas limitações.

5.1.2 U-Net

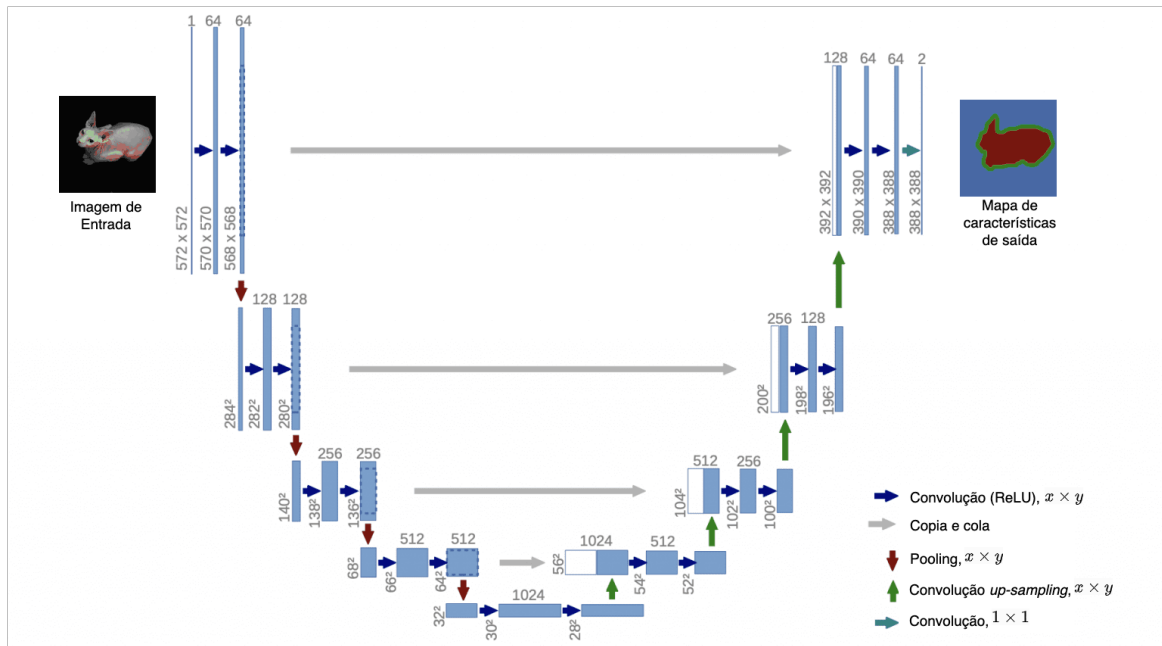
No âmbito das arquiteturas voltadas para segmentação semântica, a evolução trouxe a U-Net (Ronneberger et al., 2015), que se destacou após o período das FCNs. Introduzida em 2015, a U-Net rapidamente ganhou notoriedade, superando a popularidade das mencionadas FCNs (Sultana et al., 2020).

As U-Nets operam com um caminho de contração (também conhecido como *encoder*) que progressivamente reduz a resolução espacial dos dados e captura o contexto dos exemplos, seguido por um caminho de expansão simétrica (também conhecido como *decoder*) que restaura a resolução original por meio de convoluções transpostas e realiza a segmentação. A estrutura se assemelha a um “U”, como retrata a Figura 5.3, o que levou à sugestão do nome da arquitetura. A conexão entre as camadas de *encoder* e *decoder*, conhecida como “*skip connection*”, permite a preservação de informações de baixo nível e ajuda a mitigar problemas de perda de detalhes

durante a redução da resolução (Minaee et al., 2021a,b). Por não conter camadas densas nas U-Nets, elas podem ser consideradas modelos baseados em FCNs (Minaee et al., 2021a), sendo que a maior diferença entre essas arquiteturas está no uso de “*skip connections*” entre as convoluções e *upsamplings*, além da simetria das U-nets (Ozturk et al., 2020).

Vale citar que para a implementação de “up-samplings”, originalmente nas U-nets, foram utilizadas as convoluções transpostas com duas dimensões, que também são conhecidas como deconvoluções ou camada de deconvolução, realizando o processo contrário às camadas de *pooling* (Zeiler e Fergus, 2014). Quando observa-se a parte técnica, essa camada é denominada *Conv2DTranspose*, sendo que seu maior diferencial é que ela amplia a resolução espacial dos dados, em oposição à redução feita pelas camadas de *pooling*. No entanto, ao contrário das camadas de *pooling*, a *Conv2DTranspose* faz uso de uma janela de convolução para aumentar as dimensões dos dados. Essa camada é crucial para a U-Net, pois desempenha um papel fundamental na reconstrução das características espaciais, restaurando a resolução original e contribuindo para a precisão da segmentação, ao mesmo tempo em que mantém as conexões entre as características de nível mais baixo e mais alto, garantindo a riqueza dos detalhes na segmentação final.

Figura 5.3: Arquitetura U-Net.



Fonte: retirada e adaptado de Ronneberger et al. (2015).

As vantagens inerentes a essa arquitetura incluem uma capacidade robusta para lidar com imagens de tamanhos variados, permitindo a aplicação eficiente em diversos contextos, como na análise de imagens médicas (Ronneberger et al., 2015). Além disso, as U-Nets têm a habilidade de aprender representações altamente significativas em diferentes níveis de abstração, o que as torna eficazes na captura de detalhes importantes em uma imagem (Alom et al., 2019). Também é importante destacar que essas redes têm a capacidade de reduzir a tendência de gerar segmentações excessivamente granulares, fornecendo segmentações mais suaves e coerentes (Du et al., 2020). No entanto, é fundamental reconhecer que, como qualquer abordagem, as U-Nets também possuem desvantagens. Em particular, essas redes frequentemente possuem um grande número de parâmetros, o que pode aumentar os requisitos computacionais e tornar o treinamento mais demorado, dependendo do tamanho do conjunto de dados e da complexidade da tarefa (Ronneberger et al., 2015). Além disso, pela grande quantidade de convoluções, muitas vezes a informação espacial é perdida, segundo Zhang e Niu (2023).

Voltando ao contexto mais amplo, onde a segmentação semântica tem como objetivo atribuir rótulos de classe a cada *pixel* em uma imagem, a arquitetura da U-Net se mostra relevante e eficaz, sendo exemplificado pelos resultados obtidos desde seu desenvolvimento com Ronneberger et al. (2015). No entanto, questões que exploram a preservação da espacialidade da imagem durante a segmentação nesse tipo de estrutura ainda são pouco explorados, principalmente quando a proposta está em variações das tradicionais camadas de *pooling* que são os atuais estado-da-arte.

5.1.3 U-Net-Like

Após o surgimento e o êxito alcançado pelas U-Nets, surgiram variações dessas redes, frequentemente denominadas de U-Net-Like, inspiradas pela estrutura original proposta por Ronneberger et al. (2015) e amplamente documentada em (Kugelman et al., 2022).

Essas redes mantêm a arquitetura de contração e expansão da U-Net original, mas apresentam características distintas. Uma das diferenças significativas é a adição de *skip connections* com a inclusão de uma camada extra, a incorporação de técnicas como *Batch Normalization* proposta por Ioffe e Szegedy (2015) e a função de ativação ReLU, discutida na Seção 2.1.4, nas camadas de contração. Essas alterações têm como objetivo manter as médias e variâncias das entradas constantes, acelerando o aprendizado e aumentando a estabilidade durante a fase de treinamento (Pfister et al., 2019).

Outra modificação presente nas U-Net-Like diz respeito à forma como as convoluções são aplicadas. Enquanto [Ronneberger et al. \(2015\)](#) não aplicava preenchimento (*padding*) devido ao processo de divisão de imagens maiores em blocos menores durante o processamento, a abordagem das U-Net-Like introduz *pixels* adicionais ao redor das bordas da imagem. Esse procedimento permite que as convoluções utilizem *pixels* vizinhos, aprimorando a eficiência e a qualidade da segmentação ([Pfister et al., 2019](#)).

Dentre os últimos detalhes, vale dizer que diferente do método de *up-sampling* com as convoluções transpostas em duas dimensões, esse método utiliza de um método chamado originalmente de *Upsampling2D*, o qual emprega uma abordagem diferente para aumentar a resolução espacial dos dados. Enquanto as convoluções transpostas realizam a ampliação por meio do aprendizado de pesos, o método *Upsampling2D* opera replicando os valores existentes nos mapas de características, efetivamente duplicando-os e, assim, aumentando a resolução espacial. Além disso, sua maior mudança em relação às convoluções transpostas em duas dimensões se dá pela ausência de parâmetros treináveis, resultando em uma operação mais simples e menos custosa computacionalmente.

Finalmente, vale dizer que no uso das U-Net-Like é possível visualizar uma diminuição dos parâmetros de treinamento da rede ([Pfister et al., 2019](#)), contribuindo para uma arquitetura mais enxuta e eficiente. Além disso, o uso do *Upsampling2D* nas U-Net-Like implica não só em uma redução de parâmetros, mas também em um processo de treinamento mais rápido e menos complexo, uma vez que essa abordagem não requer parâmetros treináveis adicionais, proporcionando uma operação computacionalmente mais leve e uma convergência mais ágil durante o treinamento.

5.2 Métricas

Considerando que o objetivo principal das segmentações semânticas está relacionado à atribuição de classes para cada um dos *pixels* presentes na imagem ([Csurka et al., 2013](#)), é fundamental o uso de métricas para avaliar o desempenho positivo ou negativo dos modelos.

Nesta seção, serão discutidas métricas comuns para avaliação de segmentações semânticas: métricas baseadas em região, como a precisão de *pixel* (Seção 5.2.1), a interseção sobre a união (Seção 5.2.2) e sua variação, a média da interseção sobre a união (Seção 5.2.3), e por fim, o escore F1 (ou F1 *score*) (Seção 5.2.4).

5.2.1 Pixel Accuracy

A métrica de *pixel accuracy* (PA) é uma das métricas mais simples utilizadas no contexto de segmentação semântica, em que, na prática, se resume na razão entre o número de *pixels* classificados corretamente em relação ao total de *pixels* na imagem (Minaee et al., 2021a). Sendo assim, é possível expressar a sua fórmula pela Equação 5.1 (Minaee et al., 2021a):

$$PA = \frac{\sum_{i=0}^K p_{ij}}{\sum_{i=0}^K \sum_{j=0}^K p_{ij}}, \quad (5.1)$$

sendo K o número de classes presentes na imagem, independente se para um plano de fundo ou para primeiro plano, e p_{ij} é o número de *pixels* preditos como i pertencentes à classe j .

Todavia, é relevante citar que essa métrica indicando apenas precisão dos *pixels*, como o nome sugere, não é adequada para situações que há um desequilíbrio de classes, situação em que determinadas classes dominam a cena e que se faz presente no mundo real.

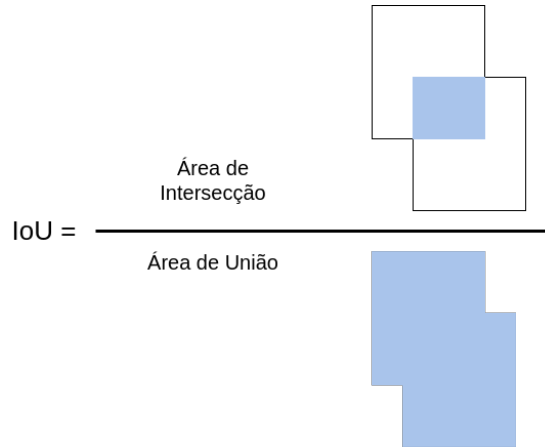
5.2.2 Intersection over Union (IoU)

Já a métrica de intersecção entre a união, do inglês *Intersection over Union* (IoU) ou Jaccard Index, é uma métrica que não passa por problemas de desbalanceamento de classes, além de ser considerada uma das métricas mais comuns para esse tipo de atividade (Minaee et al., 2021a).

Sua representação pode ser visualizada a partir da Figura 5.4, além de que pela Equação 5.2 é possível entender que o resultado de *IoU* se dá pela razão entre a intersecção das áreas da segmentação predita (A) e *ground truth* (B), e a união entre a segmentação predita e o *ground truth*, ou seja,

$$IoU = J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \quad (5.2)$$

Figura 5.4: Representação de IoU.



Fonte: do próprio autor.

5.2.3 Mean Intersection Over Union (MIoU)

Após a introdução da métrica IoU, outras variações ganharam destaque, destacando-se o *Mean Intersection-over-Union* (mIoU), que calcula a média dos valores de IoU para todas as classes presentes na cena, fornecendo um valor escalar como resultado para a segmentação como um todo (Minaee et al., 2021a). Esta métrica tem sido empregada em estudos relevantes, como exemplificado por Mohan e Valada (2020).

É relevante ressaltar que métricas como mIoU e *Mean Average Precision* (mAP), em relação à métrica *Average Precision* - também comentadas por Padilla et al. (2021) - possuem maior relevância na análise de problemas com múltiplas classes ou objetos, permitindo medir a capacidade de generalização do modelo em questão.

A Equação 5.3 pode ser utilizada para ilustrar o funcionamento do *mIoU*, onde N é o número de regiões no conjunto de dados e as demais variáveis são baseadas na Equação 5.2:

$$mIoU = \frac{\sum_{i=1}^N IoU(A_i, B_i)}{N}. \quad (5.3)$$

5.2.4 F1-Score

Para a definição da métrica *F1-score*, que é comum na esfera de segmentação de imagens, é necessário ter ciência de antemão que ela é composta pelo uso de outras duas métricas, sendo elas: 1) precisão, que é a divisão de verdadeiros positivos (VP) pela soma de verdadeiros positivos (VP) e falsos positivos (FP); 2) revocação, que é a divisão de verdadeiros positivos

(VP) pela soma de verdadeiros positivos (VP) e falsos negativos (FN), como mostrados nas Equações 5.4 e 5.5, respectivamente,

$$\text{Precisão} = \frac{VP}{VP + FP} \quad (5.4)$$

e

$$\text{Revocação} = \frac{VP}{VP + FN}. \quad (5.5)$$

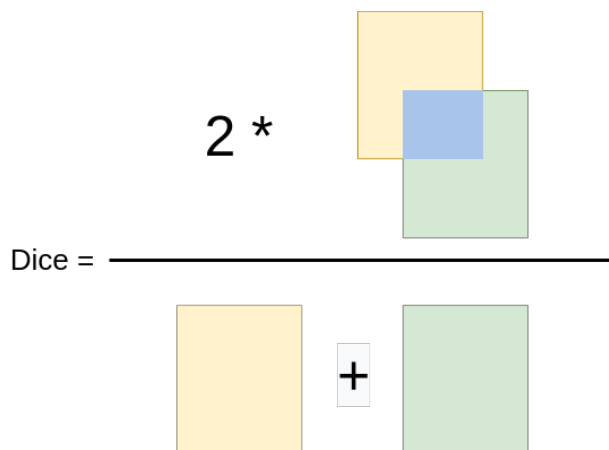
Assim, para a definição de $F1\text{-score}$ é utilizada a média harmônica entre a precisão e revocação (Minaee et al., 2021a), a qual pode ser expressa pela Equação 5.6:

$$F1 - score = \frac{2 \cdot \text{Precisão} \cdot \text{Revocação}}{\text{Precisão} + \text{Revocação}}. \quad (5.6)$$

Por fim, vale comentar que para situações em que tem-se uma segmentação binária. O valor de $F1\text{-score}$ é semelhante ao valor do coeficiente Dice, o qual normalmente é utilizado para situações médicas (Minaee et al., 2021a) e utiliza conceitos de intersecção assim como a métrica de IoU (Seção 5.2.2), visto que quando não está assimilado a situações binárias, pode ser representado pela Equação 5.7 ou pela Figura 5.5:

$$Dice = \frac{2|A \cap B|}{|A| + |B|}. \quad (5.7)$$

Figura 5.5: Representação do coeficiente Dice.



Fonte: do próprio autor.

5.3 Considerações Finais do Capítulo

Em conclusão sobre as segmentações semânticas, é notável o avanço significativo em comparação com tarefas de detecção de objetos, como destacado em [Vaillant \(1994\)](#). Esses novos modelos de segmentação permitem o trabalho com um volume maior de dados em cenas complexas, como abordado neste capítulo.

No entanto, é importante ressaltar que essas redes enfrentam dificuldades quando o problema requer a separação de objetos da mesma classe, mas distintos e sobrepostos, resultando na ausência de uma segmentação individualizada ([Kirillov et al., 2019](#)). Isso fica evidente na Figura 5.6, onde pessoas próximas estão conectadas, ou em situações que exigem uma segmentação detalhada da cena ([Ghosh et al., 2019](#)).

Figura 5.6: Exemplo de segmentação semântica com instâncias unificadas.



(a) Imagem original.



(b) Segmentação com instâncias unificadas.

Fonte: [Fischer et al. \(2017\)](#).

É válido mencionar o surgimento de abordagens estado da arte e arquiteturas baseadas em *Transformers* (Vaswani et al., 2017), que têm demonstrado excelente desempenho em uma variedade de tarefas de visão computacional, incluindo segmentação semântica. No entanto, considerando que o foco deste trabalho estava mais nas questões de *pooling* e nas arquiteturas clássicas de aprendizado profundo, essas abordagens mais novas, embora notáveis, estiveram fora do escopo da discussão atual.

Por fim, é importante ressaltar que os modelos de segmentação semântica podem não ser ideais para sistemas que requerem segmentações em tempo real, especialmente ao operar com câmeras que capturam menos de 25 quadros por segundo. Isso se deve ao seu custo computacional elevado, o qual pode dificultar a obtenção de resultados em tempo hábil para aplicações em tempo real. Essa limitação foi identificada e discutida por Minaee et al. (2021a). Além disso, é fundamental considerar as questões relacionadas ao desempenho das métricas de avaliação do modelo quando se busca preencher essa lacuna de tempo, visando compreender como os modelos se comportam em condições de processamento em tempo real. Essa análise é crucial para determinar a viabilidade e eficácia desses modelos em aplicações práticas que demandam segmentação em tempo real (Badrinarayanan et al., 2017).

6 DESENVOLVIMENTO DO PROJETO

Nos capítulos anteriores, analisamos como as Redes Neurais Convolucionais (CNNs) - Capítulo 3 - provocaram mudanças significativas nas abordagens de segmentação, expandindo as possibilidades de obtenção de resultados mais precisos e melhorando o desempenho dos métodos convencionais de segmentação, conforme discutido nos Capítulos 4 e 5.

Entretanto, ainda existem questões não resolvidas relacionadas às potenciais melhorias que modificações específicas nas componentes destas redes podem proporcionar (Asgari Taghaki et al., 2021). Dentre esses componentes, particular atenção é dada às limitações das camadas de *pooling* tradicionais (Liu et al., 2019; He et al., 2015). Estas camadas têm duas tarefas principais: reduzir a dimensionalidade dos dados, a fim de diminuir a carga computacional do modelo e, simultaneamente, preservar as informações mais relevantes, principalmente as espaciais.

Neste contexto, o objetivo primordial deste trabalho é propor uma nova técnica de *pooling* para CNNs, aplicando testes em CNNs convencionais, mas com ênfase na segmentação semântica com o modelo U-Net. A abordagem proposta visa manter as informações espaciais dos dados de entrada enquanto realiza a necessária redução de dimensionalidade, mas considerando uma visão global do mapa de atributos, algo que não ocorre com os métodos tradicionais de *pooling*. Os resultados obtidos serão posteriormente comparados com aqueles produzidos pelos métodos de *pooling* comumente utilizados.

Este capítulo trata detalhadamente dos materiais e métodos utilizados para a condução e avaliação da pesquisa, discutidos em profundidade na Seção 6.1. Enquanto a aplicação de métricas específicas para a avaliação dos resultados será discutida na Seção 6.2.

Finalmente, vale citar que nesse capítulo será possível encontrar uma apresentação detalhada do novo método de *pooling* proposto e seu impacto na performance das redes de aprendizado profundo.

6.1 Materiais e Métodos

Esta seção busca esclarecer sobre todos os recursos tecnológicos, *hardwares* e abordagens metodológicas que foram empregados para suportar a consecução dos experimentos deste estudo (Seção 6.1.1). Neste contexto, é essencial elucidar acerca das motivações e origens da técnica proposta com a extração de características (Seção 6.1.2), bem como a implementação

da nova técnica de *pooling*, denominada BPCAPooling (Seção 6.1.3), aplicada em estruturas de CNNs. Além disso, a descrição dos conjuntos de dados utilizados fornece um embasamento sólido para compreender a motivação e a importância dos resultados obtidos ao longo das fases de desenvolvimento, treinamento, validação e teste (Seção 6.1.4).

Nesta mesma seção, procura-se detalhar a metodologia e as estratégias adotadas nos experimentos realizados. A abordagem de transferência de aprendizado, em particular, recebe destaque (Seção 6.1.5). Tal técnica envolve a adaptação, ou o chamado *fine-tuning*, de um pré-modelo treinado para operar especificamente na tarefa de interesse, iniciando o processo de aprendizagem a partir de uma configuração pré-definida de pesos, como foi citado na Seção 3.6.

Além disso, é válida a discussão acerca do procedimento de aumento de dados (Seção 6.1.6), adotado como uma medida para minimizar possíveis vieses nos modelos e potencializar sua capacidade de generalização (Shorten e Khoshgoftaar, 2019).

Por fim, na Seção 6.1.7, abordaremos as respectivas modificações realizadas na camada de *pooling* dos modelos, que foram cruciais para a realização e a validação do experimento. O entendimento dessa sistematização se faz necessário para garantir a replicação das etapas pelos futuros pesquisadores da comunidade científica e proporcionar avanços consideráveis na área de visão computacional.

6.1.1 Tecnologias e *Hardware*

Um dos principais recursos tecnológicos utilizados neste trabalho é a linguagem de programação Python (versão 3.9.18) (Van Rossum e Drake, 2009). Esta linguagem interpretada de alto nível é amplamente reconhecida na comunidade científica, com uma vasta gama de bibliotecas disponíveis que facilitam o desenvolvimento de soluções tanto em aprendizado de máquina e redes neurais profundas, quanto em visão computacional (Millman e Aivazis, 2011; Perkel, 2015).

Várias bibliotecas complementaram o Python no desenvolvimento deste projeto, especialmente aquelas projetadas para auxiliar na implementação de soluções de visão computacional, aprendizado de máquina e redes neurais. Destas, destaca-se o *TensorFlow* (versão 2.13.0) (Abadi et al., 2015) e *Keras* (versão 2.13.1) (Chollet, 2015), utilizados juntos como um *framework* para modelar, treinar e testar redes neurais, bem como definir otimizadores, funções de custo e métricas de desempenho (Géron, 2017; Chollet, 2021). O *OpenCV* (versão 4.8.1) (Bradski, 2000), uma biblioteca poderosa para o processamento de imagens e aplicações de visão computacio-

nal, também desempenhou um papel crucial na aplicação de filtros de imagens e na realização de testes manuais para afinar os métodos que utilizam aprendizado profundo. Adicionalmente, a biblioteca *Numpy* (versão 1.22.3) (Harris et al., 2020) foi fundamental para realizar cálculos matemáticos e manipulações matriciais (Oliphant, 2007; Van Der Walt et al., 2011).

No que diz respeito ao *hardware* empregado para a pesquisa e desenvolvimento deste projeto, os componentes de *hardware*, tais como as placas de vídeo e os processadores, foram cruciais para o processamento de dados durante as fases de treinamento e validação dos modelos de aprendizado de máquina. Inicialmente, os ensaios foram realizados na plataforma de computação em nuvem Google Colab (Bisong, 2019), que disponibilizava uma placa de vídeo Tesla T4 GPU e um processador Intel Xeon CPU. No entanto, devido às limitações de disponibilidade e considerações de custo financeiro, foi realizada a transição para um sistema físico equipado com um processador Intel Core i9-12900F CPU e uma placa de vídeo NVIDIA RTX 4090 GPU, o qual foi considerado para todos os resultados e testes descritos no presente trabalho.

Por fim, é importante ressaltar que todos os métodos desenvolvidos neste projeto com o objetivo de realizar a segmentação semântica foram disponibilizados nos repositórios Github do autor¹ de modo *open-source*, em conformidade com a licença Apache v2.0 (The Apache Software Foundation, 2004). A intenção é fomentar o crescimento acadêmico e de pesquisa, assim como facilitar futuras melhorias no campo de segmentação com o uso de camadas de *pooling* personalizadas, como a camada BPCAPooling, que é detalhada na Seção 6.1.3.

6.1.2 Extração de Características

Lidar com grandes conjuntos de dados nas redes neurais profundas é um desafio, pois a análise desses dados pode ser pesada devido à sua densidade ou à dimensionalidade elevada (Liang et al., 2017). Essa questão demanda um hardware robusto, pois quanto maior a quantidade e a dimensionalidade dos dados, maior é a exigência de poder computacional (Benyahia et al., 2022). A abordagem de extração de características é especialmente comum em ambientes de aprendizado baseado em texto, conforme discutido por Liang et al. (2017), e se estende ao uso de convoluções em CNNs, como será abordado na Seção 3.1.

Portanto, a extração de características surge como uma estratégia crucial para a redução da dimensionalidade, onde um conjunto inicial de dados brutos é transformado em grupos mais gerenciáveis para processamento (Benyahia et al., 2022). Esse processo frequentemente envolve

¹Perfil Github do autor - <https://github.com/Lucs1590>

a combinação ou seleção de variáveis, resultando em uma significativa redução na quantidade de dados a serem processados. Além disso, essas características ainda preservam a representatividade do conjunto original, sendo capazes de reduzir redundâncias para análises específicas.

Um exemplo de algoritmo capaz de agregar essas vantagens ao processo extração de características é o *Principal Component Analysis* (PCA) (Pearson, 1901), que foi desenvolvido como uma técnica estatística de transformações ortogonais que almeja transformar um conjunto de variáveis correlacionadas em um conjunto linearmente de variáveis não correlacionadas, extraído de um conjunto de dados as características essenciais (Pearson, 1901; Wold et al., 1987) e obtendo como resultado, um conjunto de dados menor, mas sem perdas em relação ao conjunto original.

O funcionamento do algoritmo de PCA para a extração de características pode ser resumido pelo passo-a-passo a seguir (Song et al., 2010):

1. Padronização dos dados, com a subtração da média μ de todas as dimensões em relação ao valor de entrada inicial z , dividido pelo desvio padrão σ , como descrito na Equação 6.1:

$$z = \frac{x - \mu}{\sigma}; \quad (6.1)$$

2. Cálculo da matriz de covariância, com X representando a primeira dimensão da entrada, conforme a Equação 6.2:

$$\mathbf{A} = \text{cov}(X, X) = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{x})(X_i - \bar{x})^T, \quad (6.2)$$

onde $\bar{x}$ representa a média da variável X e n é o número de observações;

3. Cálculo dos autovetores e autovalores correspondentes. Para a realização desse passo, o cálculo dos autovetores em relação à matriz de covariância $\mathbf{A}$ pode ser representado pela Equação 6.3:

$$\mathbf{A} \vec{v} = \lambda \vec{v}, \quad (6.3)$$

em que λ representa um vetor de autovalores. O que significa que a transformação linear é definida por λ e a equação pode ser transcrita pela Equação 6.4:

$$\mathbf{A}\vec{v} - \lambda\vec{v} = 0 \Rightarrow \vec{v}(\mathbf{A} - \lambda\mathbf{I}) = 0, \quad (6.4)$$

tendo $\mathbf{I}$ como uma matriz identidade.

4. Escolha de k autovetores com maiores autovalores, após realizar uma organização decrescente de autovalores, como demonstra a Equação 6.5:

$$\mathbf{W}_k = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_k \end{bmatrix}. \quad (6.5)$$

Nessa equação, $\mathbf{W}_k$ representa a matriz formada pelos k autovetores correspondentes aos maiores autovalores selecionados após a organização decrescente dos autovalores. Cada autovetor $\vec{v}_i$ faz parte da matriz $\mathbf{W}_k$ e compõe as novas dimensões do conjunto de dados transformado, onde k é o número desejado de bases para o novo conjunto de dados após a aplicação do PCA.

Além disso, é importante mencionar que o PCA também pode ser obtido através do método de Decomposição em Valores Singulares (*Singular Value Decomposition* - SVD) (Stewart, 2006), o qual é utilizado para extrair os componentes principais do conjunto de dados de entrada.

Entre as vantagens do algoritmo PCA, destaca-se a eliminação das correlações entre as características, garantindo a independência entre os componentes principais e evitando viés em relação a características específicas. Além disso, o algoritmo melhora o desempenho e reduz o custo computacional ao diminuir as dimensões dos dados, especialmente útil ao lidar com conjuntos de dados de alta dimensionalidade. Finalmente, o PCA ajuda a reduzir problemas de superajustamento ao diminuir o número de características, mitigando os riscos associados a ter muitas variáveis e, frequentemente, facilita a visualização dos dados ao transformar dados de alta dimensão em uma forma de dimensão mais baixa e mais gerenciável, simplificando a compreensão dos dados.

No entanto, o PCA também apresenta desvantagens, como o fato de tornar os componentes gerados menos interpretáveis do que os dados originais, uma vez que os componentes principais gerados são combinações das características originais. Além disso, o PCA requer a padronização dos dados antes da implementação; sem isso, o algoritmo pode falhar em determinar os componentes principais ideais. Finalmente, embora o PCA busque capturar a máxima variância entre as características, uma seleção inadequada dos componentes principais pode re-

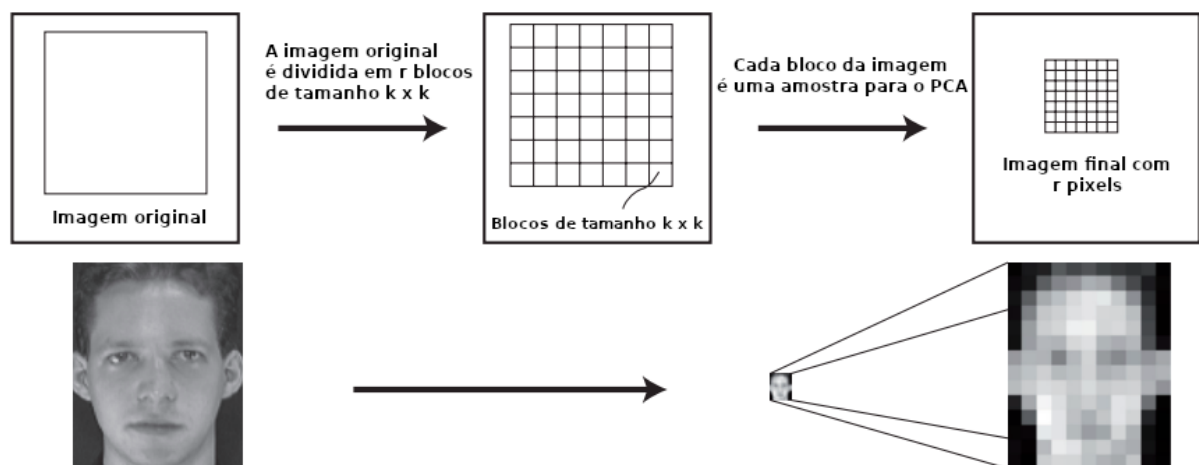
sultar na perda de alguma informação em comparação com o conjunto original de características.

6.1.3 *Block-based Principal Component Analysis Pooling (BPCAPooling)*

A técnica *Block-based Principal Component Analysis* (BPCA) foi introduzida como uma variação do método convencional do *Principal Component Analysis* (PCA) no trabalho de Salvadeo et al. (2011), propondo um método que inicialmente foi utilizado para a extração de características de imagens como parte de uma *pipeline* de reconhecimento facial no conjunto de dados *The AR Face Database*² (Martinez e Benavente, 1998), propondo um método de extração que trouxesse à análise menor custo computacional, redução de dimensionalidade e uma geometria de hiperespaço que fosse mais intuitiva, destacando o diferencial de que com esse método fosse possível preservar a espacialidade da informação.

O método proposto opera por meio da subdivisão de uma imagem em blocos $k \times k$, em que os blocos geralmente possuem tamanhos iguais (por exemplo, 3×3 , 8×8 , etc.). Em seguida, considerando cada bloco como uma amostra, o PCA é aplicado, sendo que cada um desses blocos são projetados no espaço do componente principal. Consequentemente, cada bloco $k \times k$ passa a ter um tamanho reduzido, definindo uma imagem com r características (o mesmo número de blocos), posicionadas espacialmente nos respectivos blocos. Esse processo pode ser visualizado através da Figura 6.1.

Figura 6.1: Exemplo de aplicação do BPCA.



Fonte: retirado e adaptado de Salvadeo et al. (2011).

Ao contrário de uma simples combinação linear de amostras distintas, o BPCA compõe uma representação organizada, formada por pequenos blocos do mapa de características original,

²AR Face Database Webpage - <https://www2.ece.ohio-state.edu/~aleix/ARdatabase.html>

como ilustrado na Figura 6.1. Essa técnica incorpora a teoria do PCA para realizar o mapeamento não supervisionado, visando a minimização do erro quadrático médio entre a entrada original e a representação projetada. Ao preservar mais informações relevantes, o BPCA é capaz de sintetizar características essenciais da imagem de entrada em uma representação compacta de maior relevância contextual (Kuncheva e Faithfull, 2014). Isso torna o BPCA um componente eficaz para lidar com grandes volumes de dados, como é comum no processamento de imagens. A compactação obtida pelo BPCA é particularmente benéfica, já que maximiza a variância dos dados e minimiza a perda de informações significativas, diferentemente de outros métodos de pooling que tendem a descartar uma quantidade considerável de informação relevante.

As camadas de *pooling* têm sido eficazes em CNNs para reduzir a dimensionalidade dos dados (Paul e Chaki, 2019). Neste trabalho, a inovação proposta é a aplicação da técnica BPCA como uma camada de *pooling* - sendo nomeada de BPCAPooling - não mais para extração de características, conforme proposto originalmente por Salvadeo et al. (2011), mas como uma estratégia para preservar a espacialidade da informação entre as camadas convolucionais, num olhar global para o mapa de atributos, associada ao benefício proposto pela técnica de minimização do erro quadrático médio entre a entrada e saída como uma camada de *pooling*.

Essa camada de *pooling* segue o processo original do BPCA, onde são extraídas informações principais de cada bloco, reduzindo a dimensionalidade dos dados e preservando características relevantes. Além disso, o método foi aprimorado pela normalização dos blocos utilizando média (μ) e desvio padrão (σ). Na sequência, é aplicada a técnica de *Singular Value Decomposition* (SVD) para extrair os componentes principais (C), como explicado na Seção 6.1.2, e, finalmente, os blocos são transformados por meio da projeção nesses componentes selecionados. O resultado é um mapa de características menor em tamanho, porém, preservando informações essenciais ao concatenar os blocos transformados.

A formulação matemática do processo realizado pela camada de *pooling* BPCAPooling pode ser representada pela Equação 6.6:

$$\begin{aligned} Y_{i,j,k} &= \text{BPCAPooling}(\mathbf{X}_{i',j',k'}) \\ &= \text{reshape} \left(\text{SVD} \left(\frac{\mathbf{X} - \mu}{\sigma} \right) \cdot C \right), \end{aligned} \quad (6.6)$$

onde $\mathbf{X}$ é a imagem de entrada com i' como altura de entrada, j' como largura de entrada e k' como número de canais de entrada; e Y é a imagem de saída, com i representando altura de

saída, j como largura de saída e k como profundidade de saída (ou seja, o número de canais). Aqui, *reshape* reorganiza os blocos transformados para uma estrutura de saída apropriada, SVD representa a decomposição em valores singulares dos blocos normalizados, μ é o vetor médio dos blocos extraídos, σ é o vetor de desvio padrão dos blocos extraídos, e C é uma matriz contendo os primeiros componentes principais selecionados.

Essa formulação matemática descreve o processo de *pooling* realizado pela camada BPCAPooling, que extrai informações relevantes dos blocos de entrada, reduzindo sua dimensionalidade e preservando características importantes para o aprendizado eficiente da rede neural convolucional.

Quando se analisa a complexidade de uma função em notação Big O, que descreve o quão escalável é a função, pode-se expressá-la como $O(HWD * \min(P^2, ND))$ no pior caso. Isso é resultado da complexidade em cada um dos passos dos seguintes processos:

1. Extração de *patches* do mapa de características de entrada, de tamanho (H, W, C) , onde H , W e C representam altura, largura e número de canais, respectivamente, gerando *patches* de dimensão (P, P, D) . Esse processo é representado pela notação $O(HWD * P^2)$;
2. Normalização utilizando a média e o desvio padrão, com a notação $O(HWD)$;
3. Aplicação da decomposição em valores singulares em uma matriz (HWD, P^2) , com notação $O(\min(P^2, HWD * D))$;
4. Multiplicação matricial entre os *patches* (HWD, P^2) e os componentes principais (P^2, N) , resultando em $O(HWD * N * D)$;
5. O processo de *reshaping*, considerado de complexidade linear e desconsiderado na notação Big O (Devi et al., 2011).

Resumindo, a complexidade é expressa como $O(HWD * \min(P^2, ND))$, onde P é o tamanho dos *patches*, N é o número de componentes principais, e D é a quantidade de blocos.

Em suma, a camada de BPCAPooling emerge como uma proposta inovadora para aprimorar a atuação das redes neurais convolucionais ao lidar com tarefas de segmentação semântica. Ao incorporar a técnica de BPCA, essa camada de *pooling* permite que a informação espacial seja preservada na passagem entre as camadas convolucionais, mantendo simultaneamente uma redução de dimensionalidade efetiva dos dados.

6.1.4 Conjuntos de Dados

Os conjuntos de dados selecionados para esta pesquisa são essencialmente de duas categorias distintas. Dois deles - CIFAR100 e *Food-101* - foram empregados especificamente para o treinamento, teste e validação voltados exclusivamente à classificação de imagens (Krizhevsky e Hinton, 2009; Bossard et al., 2014).

Nessa configuração, os modelos de classificação foram treinados para atribuir um rótulo a cada imagem, identificando o objeto de interesse sem foco na sua localização ou delimitação por caixas. Em outras palavras, o objetivo foi reconhecer a presença de um objeto em particular na imagem inteira, sem segmentar ou isolar essa entidade (Viitaniemi e Laaksonen, 2008).

Por outro lado, um terceiro conjunto de dados - que será detalhado na Seção 6.1.4.1 - foi preparado especificamente para treinar e validar modelos de segmentação semântica. Este conjunto de dados inclui máscaras que mapeiam a localização dos objetos de interesse em todos os *pixels* da imagem de entrada.

O emprego desses conjuntos diversificados de dados permite comparar e contrastar a eficácia de diferentes abordagens de segmentação e classificação no campo da visão computacional. Também permite avaliar como esses diferentes métodos respondem a variações nas características dos dados.

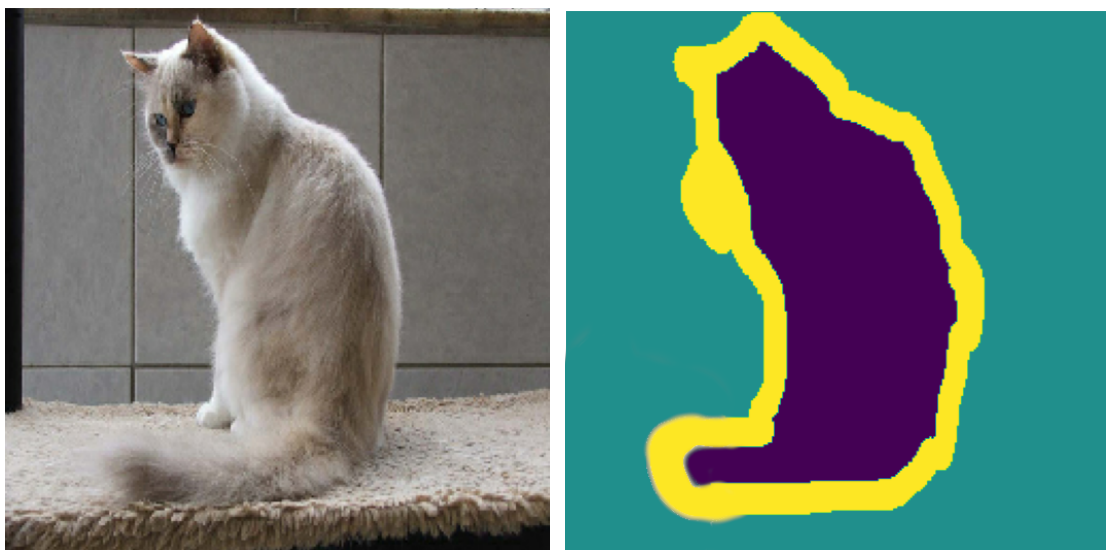
As Seções 6.1.4.2 e 6.1.4.3 a seguir detalharão mais apropriadamente os conjuntos de dados utilizados para os experimentos de classificação, e descreverão como foram utilizados no escopo desta pesquisa.

6.1.4.1 *Oxford-IIIT Pets*

O conjunto de dados *Oxford-IIIT Pets*³ (Parkhi et al., 2012) é um composto majoritariamente de exemplos de gatos e cachorros. Esse *dataset* possui cerca de 200 exemplos para 37 classes diferentes, resultando no total de 7.349 amostras. Essas classes são identificadas pela raça do animal de estimação em questão, mas cada uma dos registros nesse conjunto de dados trás consigo o nome da imagem, o rótulo baseado na raça, a espécie (gato ou cachorro), a imagem com o animal e uma máscara de características com o fundo, contorno e animal mapeados cada um de uma cor, como representado pela Figura 6.2.

³Conjuntos de dados *Oxford-IIIT Pets* - <https://www.robots.ox.ac.uk/~%7Evgg/data/pets/>

Figura 6.2: Exemplo de amostra com máscara de características.



(a) Imagem original do animal de estimação.

(b) Máscara com fundo em verde, contorno em amarelo e animal em roxo.

Fonte: retirado e adaptado de [Parkhi et al. \(2012\)](#).

Vale dizer que originalmente essas amostras estão separadas entre 3.669 (49,925%) para a etapa de teste e 3.680 (50,075%) para a etapa de treinamento. Todavia, nos testes realizados foi utilizado também um conjunto de dados voltado para validação, o qual foi extraído a partir do *dataset* de teste. Sendo assim, para o presente trabalho, a divisão passou a ser de 669 exemplos para teste, 3.000 amostras para validação e 3.680 instâncias para treino, levando em conta os conceitos tratados na Seções 2.2 e 2.3.

É relevante salientar que as imagens do conjunto de dados original são tridimensionais, ou seja, cada amostra inclui informações dos canais de cores vermelho, verde e azul. Já as máscaras são unidimensionais, apresentando um mapa de valores singulares para representar as regiões de interesse baseadas na imagem original. Os valores numéricos nas máscaras correspondem ao fundo, à borda ou ao animal.

Vale observar que diferentemente de muitos outros conjuntos de dados de imagem ([Bos-sard et al., 2014](#)), o *Oxford-III Pets* não apresenta dimensões padronizadas para suas amostras. Todavia, no atual trabalho, houve uma alteração nesse aspecto, em que foram aplicadas duas técnicas, além da técnica de aumento de dados que será discutida na Seção 6.1.6, em todos os exemplos do conjunto de dados. Sendo que a primeira técnica aplicada foi a padronização tanto das imagens originais, quanto das máscaras, em relação às suas dimensões, em que ambas passaram a ter o valor de 256×256 , que é o um valor comumente utilizado na área de imagens para substantiar as características da mesma ([Lee, 1983](#)).

A normalização dos valores do conjunto de dados foi realizada para garantir que tanto as imagens quanto as máscaras estivessem dentro de um intervalo padronizado de 0 a 1, facilitando o processamento e a comparação entre diferentes conjuntos de dados.

Para as imagens, essa normalização é feita dividindo cada valor de pixel pelo valor máximo possível, que é 255 para imagens de 8 bits, conforme mostrado na Equação 6.7:

$$I_{\text{norm}}(i, j, k) = \frac{I(i, j, k)}{255}, \quad (6.7)$$

onde I_{norm} representa a imagem de entrada normalizada, I representa a imagem de entrada e (i, j, k) são as coordenadas de altura, largura e canal, respectivamente.

No caso das máscaras, a normalização foi realizada de maneira diferente. Inicialmente, as máscaras tinham valores discretos de 1, 2 e 3, representando o animal em questão, a borda e o fundo, respectivamente. Para normalizar esses valores para o intervalo de 0 a 1, foi subtraído um valor unitário de cada pixel, conforme mostrado na Equação 6.8:

$$M_{\text{norm}} = M - 1, \quad (6.8)$$

onde M_{norm} representa a máscara de entrada normalizada, M representa a máscara de entrada e (i, j, k) são as coordenadas de altura, largura e canal, respectivamente. Essa transformação garante que as máscaras estejam na faixa desejada para facilitar o processamento subsequente.

Por fim, é relevante ressaltar que esse conjunto de dados se mostra adequado para o treinamento em segmentação semântica, uma vez que conta com todos os dados devidamente rotulados, proporcionando um cenário propício para esse propósito. O resultado gerado por essa abordagem será um mapa de características, ilustrado de forma análoga ao apresentado na Figura 6.2b.

6.1.4.2 CIFAR 100

O conjunto de dados CIFAR-100⁴ (Krizhevsky et al., 2014) é composto por um total de 60.000 imagens, cada uma com dimensões de 32×32 pixels. Estas imagens são categorizadas em 20 “superclasses”, cada uma contendo 5 “classes finas”.

De maneira geral, o *dataset* possui 100 classes, com 600 amostras em cada classe. Cada exemplo neste conjunto de dados é rotulado com uma *superclass*, que representa uma categoria

⁴Conjuntos de dados CIFAR-10 e CIFAR-100 - <https://www.cs.toronto.edu/~kriz/cifar.html>

mais abrangente (como “veículo”), e uma *fine class*, que representa uma categoria mais específica (como “bicicleta”, “ônibus”, etc.). Alguns exemplos dessas amostras podem ser visualizados na Figura 6.3.

Figura 6.3: Exemplos de amostras do conjunto de dados CIFAR-100.



Fonte: retirado e adaptado de Krizhevsky et al. (2014).

Em contraste com a configuração original proposta por Krizhevsky et al. (2014), dividimos o conjunto de dados em 83,33% para treinamento, 11,66% para validação e 5% para teste, visando evitar qualquer viés nos resultados, conforme recomendado por Domingos (2012). Além de aplicar um processo de normalização das amostras, deixando o valor de todos os *pixels* das amostras entre o domínio de $[0, 1]$, com um processo muito semelhante ao da Equação 6.7.

Finalmente, vale citar que todas as imagens do conjunto de dados se encontram rotuladas e prontas para serem usadas na tarefa de classificação.

6.1.4.3 Food-101

O conjunto de dados Food-101⁵ (Bossard et al., 2014) foi empregado como um segundo conjunto de dados para a classificação de imagens neste trabalho. Este *dataset* contém 101.000

⁵Conjunto de dados Food-101 - https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/

imagens, originalmente com uma resolução de 512×512 *pixels*, no entanto, para otimizar o processamento do modelo e manter a representatividade da amostra, essas imagens foram redimensionadas para 224×224 *pixels*.

Ao contrário do conjunto CIFAR-100, que abrange classes mais generalistas (vide Seção 6.1.4.2), o conjunto de dados *Food-101* se concentra exclusivamente em classes relacionadas à alimentação. Ele é composto por 101 classes (por exemplo, “pizza”, “bruscheta”, entre outros), cada uma com 1.000 amostras. Alguns exemplos dessas classes podem ser visualizados na Figura 6.4.

Figura 6.4: Exemplos de amostras do conjunto de dados *Food-101*.



pho



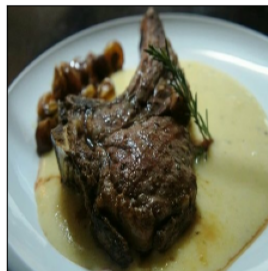
samosa



peking_duck



pancakes



prime_rib



mussels



beignets



baby_back_ribs



gnocchi

Fonte: Adaptado de [Bossard et al. \(2014\)](#).

Adicionalmente, houve uma reorganização do conjunto de dados em relação à sua configuração original, sendo que 75% das amostras foram designadas para o conjunto de treinamento, 17,5% para validação e 7,5% para teste. Além disso, foi aplicado um pré-processamento de normalização, conforme representado na Equação 6.7, com o intuito de reduzir o viés do modelo

durante o treinamento (Shorten e Khoshgoftaar, 2019).

É importante ressaltar também que o conjunto de dados *Food*-101 também foi empregado no processo de classificação de imagens. Ele atua como um conjunto alternativo, permitindo demonstrar a capacidade de generalização do modelo e da técnica utilizada em diversos cenários. Devido ao seu tamanho, uma amostra do *dataset Food*-101 contém aproximadamente dezesseis vezes mais informações do que uma amostra do CIFAR-100, o que proporciona uma maior riqueza de detalhes e complexidade para o treinamento e validação do modelo.

6.1.5 Transferência de Aprendizado

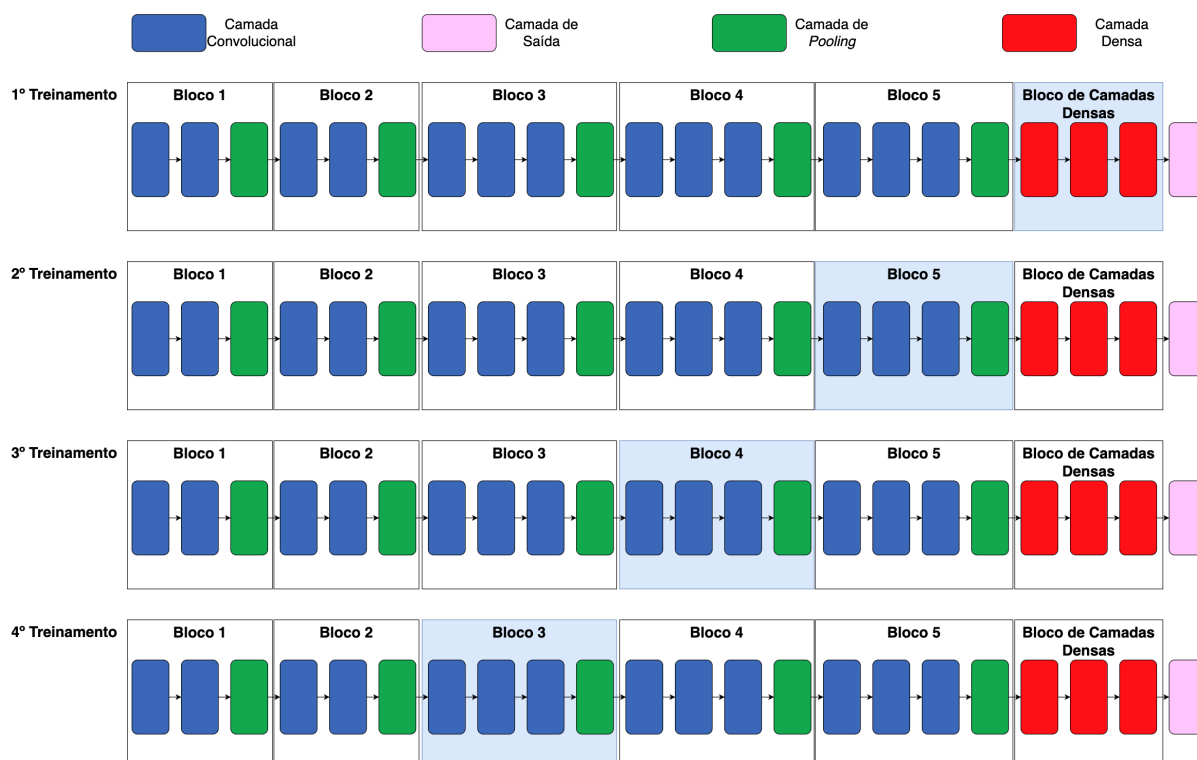
A técnica de transferência de aprendizado, empregada neste estudo, encontra inspiração em cenários do cotidiano, onde um conhecimento prévio sobre um tema pode ser crucial para acelerar e incrementar a aprendizagem de novos conteúdos (Pan e Yang, 2010). Esta técnica mostra-se especialmente útil em situações onde se dispõe de um conjunto de dados alvo com uma quantidade limitada de amostras para o treinamento de uma rede (Weiss et al., 2016), considerando que Redes Profundas necessitam de uma vastas quantidades de dados para obter um desempenho satisfatório (Goodfellow et al., 2016).

Sob essa perspectiva, o treinamento das redes de classificação VGG-16 para os conjuntos de dados CIFAR 100 e *Food*-101 se ancorou nos conceitos de transferência de aprendizado. Utilizamos os pesos da rede VGG16 pré-treinada no *dataset* ImageNet⁶ (Deng et al., 2009), que dispõe de um milhar (1.000) de classes com 1.281.167, 50.000 e 100.000 exemplos para treinamento, validação e testes, respectivamente.

Dessa forma, a inicialização da rede neural ocorreu com os pesos pré-treinados da VGG-16. Ao longo dos treinamentos, foi empregado o procedimento de descongelamento progressivo das camadas da VGG-16, começando pelas camadas densas e alcançando até o terceiro bloco convolucional, sem ultrapassar esse ponto para preservar as características de baixo nível presentes nos pesos do treinamento com o ImageNet. A estratégia adotada foi a de alternar entre o descongelamento de blocos e a continuação dos treinamentos, sempre escolhendo o melhor modelo, conforme exemplificado na Figura 6.5 e sugerido por Chollet (2021). Essa abordagem foi aplicada em ambos os conjuntos de dados e nos métodos de *pooling* avaliados (*AvgPooling*, *MaxPooling* e *BPCAPooling*).

⁶Conjunto de dados ImageNet - <https://www.image-net.org/download.php>

Figura 6.5: Processo de *fine-tuning* da VGG-16, com o descongelamento dos blocos representados pelo fundo azul.



Fonte: do próprio autor.

No que tange aos testes efetuados com os modelos U-Net e U-Net-Like, é relevante salientar que não se empregou as técnicas de transferência de aprendizado, devido a alguns aspectos cruciais da própria técnica. O primeiro desafio é a necessidade de utilizar a mesma arquitetura para o modelo de base adotado para a transferência de aprendizado e para o modelo treinado. Ainda que soluções para essa questão tenham sido propostas, como a adoção de camadas da VGG-16 dentro da estrutura de uma U-Net (Pravitasari et al., 2020), a aplicação de tal técnica poderia afetar os resultados pretendidos decorrentes exclusivamente da modificação da camada de *pooling* nos modelos U-Net e U-Net-Like. Adicionalmente, a escassez de conjuntos de dados robustos, similares ao ImageNet, direcionados ao processo de segmentação semântica, configura outro obstáculo, especialmente considerando a falta de padronização no formato das anotações. Como referência para um adequado conjunto de dados e modelo de anotação, podemos citar o COCO-stuff (Caesar et al., 2018) e o formato de anotação COCO, que pode ser desenvolvido a partir da ferramenta COCO Annotator (Brooks, 2019).

6.1.6 Aumento de Dados

O uso de técnicas de aumento de dados tem se revelado eficaz para o fortalecimento da capacidade de aprendizado dos modelos, especialmente no que tange à mitigação de problemas de *overfitting*, como discutido na Seção 3.6. Portanto, é relevante destacar a importância da implementação de transformações geométricas e de intensidade de cor, as quais foram empregadas de forma diversificada e com base em uma abordagem probabilística estocástica nos experimentos desenvolvidos neste trabalho. Adicionalmente, cabe enfatizar que a probabilidade de aplicação do aumento de dados durante o processo de treinamento a cada imagem pode resultar na aplicação simultânea de múltiplas transformações na mesma imagem.

É importante enfatizar que, em todos os conjuntos de dados utilizados, a técnica de aumento de dados foi aplicada exclusivamente nos subconjuntos de treinamento. Esta abordagem foi adotada considerando que os subconjuntos de teste e validação são empregados para avaliar o desempenho do modelo, não para ensiná-lo em diferentes cenários.

As próximas seções, especificamente 6.1.6.1, 6.1.6.2, e 6.1.6.3, discutem as transformações aplicadas nos conjuntos de dados CIFAR 100, *Food-101* e *Oxford-IIIT Pets*, respectivamente. Estas discussões proporcionam *insights* relevantes que poderão orientar futuras pesquisas na área e facilitar a reprodução dos testes conduzidos neste estudo.

6.1.6.1 CIFAR 100

Para o conjunto de dados CIFAR 100, optou-se primordialmente por aplicar transformações geométricas como técnicas de aumento de dados. Considerando a diversidade das classes presentes no conjunto e o baixo risco de distorção de significado do rótulo do exemplo em questão - uma preocupação discutida por Shorten e Khoshgftaar (2019) no contexto do conjunto de dados MNIST -, foi possível utilizar transformações com uma variação considerável sem, contudo, deixar a imagem irreconhecível para a interpretação humana.

As transformações aplicadas no conjunto de dados CIFAR 100, bem como a variação permitida para cada uma delas, estão detalhadamente descritas na Tabela 2.

Tabela 2: Transformações geométricas aplicadas no conjunto de dados CIFAR 100.

Tipo de transformação	Intervalo	Parâmetro
Deslocamento Vertical	$-25\% - 25\%$	-
Deslocamento Horizontal	$-25\% - 25\%$	-
Cisalhamento	$0^\circ - 25^\circ$	-
Zoom	$0\% - 25\%$	-
Rotação	$0^\circ - 45^\circ$	-
Espelhamento Horizontal	-	Verdadeiro
Modo de Preenchimento	-	Mais próximo

Fonte: do próprio autor.

É importante ressaltar que, no contexto dos parâmetros listados na Tabela 2, os termos “Espelhamento Horizontal” e “Modo de Preenchimento” indicam, respectivamente, que a imagem pode ser submetida a um espelhamento horizontal e que o tipo de preenchimento escolhido, dentre as opções “constante”, “mais próximo”, “refletir” ou “envolver”, foi o método “mais próximo”. O efeito dessas transformações pode ser observado por meio do exemplo apresentado na Figura 3.9.

6.1.6.2 Food-101

No que se refere ao *dataset* Food-101, é importante salientar que, apesar da técnica de aumento de dados ter sido aplicada e utilizada em alguns testes, somente foi possível implementá-la manualmente dentro do modelo. Isso resultou no aumento da complexidade do modelo padrão VGG-16, devido à necessidade de se acrescentar uma camada inicial responsável pela aplicação dos filtros de forma estocástica. Logo, os resultados comentados na Seção 7, referentes ao conjunto de dados Food-101, não contemplam a aplicação da técnica de aumento de dados.

Ressalta-se, no entanto, que similarmente ao conjunto CIFAR 100, o *dataset* Food-101 não apresenta problemas relacionados à alteração da interpretação dos rótulos em função da aplicação do aumento de dados. Dessa forma, as transformações utilizadas para os testes são descritos na seguinte Tabela 3.

Tabela 3: Transformações geométricas aplicadas no conjunto de dados *Food-101*.

Tipo de transformação	Intervalo	Parâmetro
Deslocamento Vertical	$-25\% - 25\%$	-
Deslocamento Horizontal	$-25\% - 25\%$	-
<i>Zoom</i>	$0\% - 25\%$	-
Rotação	$-45^\circ - 45^\circ$	-
Espelhamento Horizontal	-	Verdadeiro

Fonte: do próprio autor.

Neste conjunto de dados, nota-se uma ampliação na taxa de zoom em relação ao CIFAR, o que pôde ser explorado devido à maior resolução das amostras em relação ao primeiro conjunto. A representação visual do resultado da aplicação deste aumento de dados está evidenciada na Figura 6.6.

Figura 6.6: Ilustração do aumento de dados com transformações geométricas aplicadas ao conjunto de dados *Food-101*.



(a) Imagens Originais.

(b) Imagens com transformações.

Fonte: do próprio autor. com base no conjunto de dados *Food-101* (Bossard et al., 2014).

6.1.6.3 *Oxford-IIIT Pets*

Para o conjunto de dados *Oxford-IIIT Pets*, destinado à aplicação dos métodos de segmentação semântica com uso das arquiteturas U-Net-Like e convencionais U-Nets, um procedimento de aumento de dados foi implementado previamente à fase de treinamento dos modelos. Isto permitiu que as transformações empregadas fomentassem o aprendizado dos modelos e contribuíssem para a melhoria nas métricas de avaliação, que serão discutidas em detalhes na Seção 6.2.

Contrariamente aos conjuntos de dados direcionados à classificação, nesse conjunto de dados foram aplicadas transformações que alteraram a cor das imagens, além de modificações geométricas. As especificações dessas transformações podem ser visualizadas na Tabela 4.

Tabela 4: Transformações executadas no conjunto de dados *Oxford-IIIT Pets*.

Tipo de transformação	Intervalo	Parâmetro
Contraste	0% – 20%	-
Brilho	0% – 35%	-
Saturação	0% – 20%	-
Rotação	0° – 90°	-
Borramento	-	Verdadeiro
Espelhamento Horizontal	-	Verdadeiro
Espelhamento Vertical	-	Verdadeiro

Fonte: do próprio autor.

Neste conjunto de dados, percebe-se que as transformações aplicadas são similares às aplicadas nos outros conjuntos de dados senão pelas transformações de cor. Entretanto, aqui os intervalos se referem às intensidades de alteração no contraste, brilho e saturação das imagens. A Figura 6.7 ilustra a aplicação aleatória das transformações descritas na Tabela 4, juntamente com a máscara correspondente.

Figura 6.7: Ilustração do aumento de dados com transformações aplicadas ao conjunto de dados *Oxford-IIIT Pets*.



Fonte: do próprio autor. com base no conjunto de dados *Oxford-IIIT Pets* (Parkhi et al., 2012).

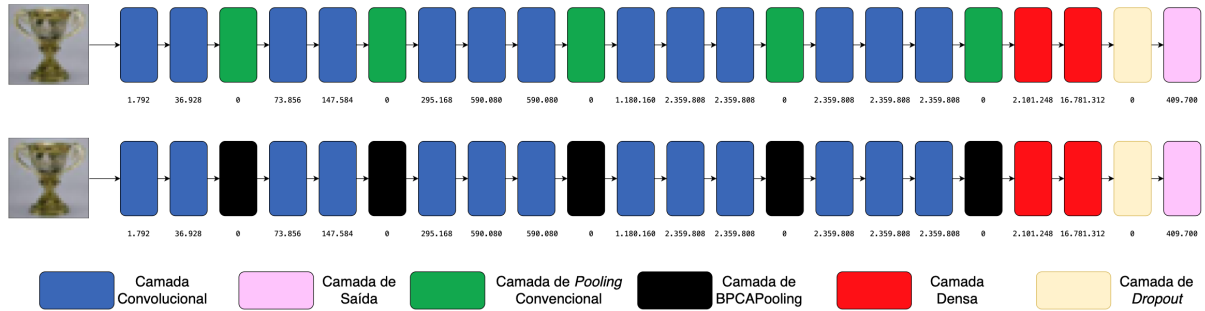
6.1.7 Alteração na Camada de *Pooling*

A camada de *pooling* desempenha um papel crucial na arquitetura das CNNs, contribuindo significativamente para a redução da dimensionalidade e a invariância de translação, como mencionado na Seção 3.2. Nessa seção será discutido sobre a substituição das operações *Max Pooling* e *Average Pooling*, amplamente utilizadas (Özdemir, 2023), nas arquiteturas VGG-16, U-Net-Like e U-Net pelo método BPCAPooling que foi proposto no presente trabalho e é detalhado na Seção 6.1.3.

É fundamental salientar que a modificação da camada de *pooling* envolveu a inclusão de argumentos adicionais para a camada. Enquanto os métodos de *pooling* convencionais normalmente recebem parâmetros como a camada anterior, tamanho de passo (*stride*) e tamanho do *kernel*, para o processo BPCAPooling, além desses parâmetros, foi necessário especificar o número de componentes principais atribuídos a cada bloco. Tipicamente, esse número foi definido como 1 para manter a consistência entre os tamanhos de blocos e o subconjunto de *pixels* (conforme discutido na Seção 6.1.3), e o tamanho esperado da saída, o que estava diretamente relacionado à quantidade de dados de entrada.

Na arquitetura VGG-16, utilizada para classificação de imagens nos conjuntos de dados CIFAR-100 e *Food-101*, as camadas de *Max Pooling* e *Avg Pooling* foram substituídas pelo BPCAPooling. A Figura 6.8 ilustra essa substituição para o conjunto de dados CIFAR-100.

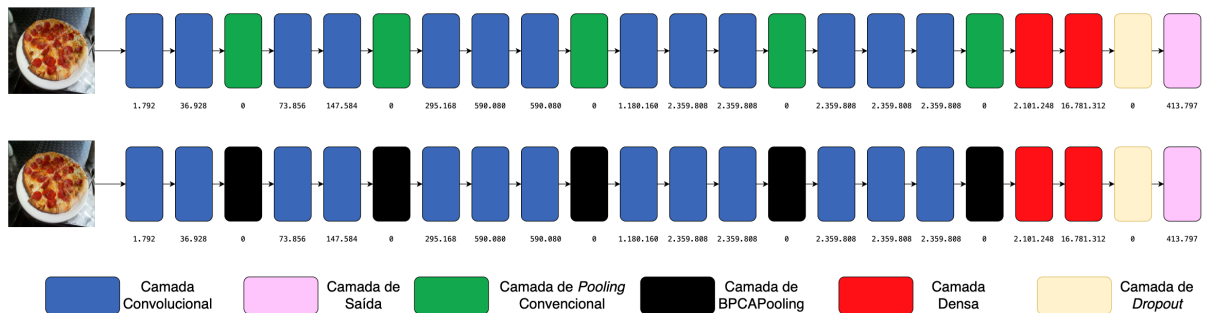
Figura 6.8: Demonstração da substituição das camadas de *pooling* convencionais por BPCAPooling no conjunto de dados CIFAR-100. Os números sob cada camada representam o número de parâmetros treináveis.



Fonte: do próprio autor.

Quanto à alteração da camada de *pooling* na VGG-16 para o conjunto de dados *Food-101*, a única diferença notável foi na camada de saída, que agora possui 413.797 parâmetros. Isso se deve ao fato de que a camada de saída é sensível ao tamanho da imagem. A Figura 6.9 ilustra essa mudança para o conjunto de dados *Food-101*.

Figura 6.9: Demonstração da substituição das camadas de *pooling* convencionais por BPCAPooling no conjunto de dados *Food-101*. Os números sob cada camada representam o número de parâmetro.



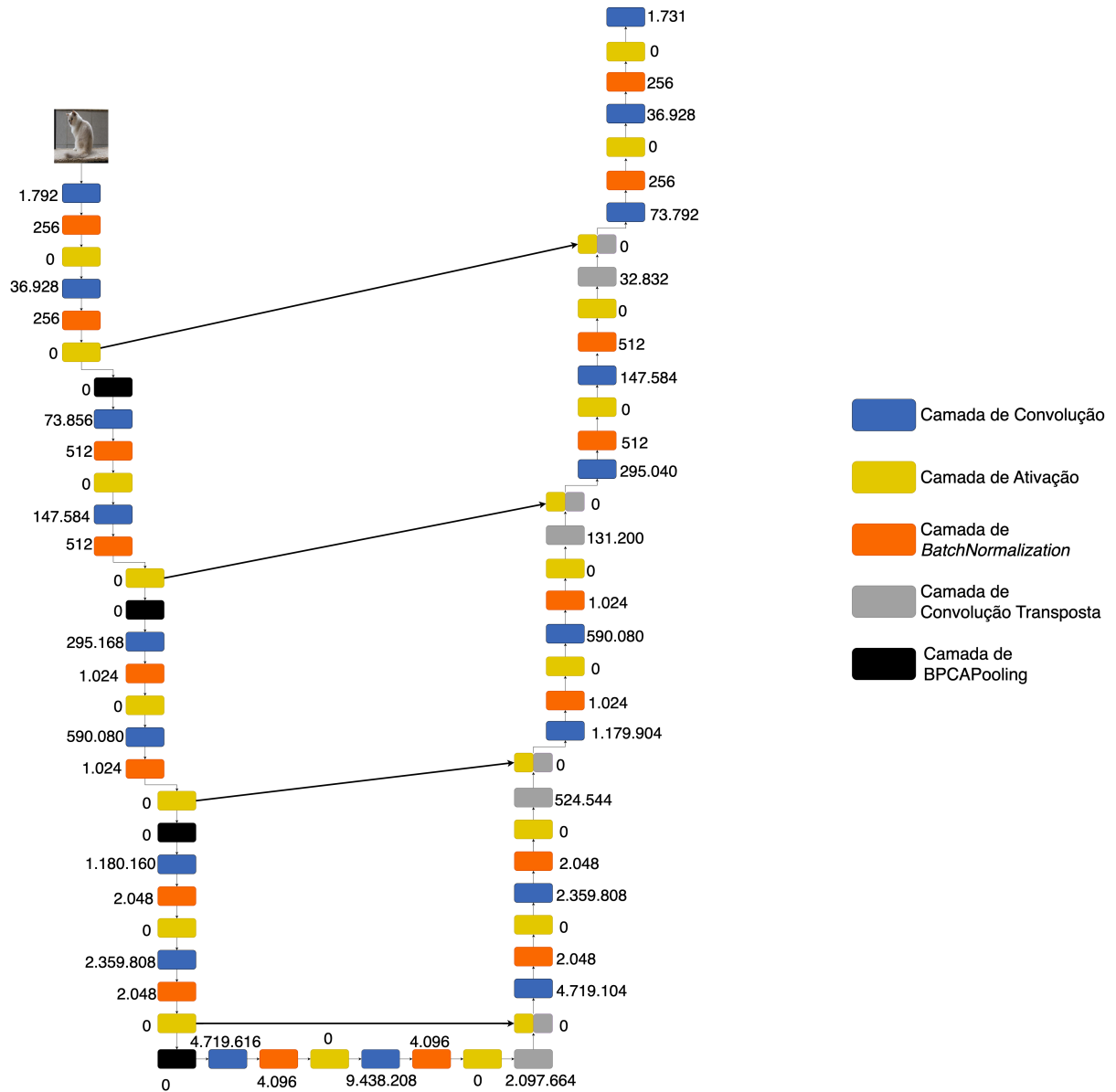
Fonte: do próprio autor.

Dessa forma, ao comparar as alterações nas camadas em relação à VGG-16 original para ambos os conjuntos de dados, nota-se a inclusão de uma camada de *dropout* no final da rede, visando mitigar problemas de *overfitting*, conforme discutido na Seção 3.3, que assim como as camadas de *pooling*, não é visível uma influência quanto à quantidade de parâmetros treináveis. A rede para o conjunto de *Food-101* possui um total de 34.011.045 de parâmetros, enquanto para o conjunto CIFAR-100, são 34.006.948 parâmetros.

Da mesma forma, na arquitetura U-Net-Like, empregada para segmentação semântica no conjunto de dados *Oxford-IIIT Pets*, as camadas de *Max Pooling* foram substituídas pela BPCA-Pooling. Vale ressaltar que, para as arquiteturas de segmentação semântica, não foi explorado o uso de *Avg Pooling*, pois esse tipo de *pooling* não é comumente empregado em arquiteturas de estado da arte (Ronneberger et al., 2015; Kugelman et al., 2022). Essa adaptação foi realizada com o intuito de aprimorar a precisão da segmentação semântica, mantendo a informação espacial das entradas durante o processamento pelas diversas camadas. A ideia é que a estrutura geral da imagem fosse preservada ao longo das operações realizadas, garantindo que a imagem não fosse distorcida.

A arquitetura com as modificações citadas, bem como a quantidade de parâmetros treináveis ao lado da camada representada, pode ser visualizado na Figura 6.10.

Figura 6.11: Arquitetura U-Net com aplicação de BPCAPooling.



Fonte: do próprio autor.

A comparação entre as arquiteturas apresentadas nas Figuras 6.10 e 6.11 permite uma análise detalhada dos aspectos abordados na Seção 5.1. Totalizando em 31.056.963 parâmetros treináveis, enquanto a U-Net-Like conta com 2.058.979 parâmetros treináveis, para ambas as abordagens de *pooling*, a U-Net possui aproximadamente quinze vezes mais parâmetros do que a U-Net-Like. Notavelmente, os parâmetros do BPCAPooling permanecem zerados, seguindo o comportamento do *Max Pooling*. Vale mencionar que a arquitetura U-Net-Like inclui camadas de convolução separáveis e camadas de *upsampling*. As camadas de *upsampling* foram

elucidadas na Seção 5.1.3, enquanto as camadas convolucionais separáveis realizam convoluções espaciais profundas seguidas por uma convolução pontual, que mescla os canais de saída resultantes (Zhao et al., 2018).

6.2 Metodologia Experimental

Esta seção, busca elucidar as métricas de avaliação escolhidas para determinar o desempenho dos experimentos realizados neste estudo para atingir os objetivos mencionados na Seção 1.1. Essas métricas são categorizadas em duas vertentes: classificação e segmentação semântica, que serão exploradas na Seção 6.2.1.

Com o intuito de promover a reproduzibilidade deste trabalho, oferece-se na Seção 6.2.2 detalhes acerca dos parâmetros empregados no treinamento e testes conduzidos. Adicionalmente, na Seção 6.2.3, será discorrido sobre as técnicas de explicabilidade de modelos de IA utilizadas neste estudo para melhor entender as classificações e segmentações à luz das percepções derivadas dos modelos.

6.2.1 Escolha das Métricas

Para a avaliação da VGG-16 nos processos de treinamento e teste nos conjuntos de dados CIFAR-100 e *Food*-101, foram empregadas duas métricas: *Sparse Categorical Crossentropy* para cálculo do erro e acurácia para avaliar o desempenho da rede. Essa métrica de erro é frequentemente utilizada em problemas de classificação multiclasse, como é o caso dos problemas nos conjuntos de dados mencionados, onde a saída é codificada de forma esparsa (categórica). Essa função de perda mede a diferença entre as previsões do modelo e os rótulos reais, penalizando as discrepâncias entre eles, conforme discutido na Seção 2.2.1. Essa função pode ser representada pela Equação 6.9:

$$J(\mathbf{w}) = - \sum_{i=1}^N y_i \cdot \log(\hat{y}_i), \quad (6.9)$$

em que $\mathbf{w}$ denota os parâmetros do modelo, N é o número total de amostras, y_i representa o rótulo real da amostra i , e $\hat{y}_i$ é a previsão do modelo para a amostra i .

Quanto à acurácia, discutida na Seção 2.3.1 e representada pela Equação 2.21, vale ressaltar que foi escolhida como métrica principal. Decisões durante o treinamento e teste, como o ponto de parada antecipado (Seção 2.2.2.3), ajuste planejado da taxa de aprendizado (Seção

2.2.2.4), seleção do melhor modelo, entre outros, foram baseadas no progresso dessa métrica.

Para avaliar as segmentações semânticas usando as arquiteturas U-Net-Like e U-Net no conjunto de dados *Oxford-IIIT Pets*, uma variedade de métricas foi empregada nos testes. Inicialmente, a métrica principal adotada foi a acurácia, conforme utilizado nos testes de classificação. Entretanto, o uso da métrica de acurácia é desencorajado para problemas de segmentação, já que muitos desses problemas podem enfrentar desequilíbrios entre as classes de interesse e o fundo (Müller et al., 2022). Assim, a métrica principal adotada para os modelos de segmentação semântica passou a ser o *Mean Intersection Over Union*, ou mIoU, detalhado na Seção 5.2.3, representado pela Equação 5.3, superando as limitações da acurácia (Müller et al., 2022).

Adicionalmente, com a adoção da métrica mIoU como métrica principal, incorporou-se o uso do *Dice Similarity Coefficient* (DSC) mencionado na Seção 5.2.4, para fornecer uma visão adicional sobre a performance dos modelos treinados. Quanto à métrica de função de custo empregada nos problemas de segmentação semântica, destaca-se que os desafios encontrados no conjunto de dados *Oxford-IIIT Pets* são semelhantes aos encontrados nos *datasets* de classificação mencionados anteriormente nesta seção. Portanto, a métrica *Sparse Categorical Crossentropy* continuou a ser utilizada.

Em suma, a seleção apropriada de métricas para avaliar o desempenho dos modelos de aprendizagem profunda é crítica para fornecer uma demonstração adequada de seu desempenho e adaptabilidade a diferentes tarefas. Nos experimentos realizados neste estudo, optou-se por uma combinação de métricas apropriadas para tarefas de classificação e segmentação semântica com base nas especificações únicas atribuídas a cada tarefa. O uso rigoroso e bem planejado dessas métricas por meio da hipótese orientada aumentou a robustez e a credibilidade dos resultados obtidos. A lista completa de métricas utilizadas, assim como suas aplicações, é chave para observar uma ótima otimização dos parâmetros do modelo de aprendizagem e, assim, atingir uma interpretação ideal da tarefa variada que o modelo é treinado para resolver.

6.2.2 Parâmetros de Treinamento e Teste

Nessa Seção serão esmiuçados os detalhes dos parâmetros utilizados para o treinamento e teste das redes neurais. Esta investigação contemplará tanto os experimentos voltados à classificação, que empregaram a arquitetura da VGG-16 nos conjuntos de dados *Food-101* e *CIFAR 100*, quanto aqueles destinados à segmentação semântica. No caso da segmentação, exploramos as arquiteturas U-Net-Like e U-Net aplicadas no conjunto de dados *Oxford-IIIT Pets*.

Os parâmetros de treinamento e teste são de extrema importância no desenvolvimento de um modelo, uma vez que eles determinam o comportamento do aprendizado da rede. Aqui, buscaremos elucidar como a seleção destes parâmetros foi realizada para os diferentes cenários de nossa pesquisa. Deste modo, permitiremos que futuras pesquisas na área possam ser facilmente replicáveis ou até mesmo aprimoradas, através do ajuste destes parâmetros.

6.2.2.1 VGG-16 + *Food*-101

Na configuração da VGG-16 para o conjunto de dados *Food*-101, é relevante destacar a definição inicial das sementes (*seeds*) foram iguais à 77 para as bibliotecas TensorFlow e NumPy, valor o qual foi escolhido empiricamente. Essas sementes foram estabelecidas visando a reprodução consistente dos resultados nos testes, reduzindo variações indesejadas (Alahmari et al., 2020). Além disso, durante a fase de preparação dos dados, um parâmetro crucial foi o tamanho do lote (*batch size*), estabelecido em 64. Este parâmetro determina a quantidade de imagens usadas no treinamento durante uma única época (Kandel e Castelli, 2020). Por fim, entre as variáveis mantidas constantes em todos os procedimentos, destaca-se a taxa de aprendizado definida como $1e^{-5}$, considerada uma taxa mais conservadora, cuja importância é discutida na Seção 2.2.2.4.

Outros parâmetros relevantes incluem o número de épocas usadas para treinamento e teste do modelo, bem como as funções de otimização empregadas nesses processos. Adicionalmente, a estratégia de atualização dinâmica da taxa de aprendizado com CLR (Smith, 2017) é digna de menção. No entanto, devido à realização de um processo de ajuste fino (*fine-tuning*) nas arquiteturas da VGG-16, é importante ressaltar que esses parâmetros variavam de acordo com a fase da execução. Cada modelo foi treinado quatro vezes, como ilustrado na Figura 6.5 e mencionado na Seção 6.1.5. Portanto, as Tabelas 5, 6 e 7 apresentam os parâmetros utilizados em cada fase do processo de *fine-tuning* com o uso de BPCAPooling, *Max Pooling* e *Avg Pooling*, respectivamente.

Tabela 5: Parâmetros de treinamento e teste com VGG-16 e BPCAPooling no conjunto *Food-101*.

Parâmetros	Warm-up		Bloco 5		Bloco 4		Bloco 3	
	Valor	Uso	Valor	Uso	Valor	Uso	Valor	Uso
Épocas	700	-	700	-	700	-	700	-
Época de Parada	188	-	167	-	700	-	219	-
Parada Antecipada	-	Verdadeiro	-	Verdadeiro	-	Verdadeiro	-	Verdadeiro
CLR	-	Falso	-	Verdadeiro	-	Verdadeiro	-	Verdadeiro
Taxa de Aprendizado Mínima	0,0001	-	0,00001	-	0,00001	-	0,00001	-
Taxa de Aprendizado Máxima	0,0001	-	0,0001	-	0,0001	-	0,0001	-
Função de Otimização	Adam	-	SGD	-	SGD	-	SGD	-

Fonte: do próprio autor.

Tabela 6: Parâmetros de treinamento e teste com VGG-16 e *Max Pooling* no conjunto *Food-101*.

Parâmetros	Warm-up		Bloco 5		Bloco 4		Bloco 3	
	Valor	Uso	Valor	Uso	Valor	Uso	Valor	Uso
Épocas	700	-	700	-	700	-	700	-
Época de Parada	700	-	700	-	514	-	105	-
Parada Antecipada	-	Falso	-	Falso	-	Verdadeiro	-	Verdadeiro
CLR	-	Falso	-	Verdadeiro	-	Verdadeiro	-	Verdadeiro
Taxa de Aprendizado Mínima	0.0001	-	0.00001	-	0.00001	-	0.00001	-
Taxa de Aprendizado Máxima	0.0001	-	0.0001	-	0.0001	-	0.0001	-
Função de Otimização	Adam	-	SGD	-	SGD	-	SGD	-

Fonte: do próprio autor.

Tabela 7: Parâmetros de treinamento e teste com VGG-16 e *Avg Pooling* no conjunto *Food-101*.

Parâmetros	Warm-up		Bloco 5		Bloco 4		Bloco 3	
	Valor	Uso	Valor	Uso	Valor	Uso	Valor	Uso
Épocas	700	-	700	-	700	-	700	-
Época de Parada	700	-	700	-	700	-	249	-
Parada Antecipada	-	Falso	-	Falso	-	Verdadeiro	-	Verdadeiro
CLR	-	Falso	-	Verdadeiro	-	Verdadeiro	-	Verdadeiro
Taxa de Aprendizado Mínima	0,0001	-	0,00001	-	0,00001	-	0,00001	-
Taxa de Aprendizado Máxima	0,0001	-	0,0001	-	0,0001	-	0,0001	-
Função de Otimização	Adam	-	SGD	-	SGD	-	SGD	-

Fonte: do próprio autor.

Analisando as Tabelas 5, 6 e 7, observa-se que o parâmetro que mais se destaca é a Época de Parada, o qual é influenciado pela técnica de Parada Antecipada (*Early Stopping*). Ademais, as variações entre as configurações se dão pelo emprego ou não da taxa de aprendizado dinâmica utilizando o método CLR (*Cyclical Learning Rate*). Essas discrepâncias refletem diferentes estratégias adotadas no treinamento dos modelos.

6.2.2.2 VGG-16 + CIFAR 100

A abordagem aplicada na combinação da VGG-16 com o conjunto de dados CIFAR-100 segue um padrão semelhante à utilizada na combinação da VGG-16 com o conjunto de dados *Food-101*, com relação ao uso das técnicas. Dentre os parâmetros mantidos constantes durante o processo, destaca-se a taxa de aprendizado máxima, o tamanho do lote (*batch size*) e o número de épocas, estabelecidos em $1e^{-5}$, 64, e 700, respectivamente. Quanto às outras variáveis, os valores específicos podem ser visualizados detalhadamente nas Tabelas 8, 9 e 10, para as configurações de BPCAPooling, *Max Pooling* e *Avg Pooling*, respectivamente.

Tabela 8: Parâmetros de treinamento e teste com VGG-16 e BPCAPooling no conjunto CIFAR 100.

Parâmetros	Warm-up		Bloco 5		Bloco 4		Bloco 3	
	Valor	Uso	Valor	Uso	Valor	Uso	Valor	Uso
Épocas	700	-	700	-	700	-	700	-
Época de Parada	700	-	700	-	473	-	700	-
Parada Antecipada	-	Falso	-	Falso	-	Verdadeiro	-	Verdadeiro
CLR	-	Falso	-	Verdadeiro	-	Verdadeiro	-	Verdadeiro
Taxa de Aprendizado Mínima	0,0001	-	0,00001	-	0,00001	-	0,00001	-
Taxa de Aprendizado Máxima	0,0001	-	0,0001	-	0,0001	-	0,0001	-
Função de Otimização	Adam	-	SGD	-	SGD	-	SGD	-

Fonte: do próprio autor.

Tabela 9: Parâmetros de treinamento e teste com VGG-16 e *Max Pooling* no conjunto CIFAR 100.

Parâmetros	Warm-up		Bloco 5		Bloco 4		Bloco 3	
	Valor	Uso	Valor	Uso	Valor	Uso	Valor	Uso
Épocas	700	-	700	-	700	-	700	-
Época de Parada	700	-	700	-	700	-	550	-
Parada Antecipada	-	Falso	-	Falso	-	Verdadeiro	-	Verdadeiro
CLR	-	Falso	-	Verdadeiro	-	Verdadeiro	-	Verdadeiro
Taxa de Aprendizado Mínima	0,0001	-	0,00001	-	0,00001	-	0,00001	-
Taxa de Aprendizado Máxima	0,0001	-	0,0001	-	0,0001	-	0,0001	-
Função de Otimização	Adam	-	SGD	-	SGD	-	SGD	-

Fonte: do próprio autor.

Tabela 10: Parâmetros de treinamento e teste com VGG-16 e *Avg Pooling* no conjunto CIFAR 100.

Parâmetros	Warm-up		Bloco 5		Bloco 4		Bloco 3	
	Valor	Uso	Valor	Uso	Valor	Uso	Valor	Uso
Épocas	700	-	700	-	700	-	700	-
Época de Parada	700	-	700	-	700	-	350	-
Parada Antecipada	-	Falso	-	Falso	-	Falso	-	Verdadeiro
CLR	-	Falso	-	Verdadeiro	-	Verdadeiro	-	Verdadeiro
Taxa de Aprendizado Mínima	0,0001	-	0,00001	-	0,00001	-	0,00001	-
Taxa de Aprendizado Máxima	0,0001	-	0,0001	-	0,0001	-	0,0001	-
Função de Otimização	Adam	-	SGD	-	SGD	-	SGD	-

Fonte: do próprio autor.

6.2.2.3 U-Net-Like + Oxford-IIIT Pets

Na configuração dos parâmetros para a arquitetura U-Net-Like aplicada ao conjunto de dados *Oxford-IIIT Pets*, é importante observar que não foi realizado um processo de *fine-tuning*, diferentemente das arquiteturas utilizadas para classificação que foram mencionadas anteriormente. No entanto, muitos dos parâmetros se mantiveram consistentes. Neste contexto, foram adotadas sementes para as bibliotecas TensorFlow e Numpy, com um valor empírico de 77, garantindo a reprodutibilidade dos testes.

Para a U-Net-Like, foram realizados treinamentos e testes ao longo de 20 e 500 épocas, respectivamente, antecedendo os experimentos relacionados à arquitetura U-Net, a qual será dis-

cutida na Seção 6.2.2.4. Em relação à taxa de aprendizado máxima, vale ressaltar que seu valor foi mantido constante em 1×10^{-4} para todos os experimentos realizados. Apesar de parecer uma taxa mais elevada em comparação com as anteriores, está dentro do limite recomendado quando comparado com outras pesquisas (Fajar et al., 2023). Além disso, a taxa de aprendizado mínima foi definida como 1×10^{-5} , sendo reduzida quando o modelo não apresentou evolução na métrica de avaliação principal após 25 épocas para os testes com 500 épocas, pois para os de 20 épocas a redução não foi utilizada. Em relação ao valor do *batch size*, esse se manteve constante em 64, seguindo o padrão dos experimentos anteriores com a rede VGG-16.

Em contraste com os experimentos anteriores, foram utilizadas fórmulas para calcular a quantidade de iterações por época, conforme representado pela Equação 6.10, e para calcular a quantidade das iterações de validação, representadas pela Equação 6.11, descritas respectivamente por:

$$\text{Passos por Época} = \frac{\text{tamanho do conjunto de treino}}{\text{tamanho do lote}} \quad (6.10)$$

e

$$\text{Passos de Validação} = \frac{\text{tamanho do conjunto de teste}}{\text{tamanho do lote}} \cdot \frac{1}{\text{divisões de validação}}, \quad (6.11)$$

onde as divisões de validação foram definidas manualmente, sendo que essas divisões de validação foram definidas manualmente com o valor de 5, representando a quantidade de partes em que o conjunto de validação foi dividido para avaliações separadas durante o treinamento. Esse método permite uma validação cruzada interna durante o processo de treinamento, o que contribui para uma avaliação mais robusta do modelo ao longo das épocas. Essa abordagem de validação cruzada ajuda a monitorar a performance e o ajuste do modelo, oferecendo *insights* valiosos sobre o seu desempenho geral. Os valores obtidos foram de 57 passos por época e 9 passos de validação.

Por fim, é importante ressaltar que o otimizador Adam foi utilizado de forma unânime em todos os testes realizados. Essa escolha se justifica pelo fato de que, sem a aplicação do processo de *fine-tuning*, a variação da função de otimização não é habitualmente realizada. Dessa maneira, optou-se por manter o otimizador Adam constante em todos os experimentos, visando uma maior consistência nos resultados obtidos.

6.2.2.4 U-Net + *Oxford-IIIT Pets*

No último experimento realizado, a arquitetura da U-Net foi combinada com os métodos de *pooling* BPCAPooling e *Max Pooling* em um treinamento de 500 épocas. É importante mencionar que a U-Net é considerada uma rede mais robusta, possuindo um número significativo de parâmetros (vide Seção 6.1.7), o que impacta diretamente no tempo e nos recursos computacionais necessários para o treinamento e teste.

Assim como nos experimentos anteriores com a arquitetura U-Net-Like, o otimizador Adam foi utilizado de forma constante. A taxa de aprendizado mínima foi mantida em 1×10^{-5} , enquanto a taxa de aprendizado máxima foi definida como 1×10^{-4} . A estratégia de redução da taxa de aprendizado foi aplicada após 25 épocas sem melhora na métrica principal.

O critério de parada antecipada foi mantido constante em todos os experimentos realizados. A utilização das Equações 6.10 e 6.11 também foi aplicada nos experimentos da U-Net, resultando em 18 passos de validação e 115 passos por época, visto que as divisões de validação mantiveram-se no valor de 5.

Por fim, é relevante destacar que o valor da semente permaneceu empiricamente em 77 para garantir a reprodutibilidade dos resultados. No entanto, o tamanho do lote teve que ser reduzido para 32, uma vez que o *hardware*, apesar de possuir 24.564 MiB, não suportava o uso do parâmetro igual a 64, como feito nos demais testes.

6.2.3 Explicabilidade de Modelos

No contexto da explicação de modelos, abordaremos o uso de ferramentas pertencentes à área de Inteligência Artificial Explicável, também conhecida como *Explainable Artificial Intelligence* (XAI) (Gunning et al., 2019). Essa área visa mitigar as limitações dos modelos de “caixa-preta”, tornando-os mais compreensíveis para os humanos, elucidando seu funcionamento de maneira mais transparente e interpretável (Angelov et al., 2021).

No decorrer dos experimentos com os modelos de classificação, aplicou-se o LIME, acrônimo para *Local Interpretable Model-Agnostic Explanations*, uma técnica desenvolvida por Ribeiro et al. (2016). O LIME oportuniza entender as previsões de classificadores de uma maneira intuitiva, ao identificar as regiões críticas de uma imagem que mais contribuem para a decisão do modelo.

O mecanismo do LIME baseia-se em aproximar o modelo original, em um determinado ponto de interesse, com um modelo linear interpretável localmente. Este processo é condu-

zido por meio da geração de novas amostras com perturbações ao redor do ponto de interesse, classificando-as com o uso do modelo original, e então ajustando um modelo linear com base na saída (Ribeiro et al., 2016). As características mais relevantes, aquelas que contribuem mais para a tomada de decisão, são destacadas na imagem, proporcionando um entendimento de qual influência cada parte da imagem tem na decisão final do modelo.

Ao se empregar essa perspectiva oferecida pelo LIME, observam-se as áreas de maior influência na classificação ressaltadas na imagem, tornando possível averiguar se a decisão do modelo é impactada pelo contexto da imagem, como o fundo, ou por elementos não pertinentes à classificação correta. Um exemplo ilustrativo dessa situação é demonstrado na Figura 6.12. Neste caso, o modelo confundiu um cachorro Husky com um lobo, dado que o fundo nevado no contexto da imagem é um ambiente comumente associado a fotos de lobos. Esta análise, viabilizada pela interpretabilidade proporcionada pelo LIME, auxilia em evidenciar e entender as falhas do modelo.

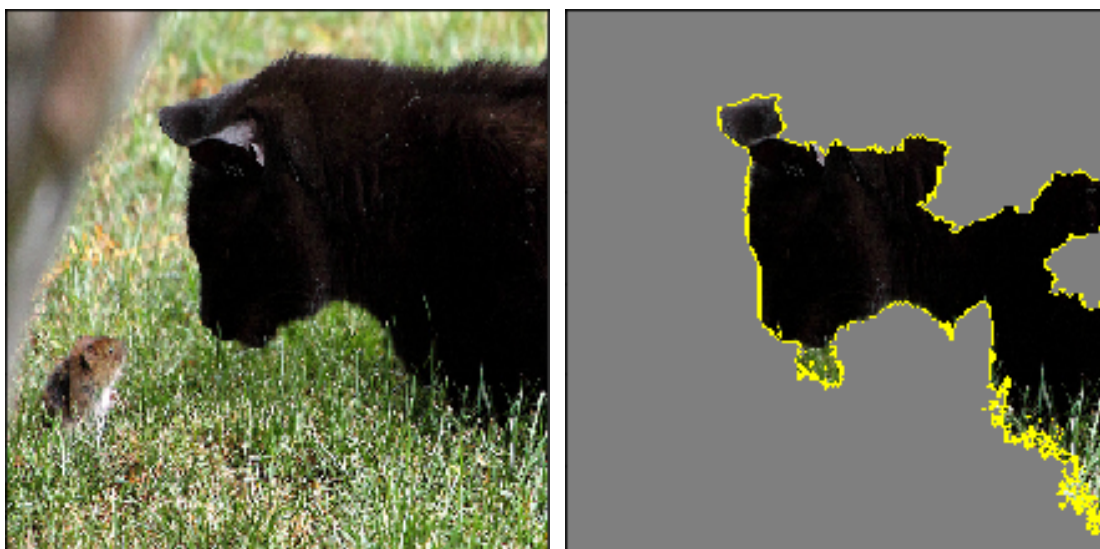
Figura 6.12: Exemplo de saída gerado pelo LIME em situação de erro.



Fonte: retirado e adaptado de Ribeiro et al. (2016).

Idealmente, o LIME destaca partes da imagem diretamente associadas às regiões do objeto que influenciam a inferência do modelo, oferecendo informações cruciais sobre o processo de tomada de decisão do modelo, conforme ilustrado na Figura 6.13.

Figura 6.13: Exemplo de saída gerado pelo LIME em situação correta.



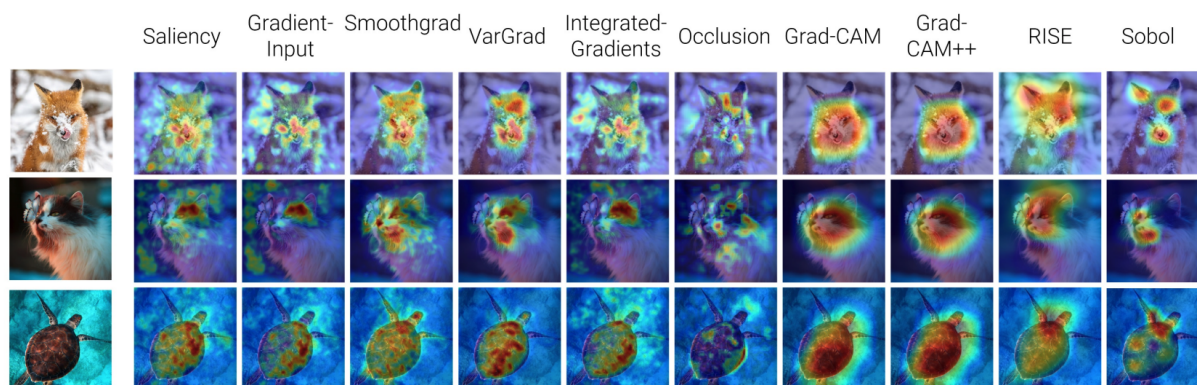
(a) Classificação correta do “Gato” como “gato”.

(b) Destaque na explicação da predição.

Fonte: retirado e adaptado de [Ribeiro et al. \(2016\)](#).

Contudo, ao tentar aplicar os conceitos de *Explainable AI* (XAI) com o uso do LIME em problemas de segmentação semântica, não se obteve sucesso, uma vez que todos os *pixels* nessas segmentações representam áreas de interesse. Diante dessa limitação, recorreu-se ao *framework* Xplique ([Fel et al., 2022](#)), que, dentre suas funcionalidades, utiliza da aplicação de métodos de atribuição destinados a produzir mapas de calor (*heatmaps*) com o intuito de explicar as áreas de foco dos modelos em relação às instâncias segmentadas ([Fel et al., 2022](#)). Essa abordagem assemelha-se ao propósito do LIME e a aplicação dos métodos de atribuição evidenciando áreas de maior influência mapeadas pelo modelo, por meio de mapas de calor, podem ser visualizados na Figura [6.14](#).

Figura 6.14: Exemplo de aplicação de métodos de atribuição da ferramenta Xplique.



Fonte: retirado e adaptado de [Fel et al. \(2022\)](#).

Por fim, é evidente que a incorporação de ferramentas de Inteligência Artificial Explicável (XAI) proporciona *insights* cruciais sobre a dinâmica interna dos modelos de aprendizagem profunda. A capacidade de revelar as regiões da imagem de entrada que mais influenciam as previsões do modelo, viabilizada pelo uso do LIME e do Xplique, oferece uma visão mais detalhada do processo interpretativo dos modelos. Assim, a adoção dessas abordagens explanatórias não apenas promove a confiança nos modelos de Aprendizado Profundo, mas também aprimora sua compreensibilidade, aspectos essenciais tanto em aplicações do mundo real quanto em propostas de pesquisas futuras.

As descobertas e desafios enfrentados durante a utilização dessas técnicas serão minuciosamente explorados no Capítulo 7, o qual se dedica a apresentar os resultados dos experimentos em profundidade.

6.3 Considerações Finais do Capítulo

Este capítulo abordou várias facetas do projeto, proporcionando uma compreensão aprofundada da metodologia adotada e dos procedimentos experimentais realizados. Inicialmente, a seção de materiais e métodos ofereceu uma descrição detalhada das tecnologias e do hardware que foram necessários para implementar e executar as arquiteturas de redes neurais. Apresentou também o conjunto de dados utilizados, o conceito de transferência de aprendizagem e o processo de aumento de dados nos problemas explorados pelo projeto, com o intuito de evidenciar os benefícios do método proposto (BPCAPooling) e as alterações resultantes na camada de *pooling* em diferentes arquiteturas de redes. Precisamente, todas essas descrições foram cruciais

para estabelecer o contexto adequado para a pesquisa realizada.

A Metodologia Experimental, por sua vez, aprofundou-se na seleção das métricas para avaliar o desempenho dos modelos, com uma distinção clara entre as métricas associadas a problemas de classificação e aquelas relacionadas a desafios de segmentação semântica. Foi elucidado, também, sobre as configurações dos parâmetros de treinamento e de teste. Outrossim, esta seção enfocou na explicabilidade dos modelos, apresentado o uso de ferramentas do âmbito da Inteligência Artificial Explicável (XAI) para tornar os modelos de aprendizado profundo mais transparentes.

Com isso, espera-se que este capítulo tenha fornecido uma visão clara e completa da metodologia adotada nesta pesquisa, propiciando assim um arcabouço sólido para as análises subsequentes dos resultados experimentais obtidos, que serão detalhados no próximo capítulo.

7 RESULTADOS E DISCUSSÕES

Neste capítulo, dedicado a Resultados e Discussões, serão apresentados e analisados os resultados dos experimentos realizados no transcurso deste estudo, acompanhados de uma reflexão criteriosa sobre os dados obtidos. Nesta etapa, será possível, portanto, avaliar a eficácia dos métodos que foram propostos e implementados, além de discutir as ideias futuras de implementação e as perspectivas coletadas para a evolução do tema em questão.

A estrutura do capítulo é delineada em seções que facilitam a compreensão dos resultados de acordo com o contexto do experimento aplicado. Na Seção 7.1, será discutido sobre os experimentos realizados prioritariamente para a tarefa de classificação. Esses experimentos iniciais foram fundamentais para balizar as análises e refinamentos subsequentes da técnica proposta.

Na sequência, a Seção 7.2 irá abordar os resultados obtidos nos experimentos que se destinam diretamente à área de segmentação, que constitui o núcleo da aplicação da técnica proposta. Aqui, expandiremos o foco para a parcela mais crítica de nossa pesquisa, apresentando a eficácia de nosso método na tarefa de segmentação semântica de imagens.

Por fim, na Seção 7.3 proporcionaremos as considerações finais em relação as seções anteriores, salientando às implicações dos resultados no campo de estudo do trabalho.

7.1 Resultados da Classificação de Imagens

Os experimentos e resultados desta seção fornecem valiosas percepções sobre o desempenho do método BPCAPooling em comparação com as estratégias de *pooling* convencionais (*Avg Pooling* e *Max Pooling*) (Özdemir, 2023) quando aplicados na arquitetura VGG-16 para a tarefa de classificação.

Ao analisar o desempenho do modelo treinado no conjunto de dados CIFAR 100 utilizando o método proposto, observamos uma tendência indesejada. Este modelo apresentou a maior perda e a menor precisão quando comparado aos modelos utilizando os métodos *Avg Pooling* e *Max Pooling* no processo de descongelamento do terceiro bloco convolucional. Os resultados são evidenciados na parte de aquecimento do processo de *fine-tuning* na Tabela 11.

Mesmo após a conclusão do processo de *fine-tuning* completo, o modelo treinado com a técnica BPCAPooling apresentou resultados inferiores, conforme evidenciado na Tabela 11. Os métodos *Max Pooling* e *Avg Pooling* continuaram a demonstrar uma superioridade significativa

em relação ao método BPCAPooling. Os resultados mais promissores foram obtidos pela arquitetura VGG-16 com o método Max Pooling. A análise detalhada dos motivos subjacentes a essa discrepância será discutida na Seção 7.1.1.

Tabela 11: Resultados por fase de *fine-tuning* de VGG-16 aplicada no conjunto de dados CIFAR 100.

Método de <i>Pooling</i>	Aquecimento		Bloco 5		Bloco 4		Bloco 3	
	Acurácia(%)	Loss	Acurácia(%)	Loss	Acurácia(%)	Loss	Acurácia(%)	Loss
<i>Avg Pooling</i>	36,300	2,7309	39,340	2,4971	41,371	2,2958	42,371	2,2364
BPCAPooling	14,471	3,7645	15,185	3,6677	16,229	3,6020	17,543	3,5121
<i>Max Pooling</i>	39,857	2,0481	43,671	1,7166	45,100	1,5448	46,071	1,4862

Fonte: do próprio autor.

Para o conjunto de dados *Food-101*, os resultados não divergiram muito dos obtidos no primeiro conjunto de dados. Durante a fase inicial do treinamento, os resultados do método proposto já se mostraram desafiadores em comparação aos outros métodos, como pode ser observado na parte de aquecimento da Tabela 12.

No contexto da aplicação do processo de *fine-tuning*, observa-se consistentemente que o modelo de *pooling* proposto demonstrou desempenho inferior em todas as fases quando comparado ao melhor método, que continuou sendo a junção de VGG-16 com *Max Pooling*, conforme ilustrado na Tabela 12.

Tabela 12: Resultados por fase de *fine-tuning* de VGG-16 aplicada no conjunto de dados *Food-101*.

Método de <i>Pooling</i>	Aquecimento		Bloco 5		Bloco 4		Bloco 3	
	Acurácia(%)	Loss	Acurácia(%)	Loss	Acurácia(%)	Loss	Acurácia(%)	Loss
BPCAPooling	3,8586	4,7057	4,3395	4,6632	4,4130	4,4404	3,423	4,5187
Max Pooling	41,623	2,3718	49,471	2,2033	50,080	2,3745	50,031	2,4335

Fonte: do próprio autor.

Na Tabela 12, observa-se um comportamento distinto em comparação ao primeiro conjunto de dados. Os resultados de maior acurácia estão representados no quarto bloco, não no último bloco adicionado, o que levanta questionamentos sobre a influência desse bloco, assunto que será discutido na Seção 7.1.1.

Em relação ao custo computacional, observou-se que o método proposto, desenvolvido utilizando o *framework* Tensorflow, exigiu um processamento considerável, resultando em desempenho mais lento durante os testes devido ao aumento da complexidade. Por exemplo, no caso do conjunto de dados CIFAR 100, com o descongelamento de blocos e 700 épocas utilizando a configuração de *hardware* de referência, os métodos *Max Pooling* e *Avg Pooling* levaram cerca de 7 horas para serem executados. Em contrapartida, o BPCAPooling exigiu aproximadamente 19 horas, cerca de 2,71 vezes mais tempo, ressaltando que o conjunto de dados CIFAR 100 é otimizado para processamento em GPU.

Resultados semelhantes foram observados no conjunto de dados *Food-101*, onde *Max Pooling* e *Avg Pooling* levaram cerca de 33 horas para o processo de treinamento e validação, enquanto o BPCAPooling demandou aproximadamente 38 horas. O aumento no tempo de processamento se deve principalmente ao tamanho maior das imagens. É importante mencionar que este conjunto de dados não está otimizado para processamento em GPU, o que resulta em um desempenho do *hardware* abaixo do ideal.

Apesar dos resultados inesperados em relação ao método BPCAPooling, alguns *insights* puderam ser extraídos das fases de treinamento e validação. Esses pontos de introspecção serão abordados a seguir nesta seção, juntamente com informações sobre a influência da aplicação desse método nas características específicas dos diferentes conjuntos de dados (Seção 7.1.1) e a exploração da interpretabilidade das classificações utilizando a abordagem LIME e conceitos de preservação de espacialidade (Seção 7.1.2).

7.1.1 Diferença de conjuntos de dados

A utilização de múltiplos conjuntos de dados foi fundamental para o presente estudo, uma vez que a diversidade de informações teve um impacto significativo na capacidade da rede VGG-16 em capturar detalhes e características. Esse comportamento está diretamente alinhado com a motivação subjacente ao uso das Redes Neurais Convolucionais (CNNs), como discutido no Capítulo 3. A Figura 7.1 ilustra a diferença de resolução entre amostras dos conjuntos de dados CIFAR 100 e *Food-101*, proporcionando uma melhor compreensão das disparidades entre eles.

Figura 7.1: Exemplo do conjunto de dados CIFAR-100 à esquerda e *Food*-101 à direita, em escala real.



Fonte: do próprio autor.

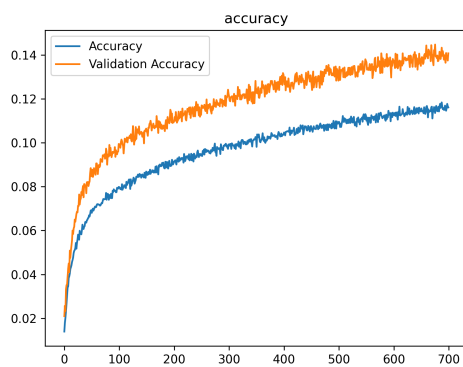
Adicionalmente, é relevante mencionar que em conjuntos de dados com amostras pequenas, como é o caso do CIFAR-100, a quantidade de informação a ser preservada é mínima. Isso pode explicar o desempenho inferior do método BPCAPooling em comparação com os métodos de *pooling* tradicionais, como *Max Pooling* e *Avg Pooling*. Outra possível explicação para o resultado inferior do uso do conjunto de dados *Food*-101 em relação ao CIFAR-100 é que, com o CIFAR-100 possuindo amostras de menor resolução, a variabilidade dos dados também é menor. Isso permite que o BPCAPooling tenha uma representação mais eficiente de blocos. No entanto, no conjunto de dados *Food*-101, que apresenta uma variabilidade mais alta nas configurações de blocos, o BPCAPooling pode não preservar informações suficientes no mapa de características ao representar com o uso de um único componente.

Além disso, conforme discutido na Seção 7.1, ao analisar o processo de *fine-tuning* utilizando o método BPCAPooling, destaca-se um comportamento notável no quarto bloco convolucional, onde ocorre uma tendência ascendente na acurácia e uma tendência de redução na

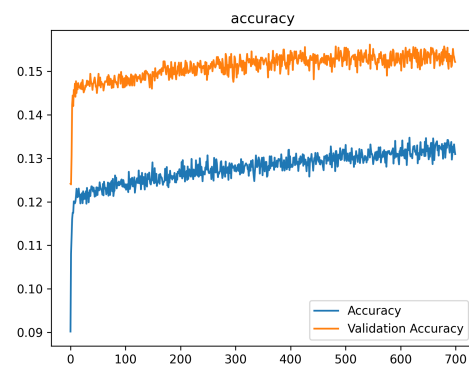
loss ao longo das épocas de treinamento e validação. Essa tendência sugere que características espaciais importantes provavelmente estão sendo capturadas e retidas no quarto bloco convolucional. É notável que o modelo não apresenta sinais de *overfitting* nesse bloco, indicando que mais épocas poderiam ser empregadas nessa etapa do processo para explorar melhor essa tendência ascendente. Contudo, é crucial considerar que a condução de épocas adicionais demandaria recursos computacionais significativos, devido à complexidade do método e ao custo computacional associado.

A representação da acurácia e *loss* para o conjunto de dados CIFAR 100 está ilustrada nas Figuras 7.2 e 7.3, respectivamente, enquanto as Figuras 7.4 e 7.5 mostram a evolução da acurácia e *loss* para o conjunto de dados *Food*-101, todas referentes ao uso do método BPCAPooling.

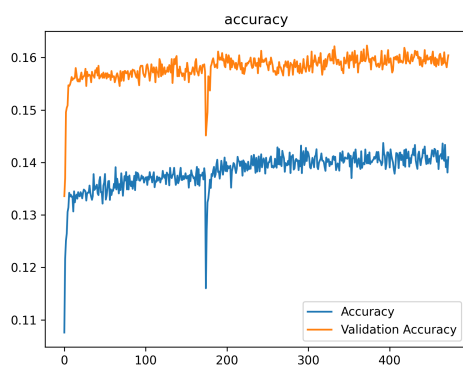
Figura 7.2: Evolução de Acurácia no conjunto de dados CIFAR 100.



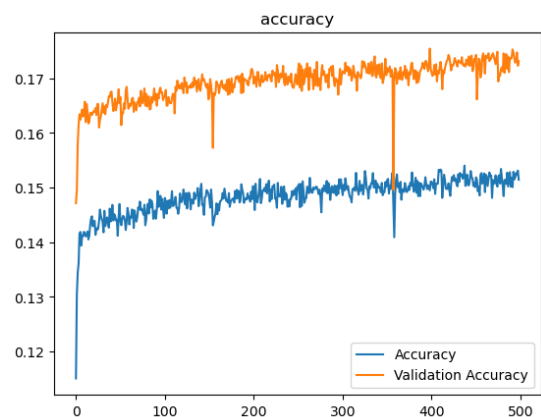
(a) Aquecimento.



(b) Bloco 5.



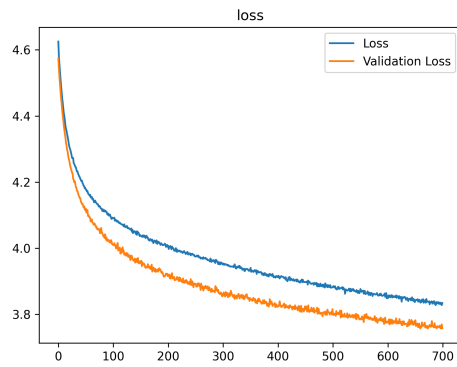
(c) Bloco 4.



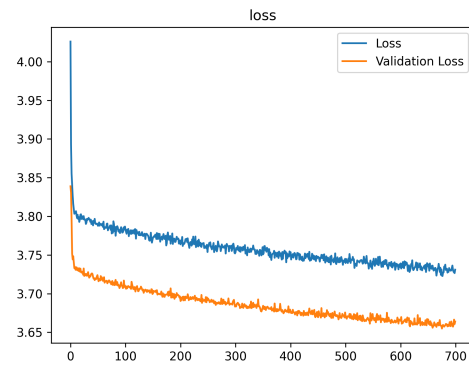
(d) Bloco 3.

Fonte: do próprio autor.

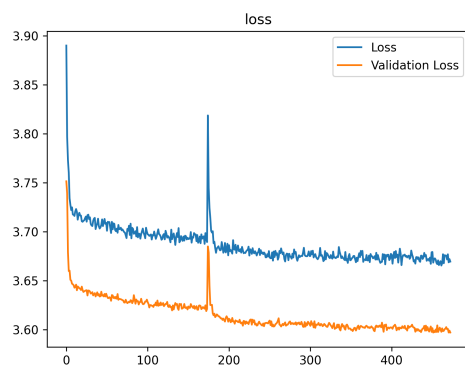
Figura 7.3: Evolução de *Loss* no conjunto de dados CIFAR 100.



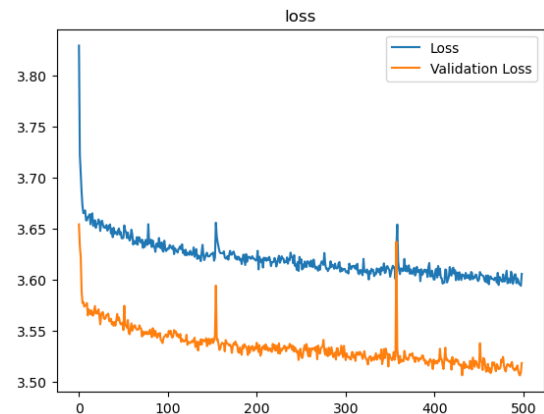
(a) Aquecimento.



(b) Bloco 5.



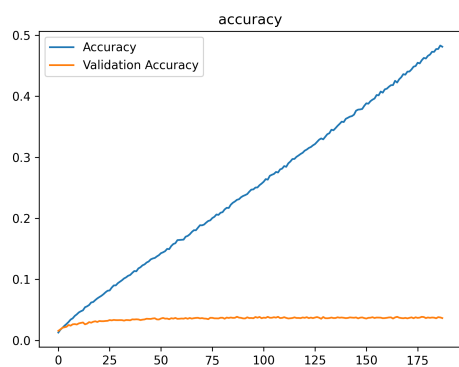
(c) Bloco 4.



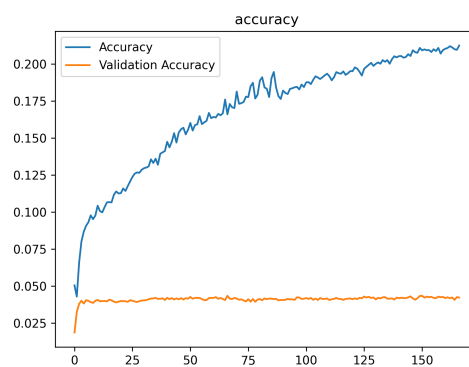
(d) Bloco 3.

Fonte: do próprio autor.

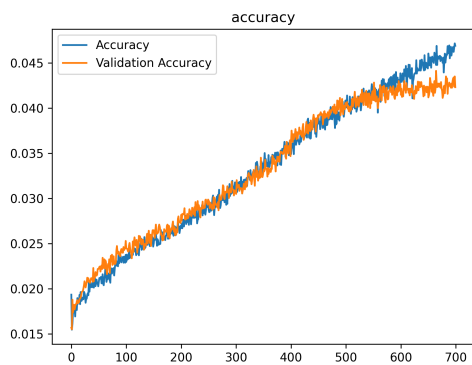
Figura 7.4: Evolução de Acurácia no conjunto de dados *Food-101*.



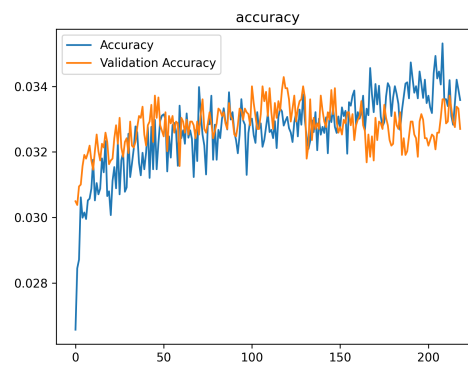
(a) Aquecimento.



(b) Bloco 5.



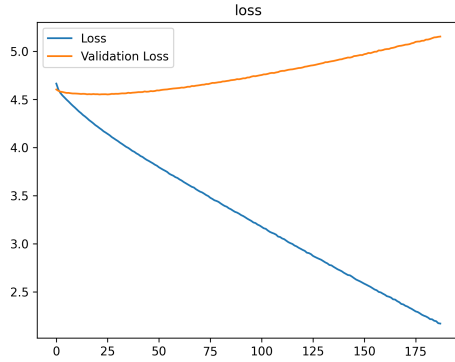
(c) Bloco 4.



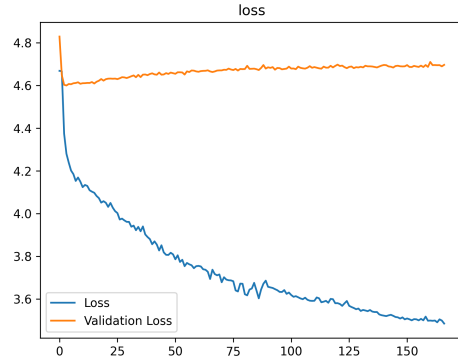
(d) Bloco 3.

Fonte: do próprio autor.

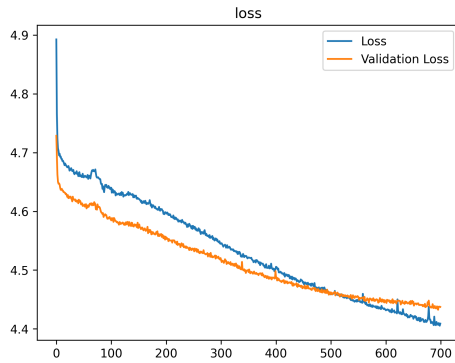
Figura 7.5: Evolução da *Loss* no conjunto de dados *Food-101*.



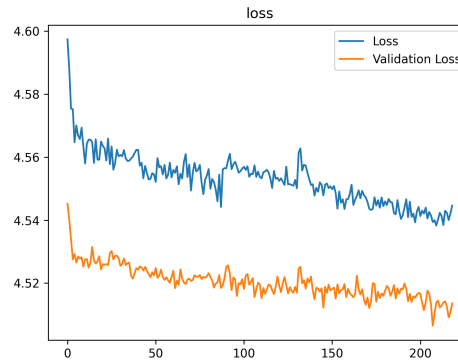
(a) Aquecimento.



(b) Bloco 5.



(c) Bloco 4.



(d) Bloco 3.

Fonte: do próprio autor.

Em suma, a diversidade dos conjuntos de dados utilizados neste estudo forneceu uma avaliação do método BPCAPooling, destacando pontos de atenção e a necessidade de otimizações específicas para diferentes cenários. As diferenças nos resultados reforçam a importância da seleção apropriada do conjunto de dados considerando suas características particulares.

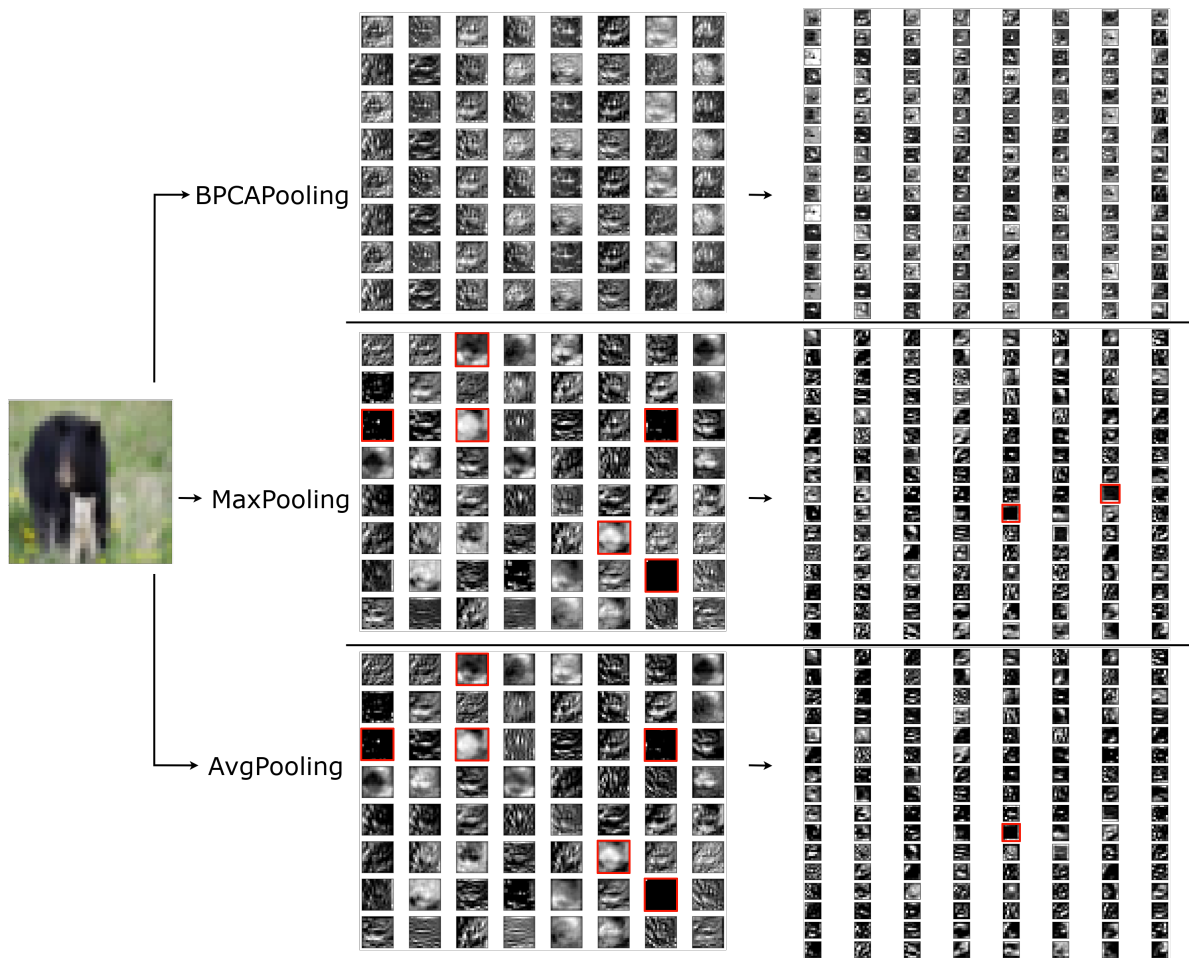
7.1.2 LIME e Preservação de Espacialidade

Em termos de preservação das informações espaciais entre as camadas de *pooling*, o método BPCAPooling se destacou em relação a outros métodos quando analisado em camadas de *pooling* de alto nível em ambos os conjuntos de dados (3ª camada).

No conjunto de dados CIFAR 100, é visível que as características geométricas das imagens são mantidas ao usar o BPCAPooling. Em contraste com os métodos convencionais, não se observam imagens completamente escurecidas ou com ruído nas camadas de mais baixo nível, como ilustrado na Figura 7.6 e indicado em destaque em vermelho. Apesar do tamanho

menor da imagem no conjunto CIFAR 100, resultando em mapas de características de dimensão 2×2 nas camadas posteriores, o que torna a preservação espacial um desafio, optou-se por manter a arquitetura VGG-16 original sem alterações estruturais, em vez de remover os blocos convolucionais ulteriores.

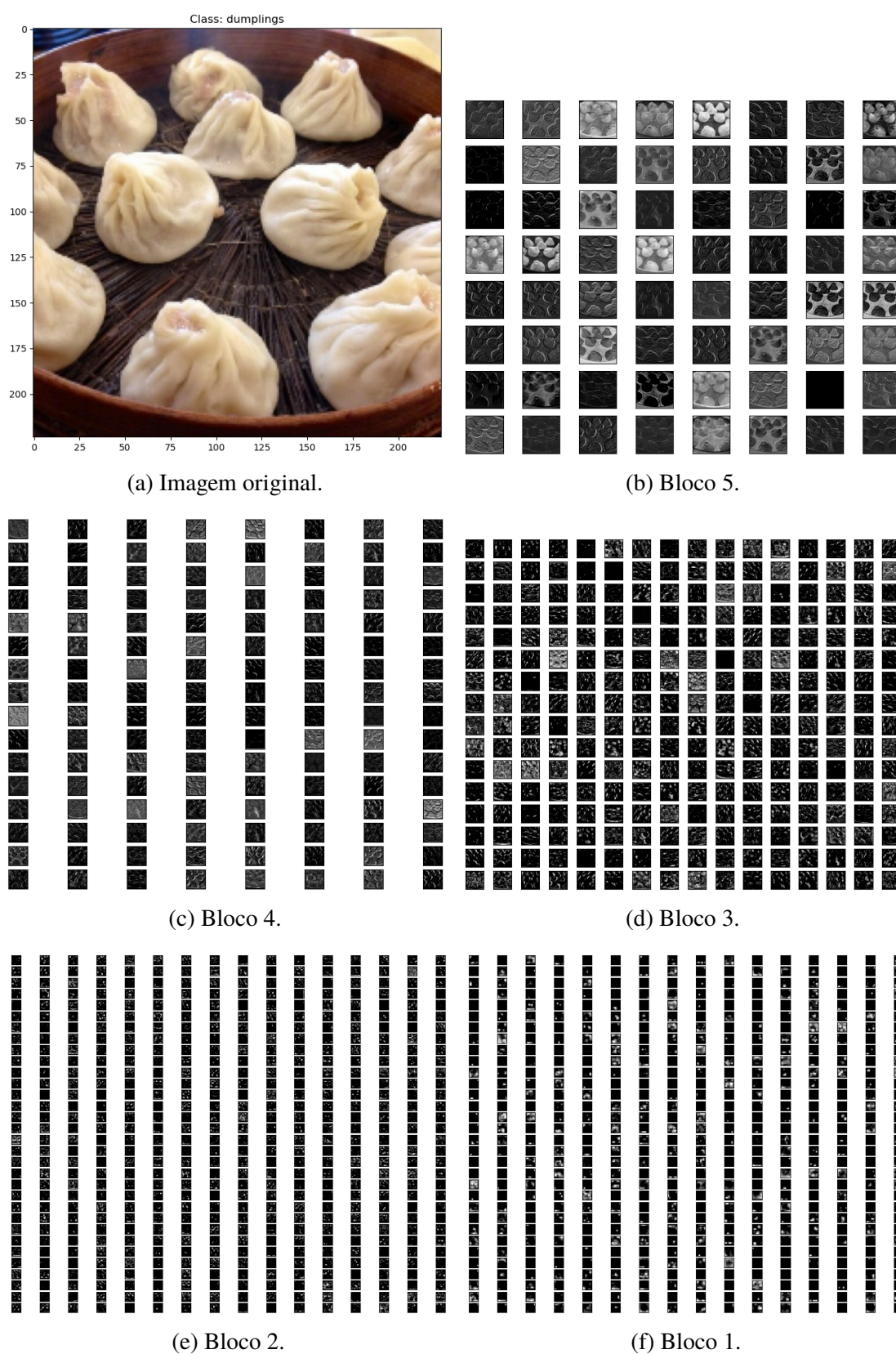
Figura 7.6: Representação com primeira e segunda camada de *pooling* aplicada no conjunto de dados CIFAR-100.



Fonte: do próprio autor.

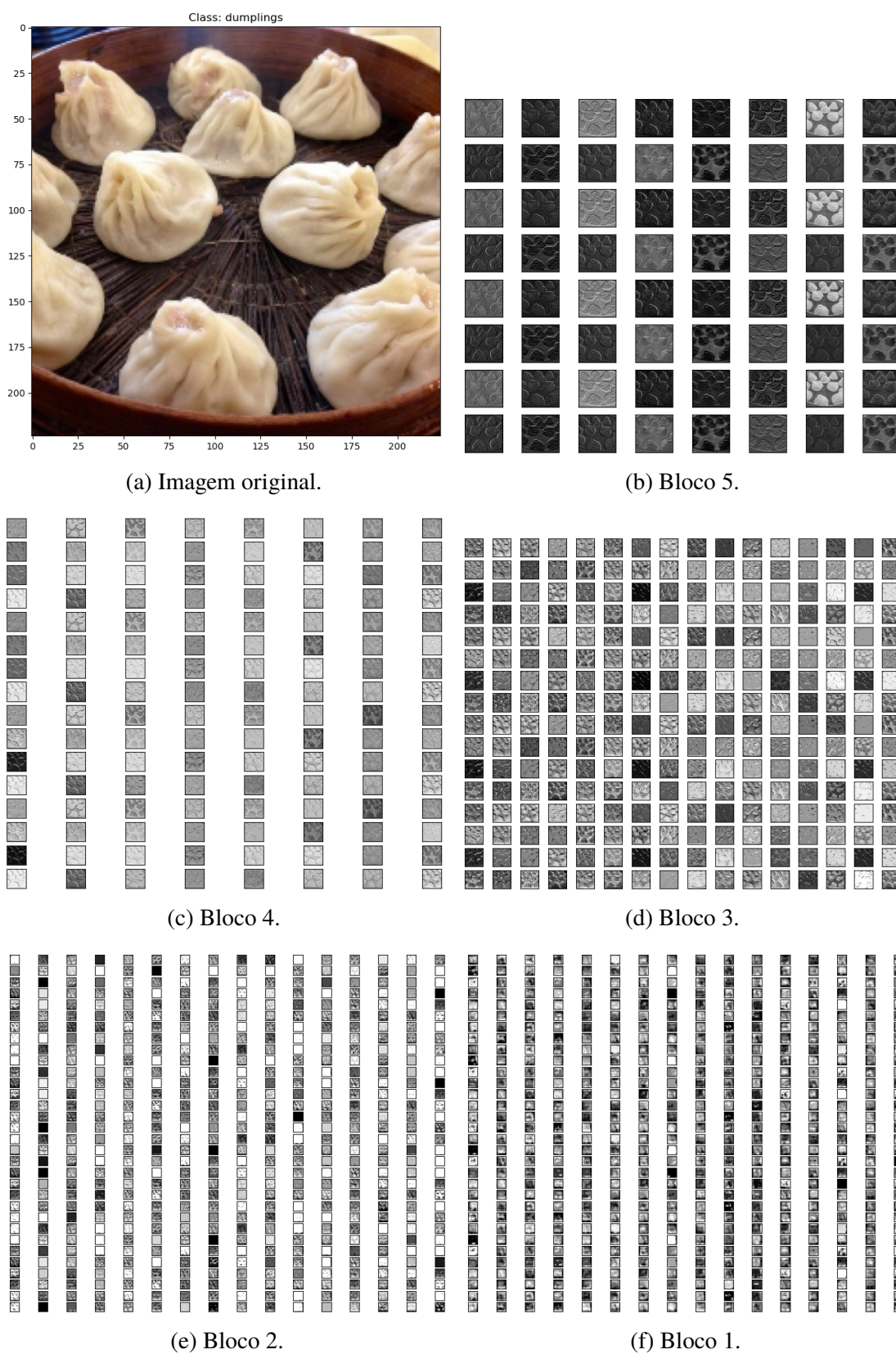
Ao analisar as camadas de *pooling* aplicadas ao conjunto de dados *Food-101*, representadas nas Figuras 7.7 e 7.8, observa-se diferenças nas intensidades das saídas dessas camadas. As saídas resultantes do método *Max Pooling* apresentam menor intensidade, enquanto as saídas do método BPCAPooling exibem maior intensidade, resultando em imagens com maior clareza e definição, principalmente de bordas.

Figura 7.7: Resultado visual de camadas de *pooling* do conjunto *Food-101*. Camadas com *Max Pooling*.



Fonte: do próprio autor.

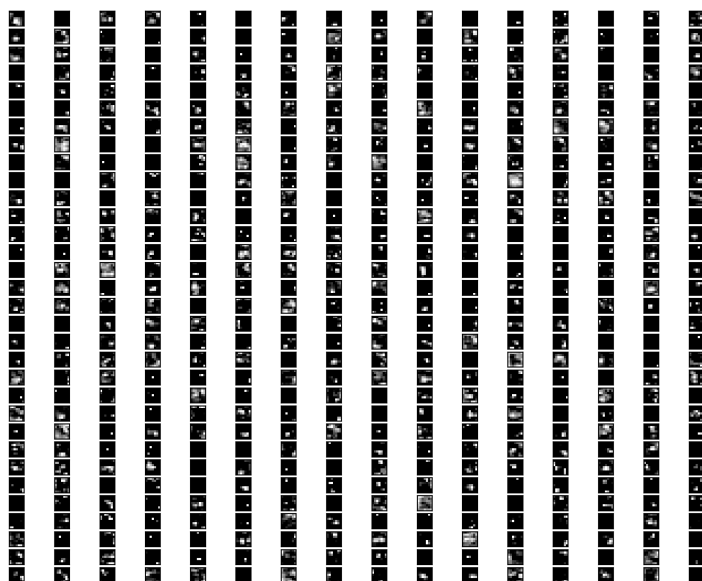
Figura 7.8: Resultado visual de camadas de *pooling* do conjunto *Food-101*. Camadas com BPCAPooling.



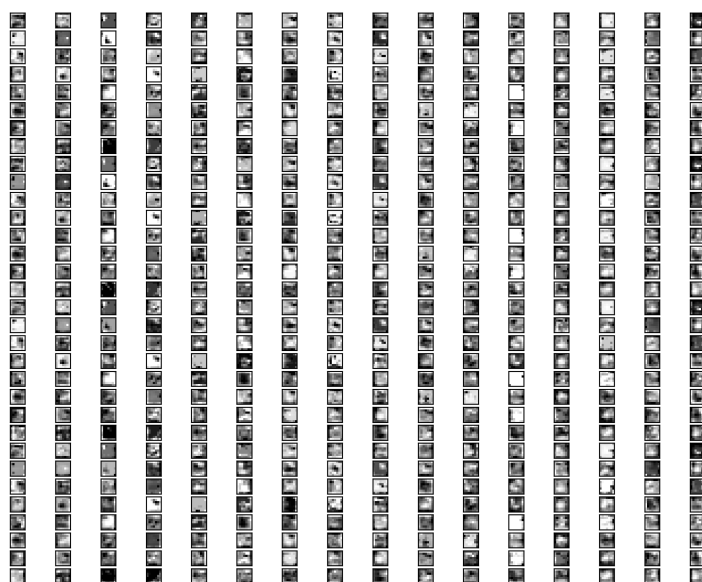
Fonte: do próprio autor.

Outro ponto é a discrepância nas últimas camadas de BPCAPooling, como mencionado anteriormente no contexto do conjunto de dados CIFAR 100. As saídas dessas camadas finais de BPCAPooling parecem conter ruídos visuais, enquanto as saídas das últimas camadas com *Max Pooling* aparentam conter mais informações agregadas. Isso é evidenciado na Figura 7.9, que mostra resultados com resolução superior a 2×2 , devido à resolução maior das imagens de entrada.

Figura 7.9: Visualização das últimas camadas de *pooling* no conjunto *Food-101*.



(a) Camada com *Max Pooling*.



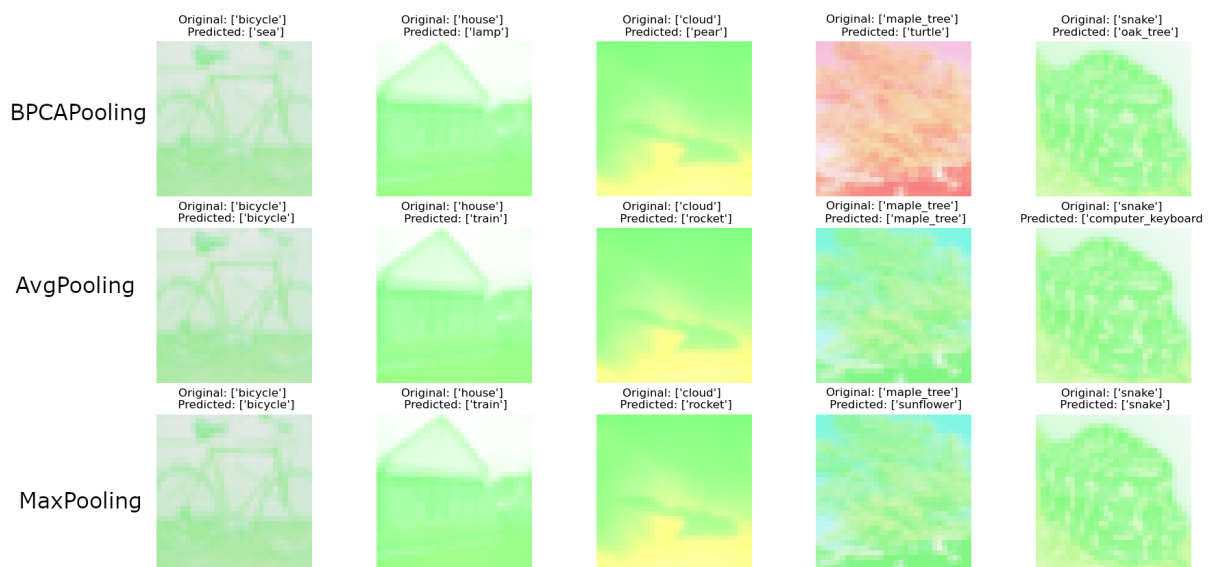
(b) Camada com BPCAPooling.

Fonte: do próprio autor.

No contexto das análises realizadas com LIME, é importante ressaltar um fator crucial para o desempenho dessa técnica: o tamanho da amostra de imagem. Ao considerar as imagens

do conjunto de dados CIFAR 100, obter resultados satisfatórios foi desafiador devido ao pequeno tamanho de 32×32 *pixels* de cada exemplo do conjunto de dados. Nessa escala, cada *pixel* possui um peso significativo em comparação com a área total da imagem. Na Figura 7.10, as contribuições positivas são representadas por *pixels* verdes e as contribuições negativas por *pixels* vermelhos para uma inferência de classe específica. Entretanto, como evidenciado na Figura, todos os *pixels* influenciam positiva ou negativamente a classificação devido ao pequeno tamanho da imagem, não havendo distinções por região.

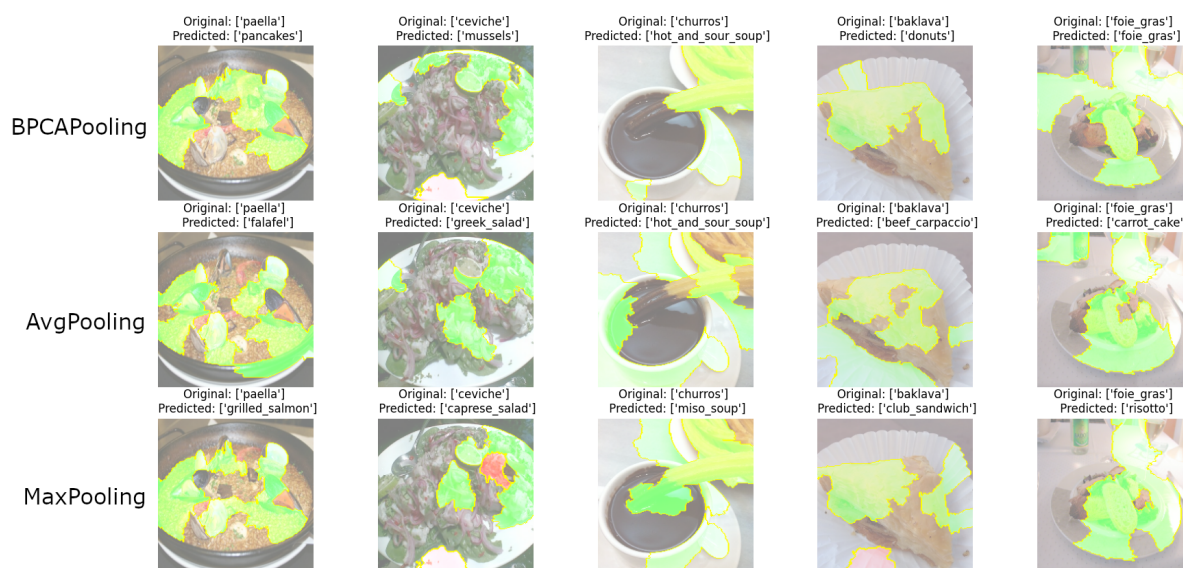
Figura 7.10: Exemplo de imagens explicadas por LIME no conjunto de dados CIFAR 100.



Fonte: do próprio autor.

Por outro lado, ao examinar as imagens relacionadas ao conjunto de dados *Food-101*, é evidente que o LIME oferece mais *insights* sobre as classificações realizadas. Como demonstrado na Figura 7.11, apenas as regiões relevantes (positivamente ou negativamente) são destacadas, apresentando um tamanho de imagem apropriado, ao contrário do conjunto de dados CIFAR 100. Além disso, a análise dos resultados revela que, em alguns casos, a rede enfoca mais em embalagens, utensílios e louças, diminuindo a relevância das informações relacionadas à alimentação. Esta questão poderia ser abordada ao melhorar a estratégia de aquisição de dados, como ampliar as imagens ou aplicar técnicas de segmentação (por exemplo, [Rother et al. \(2004\)](#)), garantindo que a comida ocupe a maior parte da área da imagem.

Figura 7.11: Exemplo de imagens explicadas por LIME no conjunto de dados *Food-101*.



Fonte: do próprio autor.

7.2 Resultados da Segmentação Semântica

Em relação aos resultados da segmentação semântica, é relevante destacar que as discussões e descobertas abordadas na tarefa de classificação (Seção 7.1) foram consideradas como um primeiro passo rumo à implementação do método BPCA como camada de *pooling* (BPCAPooling) na tarefa subsequente de segmentação semântica. Esta transição para uma nova tarefa demandaria validações finais para aprimorar o processo experimental específico para esse contexto.

É importante observar que, ao considerar as segmentações realizadas, os resultados obtidos com a aplicação do método BPCAPooling apresentaram similaridades em termos de valores quando comparados ao método tradicional *Max Pooling* na arquitetura da U-Net. Essa comparação pode ser visualizada nas Tabelas 13 e 14.

Tabela 13: Resultados de U-Nets treinadas por 500 épocas no conjunto de dados *Oxford-IIIT Pets* com base em acurácia.

Método de <i>Pooling</i>	Tempo de Execução (minutos)	Acurácia (%)		<i>Loss</i>	
		Treino	Validação	Treino	Validação
BPCAPooling	252:82	87,982	87,50	0,5160	0,7284
<i>Max Pooling</i>	96:98	89,381	88,64	0,5484	0,8109

Fonte: do próprio autor.

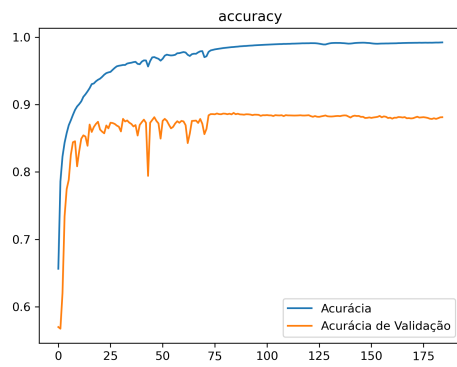
Tabela 14: Resultados de U-Nets treinadas por 500 épocas no conjunto de dados *Oxford-IIIT Pets* com base em mIoU.

Método de <i>Pooling</i>	Tempo de Execução (minutos)	Acurácia (%)		<i>Loss</i>		mIoU		Dice	
		Treino	Validação	Treino	Validação	Treino	Validação	Treino	Validação
BPCAPooling	154:09	86,30	86,77	0,5756	0,6659	0,3344	0,3333	0,8892	0,8953
<i>Max Pooling</i>	57:23	88,750	88,61	0,4683	0,6668	0,3367	0,3367	0,8887	0,8953

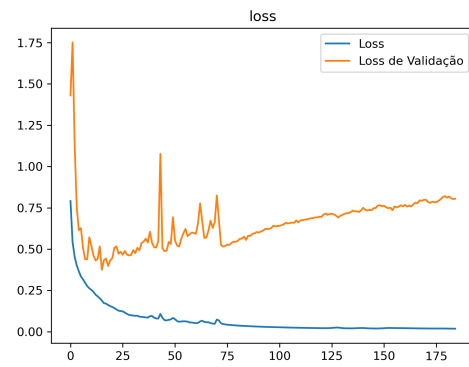
Fonte: do próprio autor.

Analisando as tabelas anteriores, observa-se que, embora as métricas adotadas para conduzir os experimentos sejam acurácia e mIoU, a *loss* associada ao BPCAPooling foi menor nas etapas de validação, conforme evidenciado na Figuras 7.12b e 7.12d. Além disso, o método de *pooling* se demonstrou ser igual ou superior aos resultados obtidos com o uso do BPCAPooling quando mensurado pela métrica Dice, apesar da diferença não ser significativa. Entretanto, é importante ressaltar que, quantitativamente, a métrica mIoU tende a penalizar mais instâncias únicas com classificações ruins do que a métrica Dice, mesmo quando ambas concordam que essa instância é ruim. Ademais, a métrica mIoU tende a agravar os erros, resultando em um efeito de “quadratura” em relação ao Dice. Como resultado, a métrica Dice parece medir algo mais próximo do desempenho médio, como evidenciado nas Figuras 7.13c e 7.14c, enquanto a mIoU mede algo mais próximo do pior caso de desempenho.

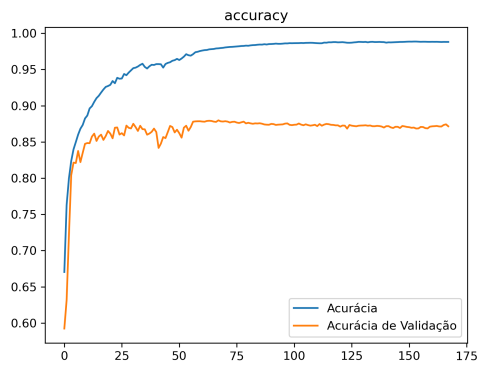
Figura 7.12: Métricas de U-Nets com 500 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em acurácia.



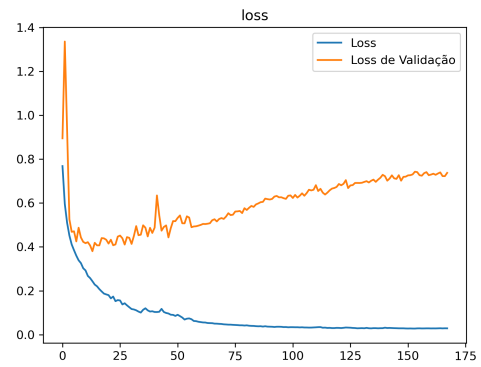
(a) Acurácia do *Max Pooling*.



(b) *Loss* do *Max Pooling*.



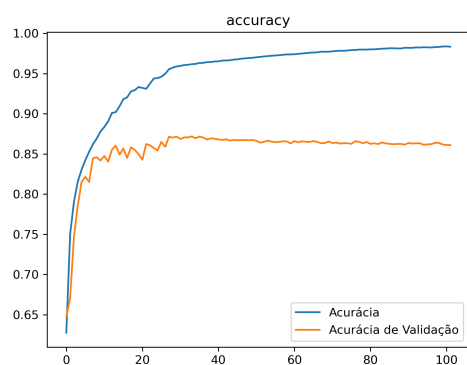
(c) Acurácia do BPCAPooling.



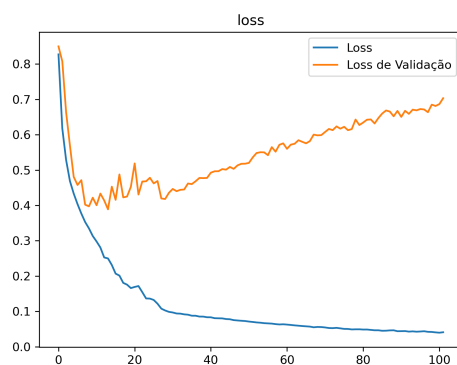
(d) *Loss* do BPCAPooling.

Fonte: do próprio autor.

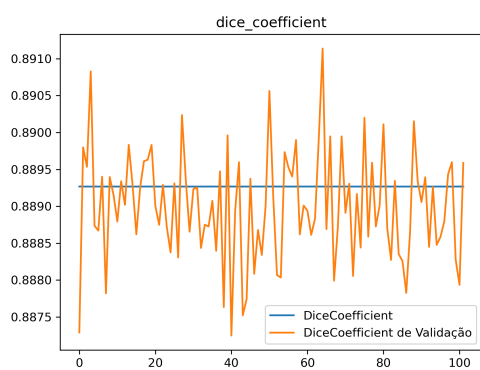
Figura 7.13: Métricas de U-Net com BPCAPooling e 500 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em mIoU.



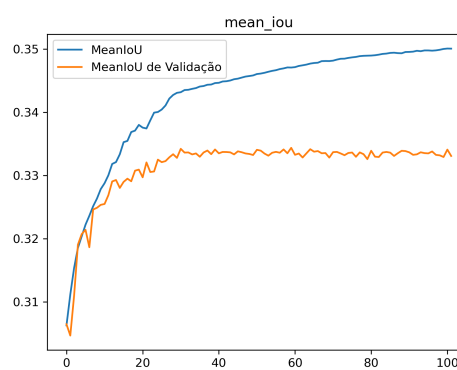
(a) Acurácia.



(b) Loss.



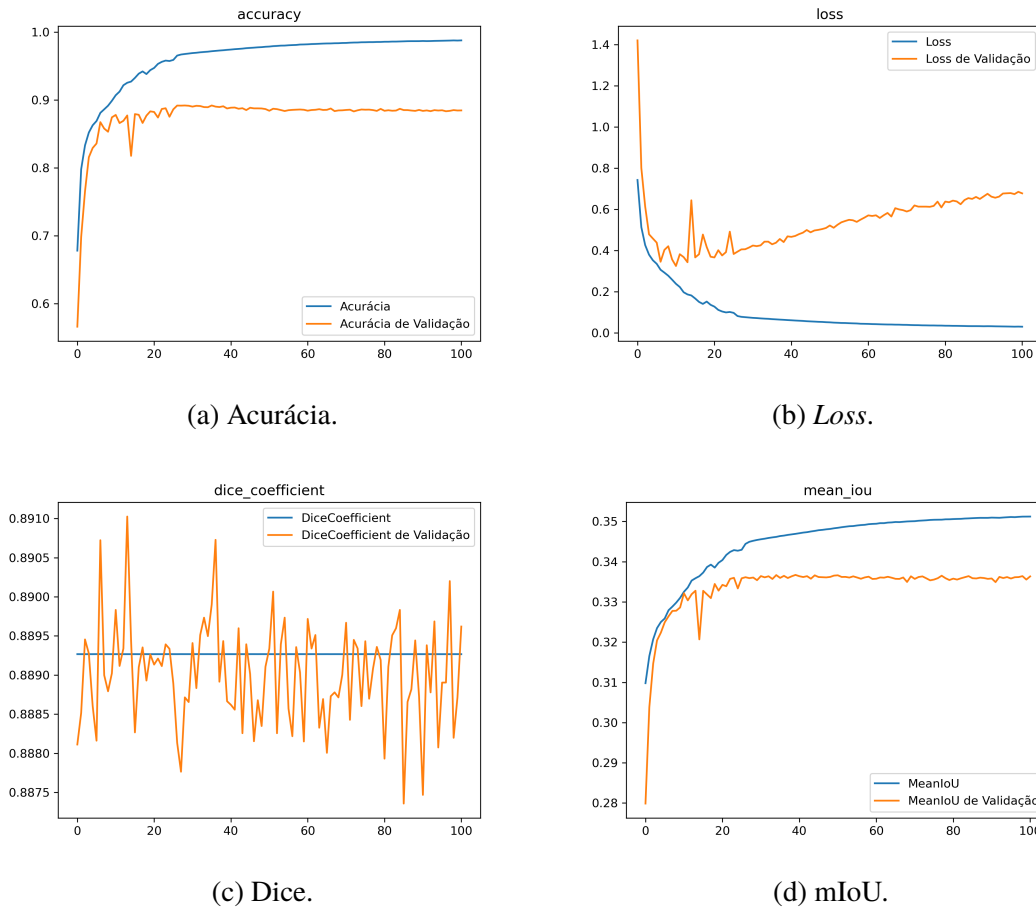
(c) Dice.



(d) mIoU.

Fonte: do próprio autor.

Figura 7.14: Métricas de U-Net com *Max Pooling* e 500 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em mIoU.



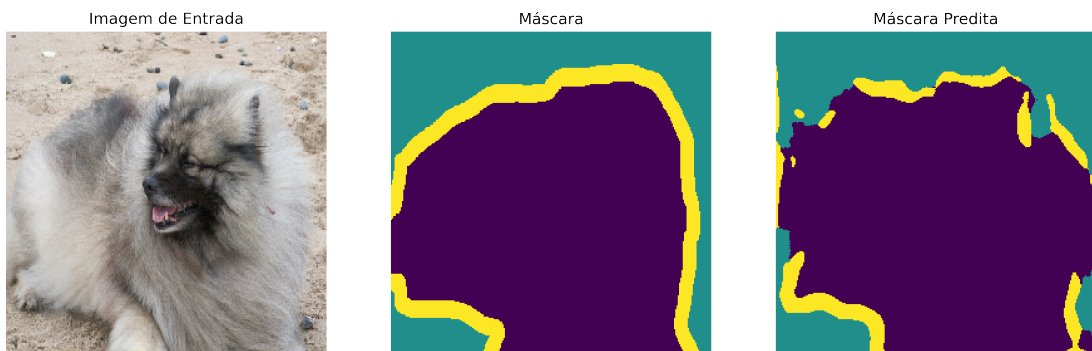
Fonte: do próprio autor.

Outro aspecto a se considerar ao analisar as Tabelas 13 e 14 é o tempo de execução durante os processos de treinamento e validação. Notavelmente, os tempos para o método *Max Pooling* são significativamente mais rápidos quando comparado ao BPCAPooling, atribuídos à complexidade do método proposto discutida na Seção 8.1. No entanto, é importante destacar que os modelos atingiram a convergência em um número próximo de épocas, os modelos baseados na métrica de acurácia convergiram em 168 épocas para BPCAPooling e 172 épocas para *Max Pooling*. Enquanto, os modelos baseados na métrica mIoU convergiram em 102 épocas para BPCAPooling e 101 para *Max Pooling*.

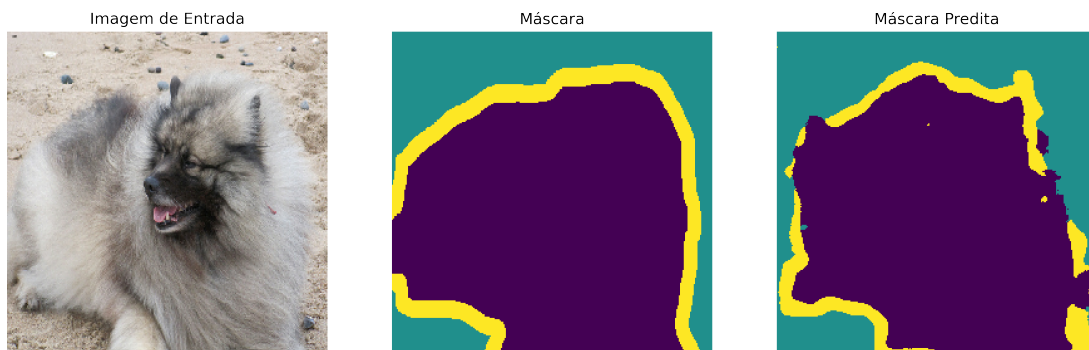
Por fim, ao considerar um nível de significância de 5% para os modelos U-Nets treinados durante 500 épocas, observa-se que não há diferença significativa nos campos de acurácia, tanto nos conjuntos de teste quanto nos de validação, apresentando p-valores de 0.1479 e 0.2192, respectivamente. No entanto, ao analisar os valores de mIoU, embora os p-valores sejam de 0.0678 e 0.0110 para teste e validação, respectivamente, indicando ausência de diferença significativa

no primeiro caso e uma diferença no segundo caso (considerando um nível de significância de 5%), é importante ressaltar que visualmente é desafiador discernir tais diferenças, uma vez que a segmentação apresenta uma semelhança notável, sendo assim maiores detalhes sobre as diferenças serão relatados ainda nessa seção. Os resultados visuais podem ser observados nas Figuras 7.15 e 7.16, correspondendo ao primeiro e segundo caso mencionados, respectivamente.

Figura 7.15: Imagem de entrada, máscara e saída do modelo U-Net baseado em acurácia, respectivamente.



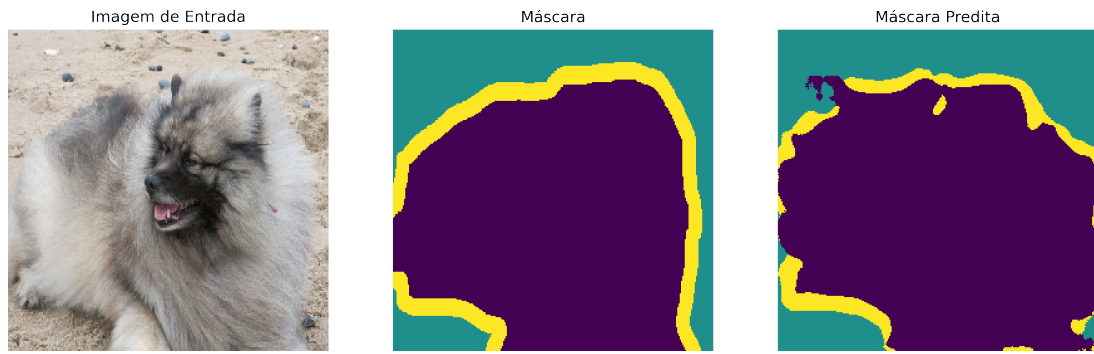
(a) Modelo com *Max Pooling*.



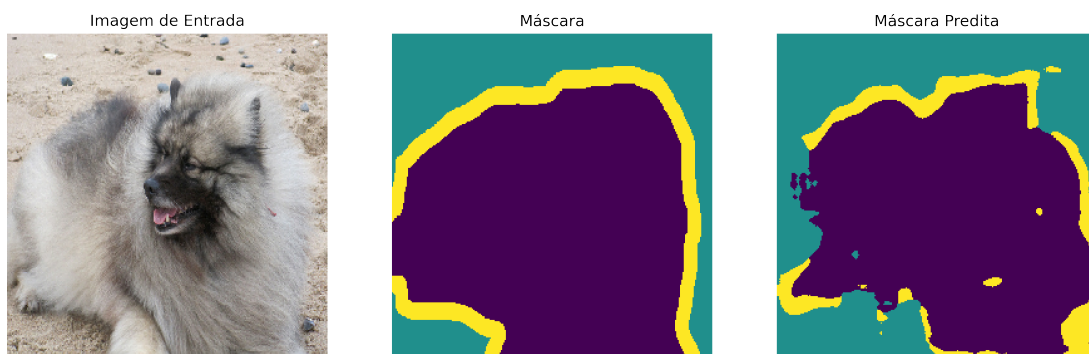
(b) Modelo com BPCAPooling.

Fonte: do próprio autor.

Figura 7.16: Imagem de entrada, máscara e saída do modelo U-Net baseado em mIoU, respectivamente.



(a) Modelo com *Max Pooling*.

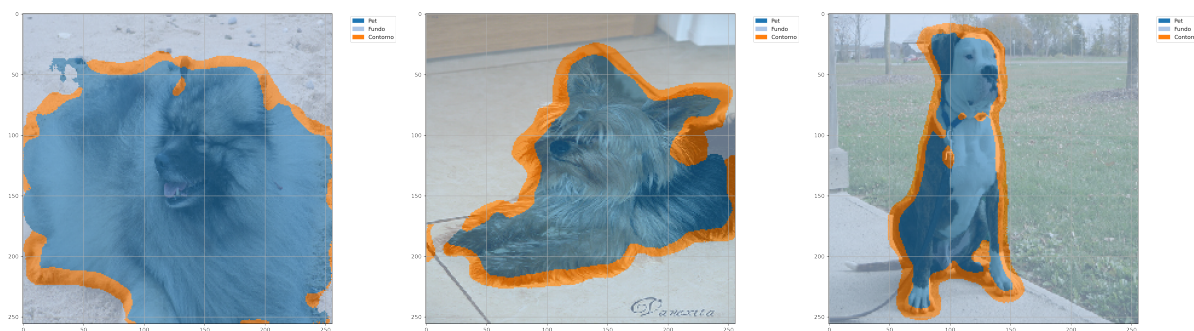


(b) Modelo com BPCAPooling.

Fonte: do próprio autor.

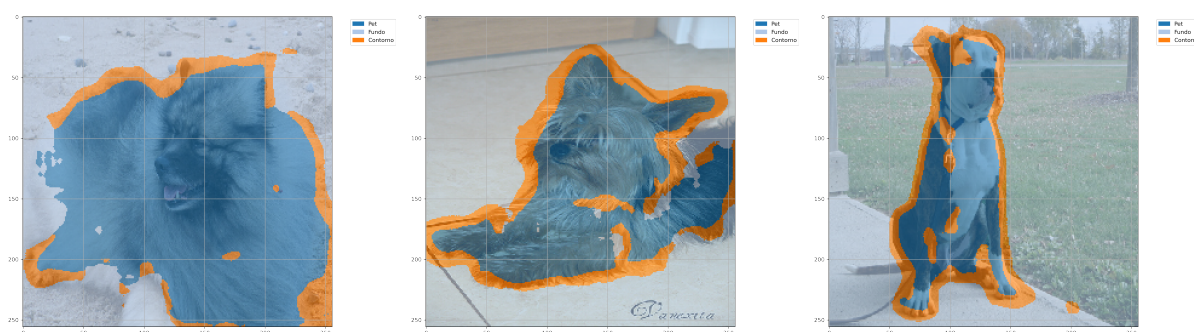
Três representações das máscaras previstas sobrepondo a imagem de entrada são exibidas para cada um dos modelos nas Figuras 7.17, 7.18, 7.19 e 7.20. Observa-se que a aplicação da métrica mIoU se mostra mais eficaz quando comparada à acurácia. Além disso, o método BPCAPooling enfatiza mais os contornos em comparação com o método *Max Pooling*, que aparentemente enfatiza mais os *pets*, tendo que para ambos os métodos o fundo possui a mesma eficácia de segmentação.

Figura 7.17: Exemplos segmentados a partir de U-Net com *Max Pooling* e 500 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em mIoU.



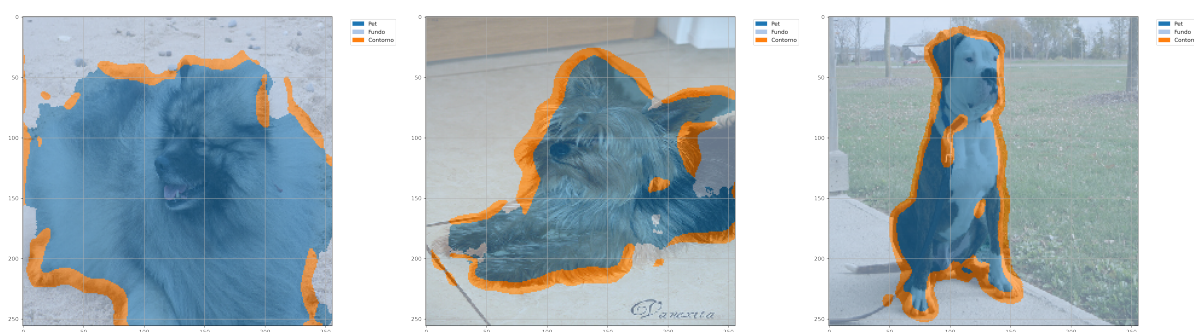
Fonte: do próprio autor.

Figura 7.18: Exemplos segmentados a partir de U-Net com BPCAPooling e 500 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em mIoU.



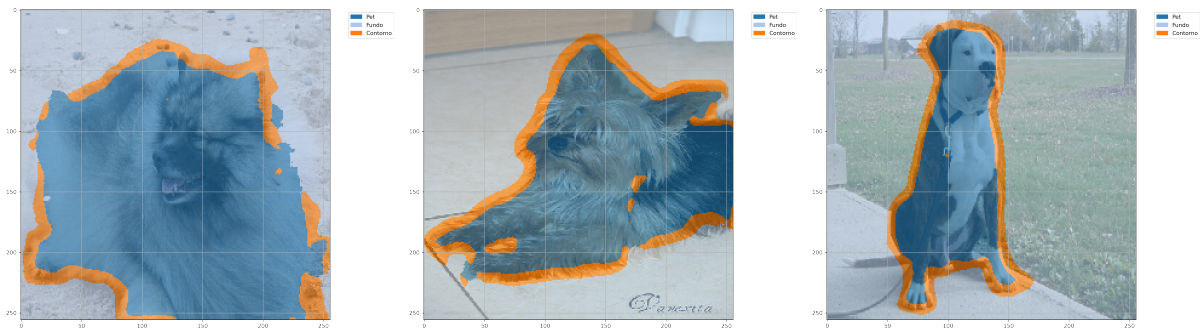
Fonte: do próprio autor.

Figura 7.19: Exemplos segmentados a partir de U-Net com *Max Pooling* e 500 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em acurácia.



Fonte: do próprio autor.

Figura 7.20: Exemplos segmentados a partir de U-Net com BPCAPooling e 500 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em acurácia.



Fonte: do próprio autor.

Os resultados obtidos com a arquitetura U-Net-*Like* não apresentaram grandes diferenças quando comparados com os casos anteriores ou com a utilização do método *Max Pooling*. No entanto, é importante ressaltar que, ao empregar a U-Net-*Like*, os resultados do BPCAPooling mostraram-se inferiores em todos os aspectos, exceto nos valores de Dice na etapa de validação, como também ocorreu na arquitetura U-Net. Essa comparação é ilustrada nas Tabelas 15 e 16.

Tabela 15: Resultados de U-Net-*Likes* treinadas por 500 épocas no conjunto de dados *Oxford-IIIT Pets* com base em acurácia.

Método de <i>Pooling</i>	Tempo de Execução (minutos)	Acurácia (%)		<i>Loss</i>	
		Treino	Validação	Treino	Validação
BPCAPooling	116:99	79,312	78,61	0,7032	0,8109
<i>Max Pooling</i>	24:81	82,472	81,84	0,6144	0,7936

Fonte: do próprio autor.

Tabela 16: Resultados de U-Net-*Likes* treinadas por 20 épocas no conjunto de dados *Oxford-IIIT Pets* com base em mIoU.

Método de <i>Pooling</i>	Tempo de Execução (minutos)	Acurácia (%)		<i>Loss</i>		mIoU		Dice	
		Treino	Validação	Treino	Validação	Treino	Validação	Treino	Validação
BPCAPooling	10:99	70,95	71,55	0,7133	0,6935	0,3086	0,3102	0,8889	0,8970
<i>Max Pooling</i>	3:19	77,97	78,19	0,5571	0,5508	0,3164	0,3177	0,8892	0,8970

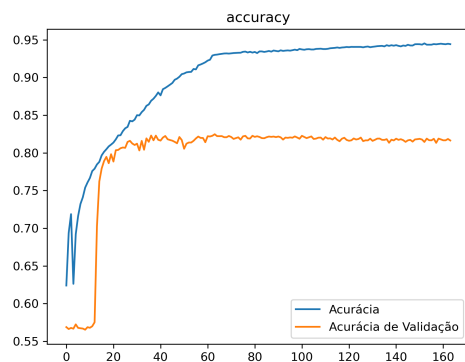
Fonte: do próprio autor.

Além da métrica Dice, vale ressaltar que o comportamento do tempo de execução é

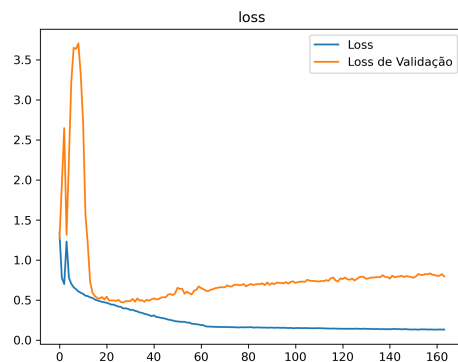
muito semelhante ao observado nos casos de uso das U-Nets, em que as abordagens com *Max Pooling* demonstram um melhor desempenho. Contudo, ao comparar os tempos de execução da Tabela 15 com as Tabelas 13 e 14, nota-se que o desempenho das U-Net-Like é superior. Isso se deve à influência das estratégias de otimização, como normalização de *batch*, convolução separável e *upsampling*, que reduzem a quantidade de parâmetros em comparação com as U-Nets convencionais. No entanto, as comparações em relação aos valores de avaliação ainda se mostram superiores nas U-Nets convencionais, demonstrando que a robustez da arquitetura original tem seus benefícios.

A Figura 7.21 ilustra o progresso das métricas durante o processo de treinamento e validação dos modelos de U-Net-Like, guiados pela métrica de acurácia. Já as Figuras 7.22 e 7.23 mostram os resultados dos modelos U-Net-Like treinados por menos épocas, mas que apresentam métricas mais apropriadas para o contexto de segmentação semântica.

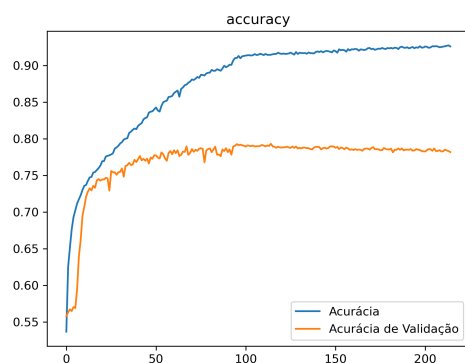
Figura 7.21: Métricas de U-Net-Likes com 500 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em acurácia.



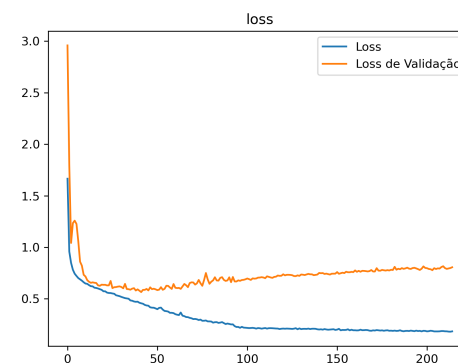
(a) Acurácia do *Max Pooling*.



(b) *Loss* do *Max Pooling*.



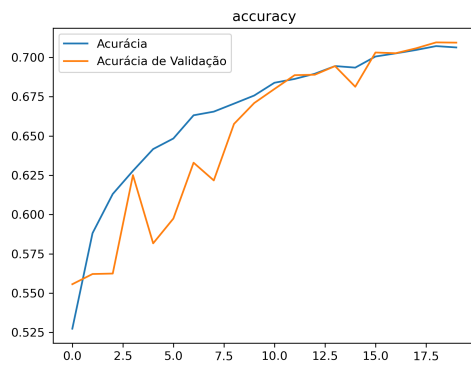
(c) Acurácia do BPCAPooling.



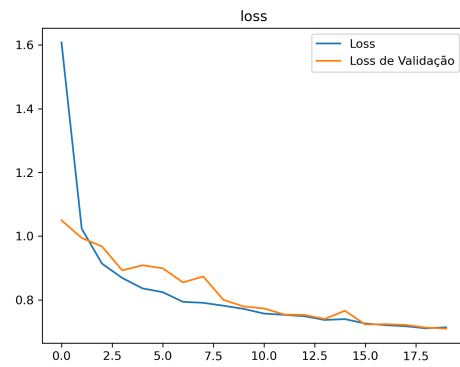
(d) *Loss* do BPCAPooling.

Fonte: do próprio autor.

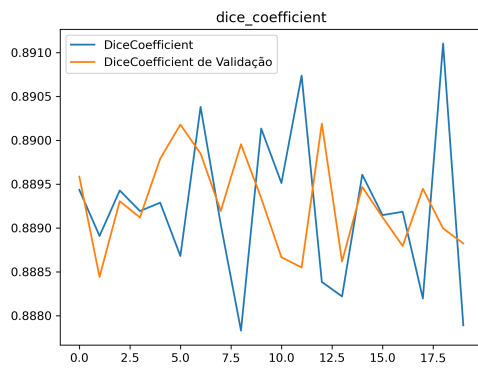
Figura 7.22: Métricas de U-Net-Likes com BPCAPooling e 20 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em mIoU.



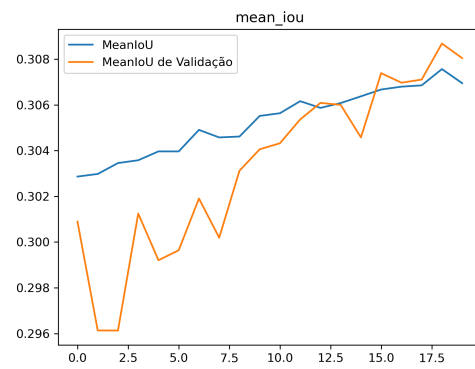
(a) Acurácia.



(b) Loss.



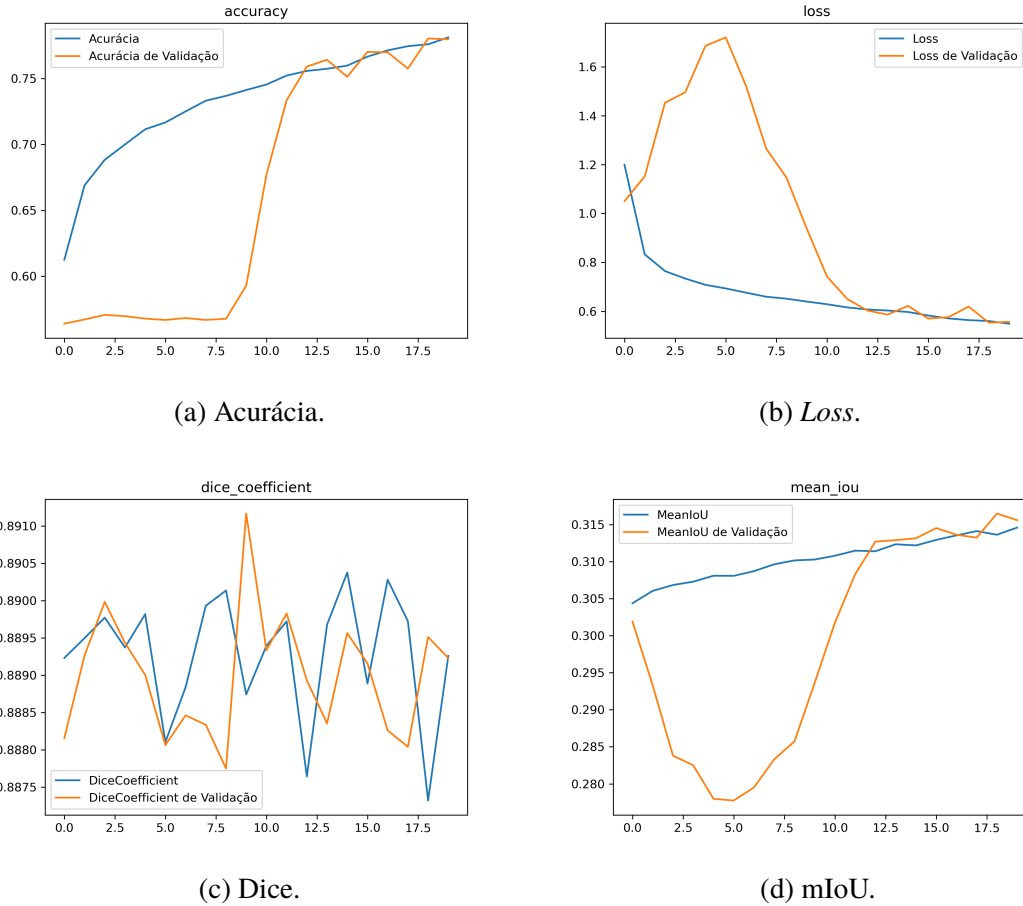
(c) Dice.



(d) mIoU.

Fonte: do próprio autor.

Figura 7.23: Métricas de U-Net-Likes com *Max Pooling* e 20 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em mIoU.

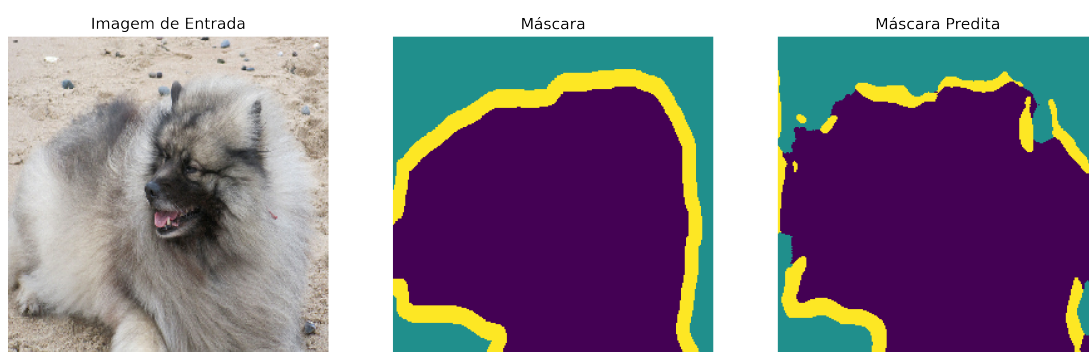


Fonte: do próprio autor.

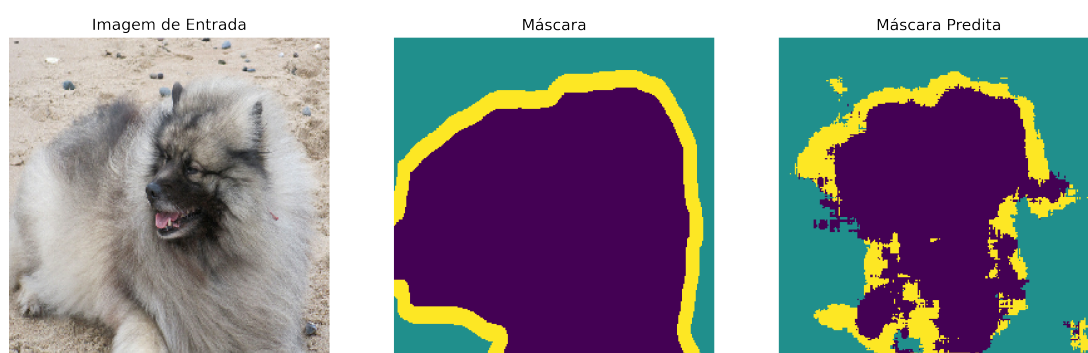
Analisando essas tabelas por meio de testes estatísticos, observa-se uma diferença estatisticamente significativa para as métricas de acurácia e *loss* nas fases de treinamento e validação das U-Net-Like baseadas na acurácia, considerando um nível de significância de 5%. No entanto, essa diferença não é observada para as demais métricas, como Dice, acurácia de validação, Dice de validação, *loss* de validação e mIoU de validação, apresentando os respectivos valores de p-valor de 0.7962, 0.6730, 0.2130, 0.1175 e 0.2198, ainda mantendo o limiar de significância de 5%.

Em termos visuais, assim como constatado anteriormente com as U-Nets, além da análise das métricas, as imagens também indicam uma qualidade inferior na segmentação em comparação com os modelos U-Net. As Figuras 7.24 e 7.25 exibem as imagens de entrada, as máscaras e as máscaras geradas pelo modelo para um melhor entendimento visual.

Figura 7.24: Imagem de entrada, máscara e saída do modelo U-Net-*Like* baseado em acurácia, respectivamente.



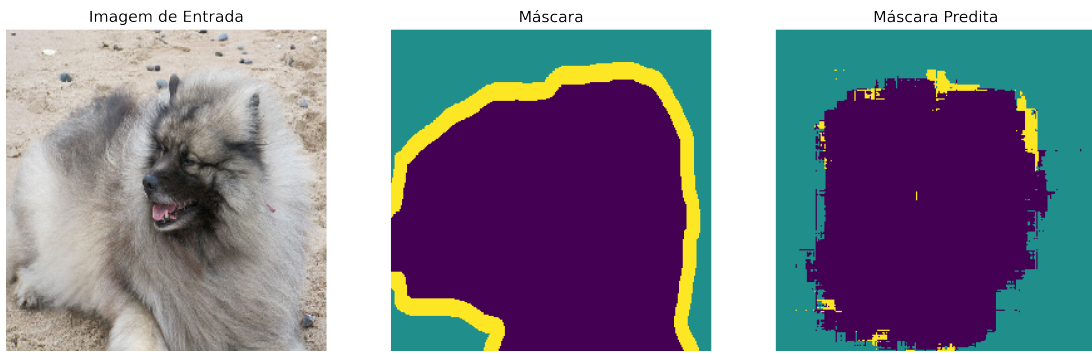
(a) Modelo com *Max Pooling*.



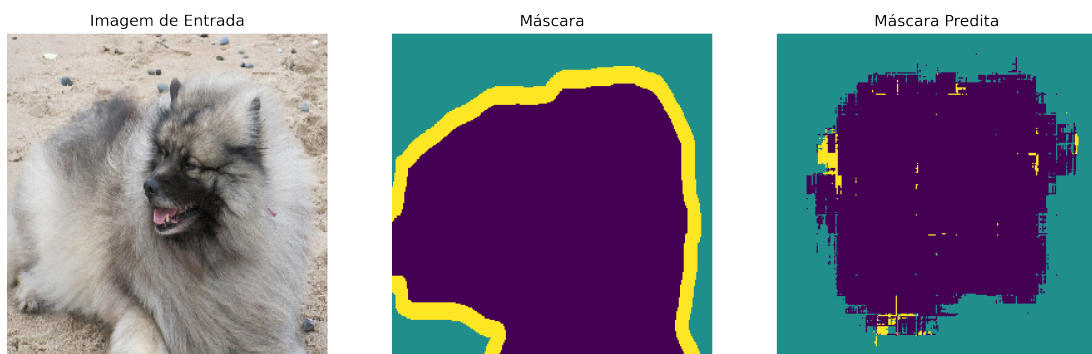
(b) Modelo com BPCAPooling.

Fonte: do próprio autor.

Figura 7.25: Imagem de entrada, máscara e saída do modelo U-Net-*Like* baseado em mIoU, respectivamente.



(a) Modelo com *Max Pooling*.

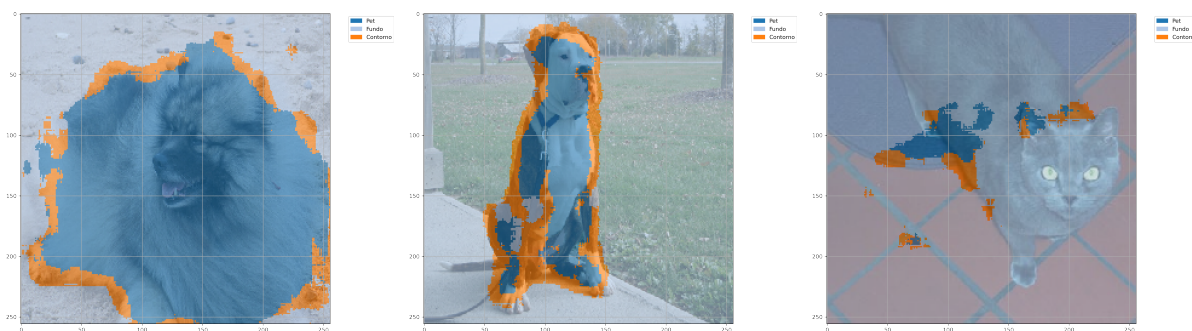


(b) Modelo com *BPCAPooling*.

Fonte: do próprio autor.

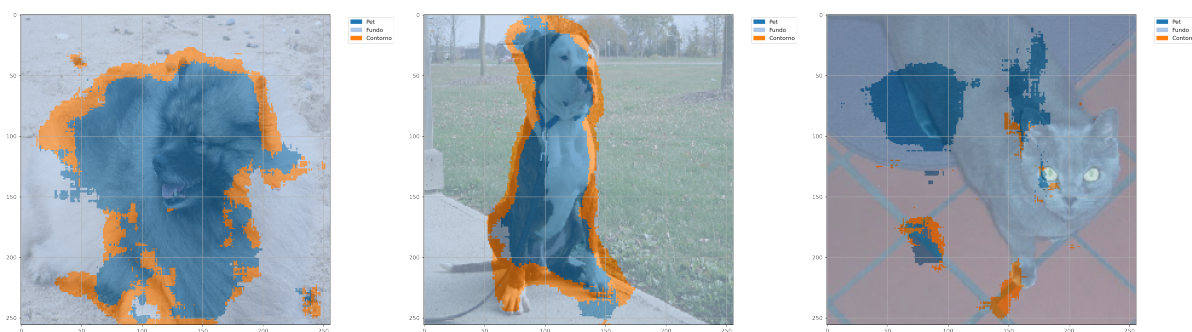
Por fim, ao analisar as Figuras 7.26, 7.27, 7.28 e 7.29, evidencia-se a sobreposição dos resultados do modelo nas imagens de entrada. Esta visualização sinaliza a necessidade de estender o número de épocas para os modelos de segmentação U-Net-*Like* treinados por apenas 20 épocas. Apesar da diferença nas acurácias não exceder 8%, a análise visual sugere um potencial de melhoria substancial, indicando que o modelo possivelmente pode obter resultados mais robustos com um treinamento prolongado.

Figura 7.26: Exemplos segmentados a partir de U-Net-Like com *Max Pooling* e 500 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em acurácia.



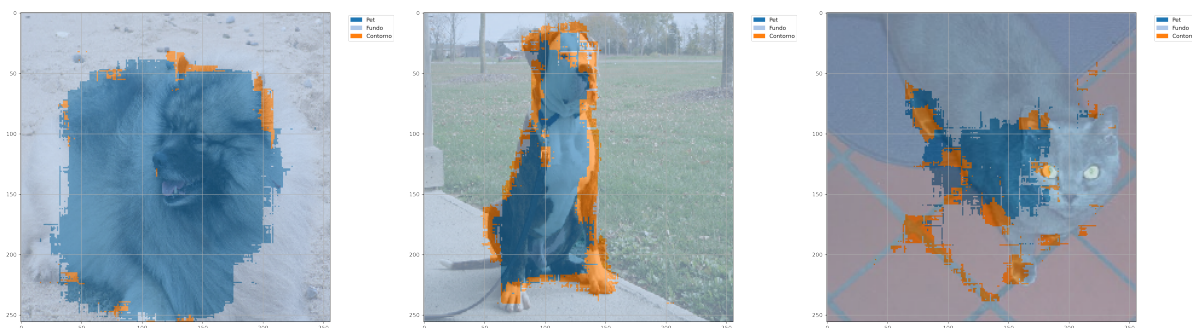
Fonte: do próprio autor.

Figura 7.27: Exemplos segmentados a partir de U-Net-Like com BPCAPooling e 500 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em acurácia.



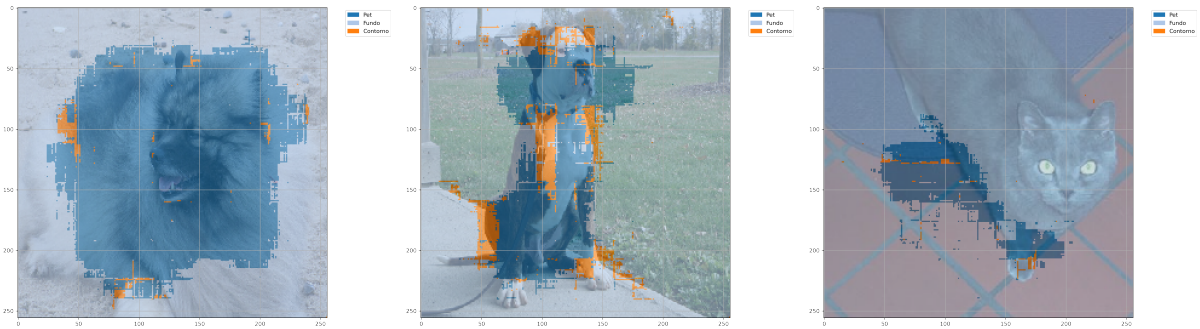
Fonte: do próprio autor.

Figura 7.28: Exemplos segmentados a partir de U-Net-Like com *Max Pooling* e 20 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em mIoU.



Fonte: do próprio autor.

Figura 7.29: Exemplos segmentados a partir de U-Net-Like com BPCAPooling e 20 épocas no conjunto de dados *Oxford-IIIT Pets* baseada em mIoU.



Fonte: do próprio autor.

Como discutido na Seção 6.2.3, para realizar as análises de interpretabilidade nas segmentações semânticas, a ferramenta LIME não se mostrou adequada. Portanto, optou-se pelo uso da ferramenta Xplique, a qual oferece métodos de atribuição que permitem a visualização de pontos de destaque para a compreensão das segmentações, mapeamento das bordas das áreas de interesse ou bordas entre duas áreas distintas de interesse.

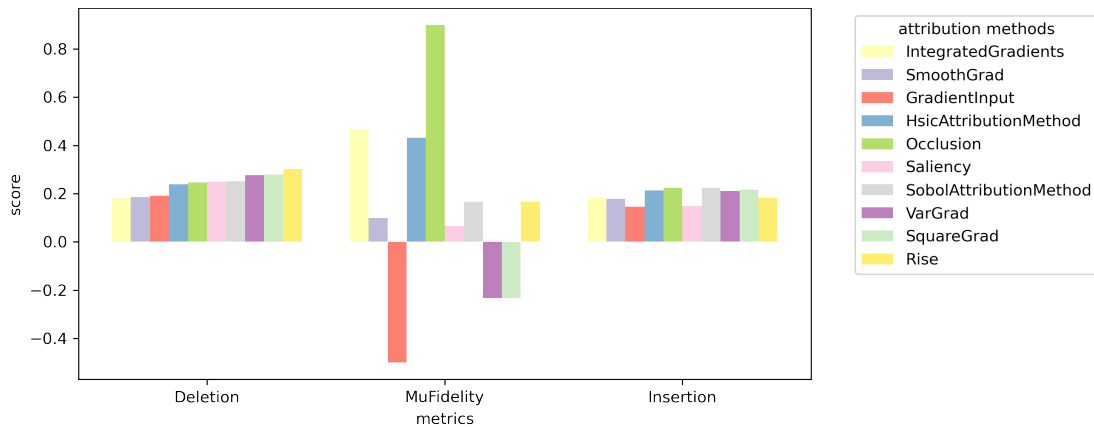
Neste estudo, as principais classes de interesse consistem nos animais de estimação, conforme proposto pelo conjunto de dados *Oxford-IIIT Pets*. Para cada um dos modelos, foram aplicados dez métodos de atribuição destinados à tarefa de segmentação semântica. Isso visava identificar os pontos mais relevantes para a compreensão da qualidade da segmentação, por meio do mapa de calor gerado por cada um dos métodos, como representado na Figura 6.14. Os métodos específicos utilizados foram: *Gradient Input*, *Integrated Gradients*, *Smooth Grad*, *Var Grad*, *Square Grad*, *Occlusion*, *Rise*, *Sobol*, *HSIC* e *Saliency*.

Ao término da execução de cada um desses métodos nos modelos, foram gerados escores para avaliar a qualidade de cada método com base em três métricas de fidelidade. Estas incluem: *Deletion*, que mede o quão bem uma explicação baseada no mapa de saliência localiza as características importantes; *Insertion*, que avalia a capacidade de uma explicação baseada no mapa de saliência em identificar elementos mínimos necessários para as classificações da classe de saída do modelo; e *MuFidelity*, que quantifica a correlação entre variáveis importantes definidas pelo método de explicação e a queda na pontuação do modelo quando essas variáveis são redefinidas para um estado inicial. É importante citar que quanto menor o escore na coluna de *Deletion*, melhor se apresenta o método, todavia, quanto maior o escore nas colunas *Insertion* e *MuFidelity*, melhor o desempenho do método.

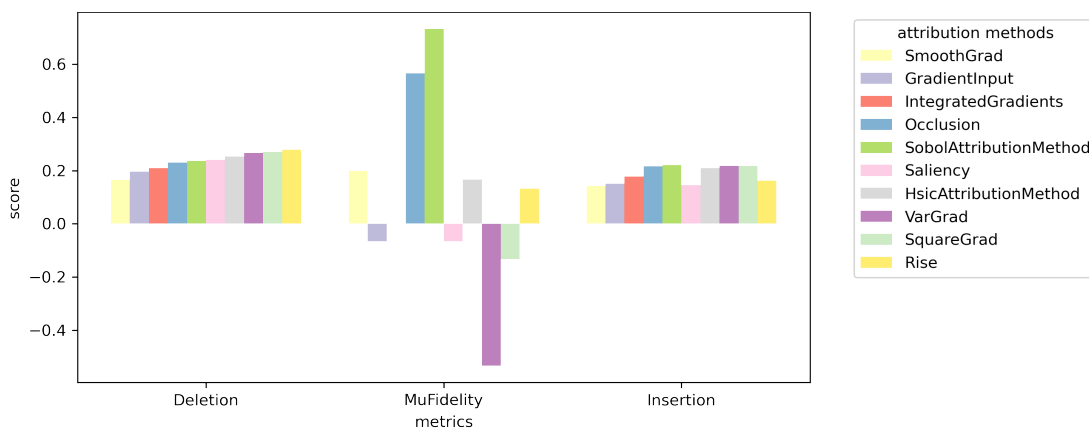
A seguir, serão apresentados os gráficos comparativos para cada um dos modelos em uma das três imagens testadas, aleatoriamente, juntamente com as saídas dos métodos de atribuição principais para cada uma das métricas de fidelidade.

Nos modelos U-Nets baseados na métrica de acurácia, os gráficos dos escores estão disponíveis na Figura 7.30. Observa-se, através desta visualização, que os destaques com o uso do método BPCAPooling foram para os métodos *Integrated Gradients*, *Occlusion* e *Sobol*, conforme representado na Figura 7.31. Estes métodos se referem, respectivamente, às métricas *Deletion*, *MuFidelity* e *Insertion*. Já para o método *Max Pooling*, os resultados mais significativos, seguindo a mesma ordem, foram obtidos pelos métodos *Smooth Grad*, *Sobol* e, novamente, *Sobol*, como evidenciado na Figura 7.32.

Figura 7.30: Gráfico com os escores dos métodos de atribuição nas U-Nets baseada na acurácia.



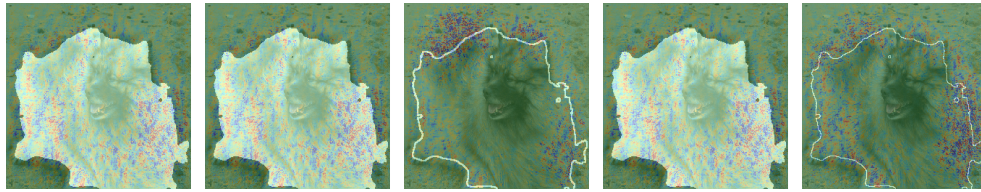
(a) Modelo com BPCAPooling.



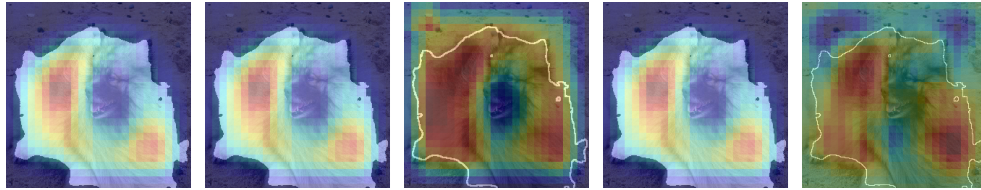
(b) Modelo com *Max Pooling*.

Fonte: do próprio autor.

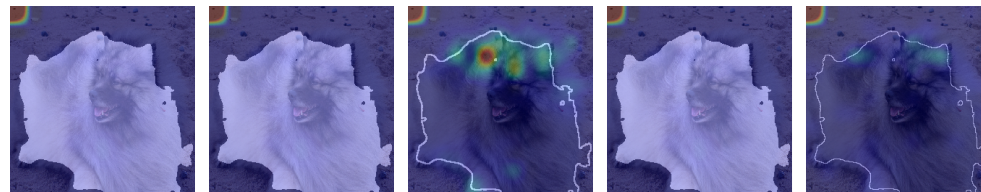
Figura 7.31: Métodos de atribuição destaque na U-Net com BPCAPooling baseada em acurácia para *Deletion*, *Mu Fidelity* e *Insertion*, respectivamente.



(a) *Integrated Gradients*.



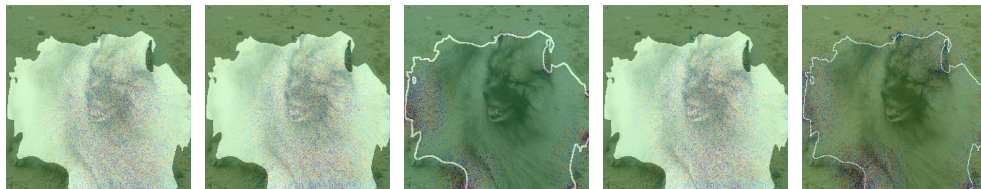
(b) *Occlusion*.



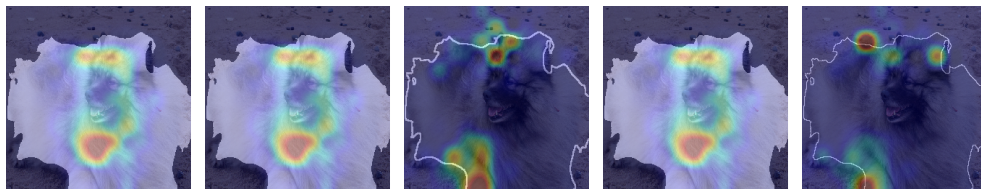
(c) *Sobol*.

Fonte: do próprio autor.

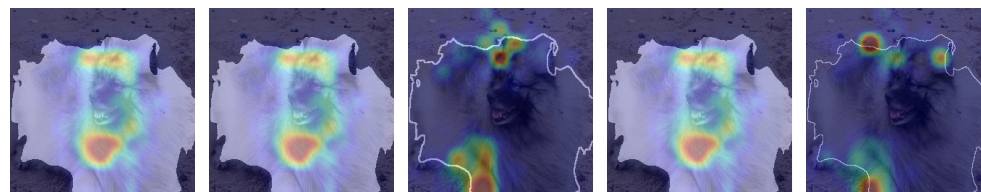
Figura 7.32: Métodos de atribuição destaque na U-Net com *Max Pooling* baseada em acurácia para *Deletion*, *Mu Fidelity* e *Insertion*, respectivamente.



(a) *Smooth Grad*.



(b) *Sobol*.



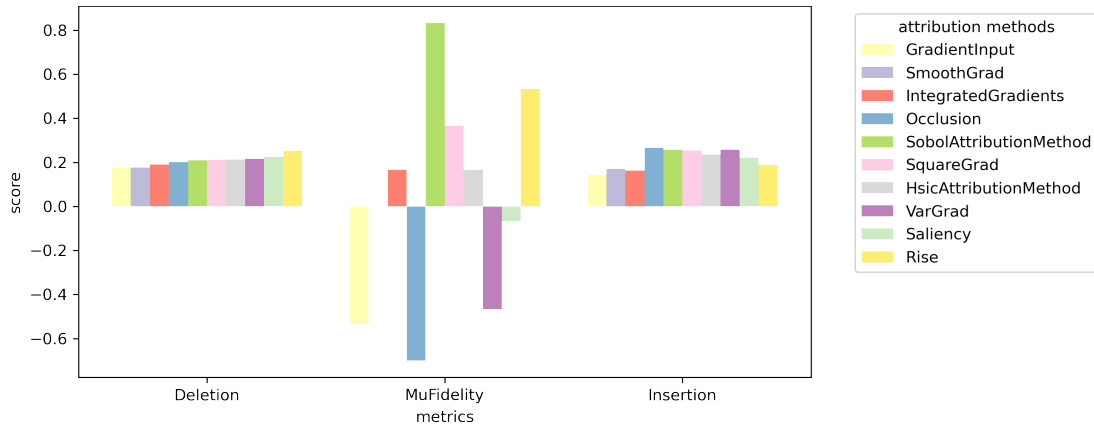
(c) *Sobol*.

Fonte: do próprio autor.

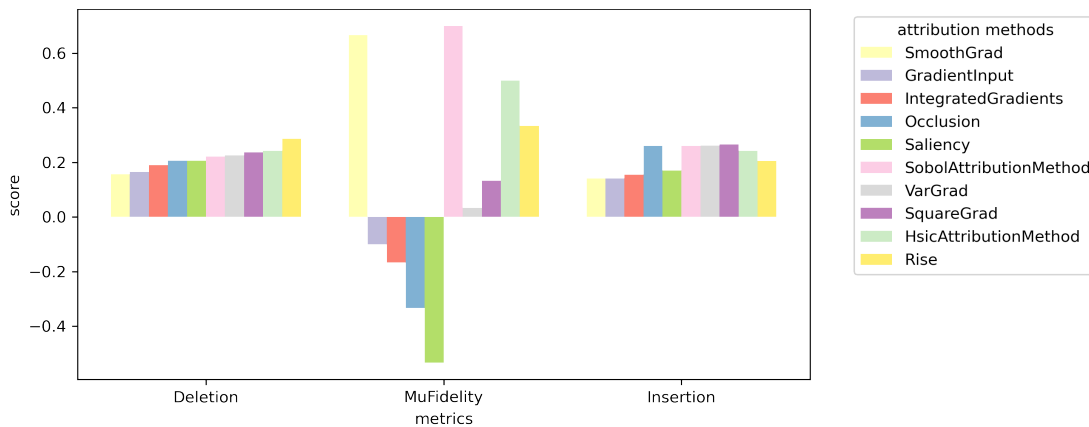
É importante ressaltar que, nesse ponto, algumas observações se destacam: nos resultados do método *Sobol* utilizando o BPCAPooling, notam-se pontos fora da região do animal de estimação considerado como mínimos para a classificação. Por outro lado, no método *Sobol* empregado no *Max Pooling*, as características internas do animal, como regiões dos olhos e peito, são apontadas como pontos mínimos de relevância.

Para a utilização do modelo U-Net com o BPCAPooling, baseado na métrica de mIoU, o gráfico dos escores está disponível na Figura 7.33. Nessa combinação, os métodos que se destacaram foram *Gradient Input*, *Sobol* e *Occlusion*, conforme representado na Figura 7.34. Já para a utilização do *Max Pooling*, os métodos mais relevantes foram *Smooth Grad*, *Sobol* e *Squared Gradients*, representados na Figura 7.35.

Figura 7.33: Gráfico com os escores dos métodos de atribuição na U-Nets baseadas em mIoU.



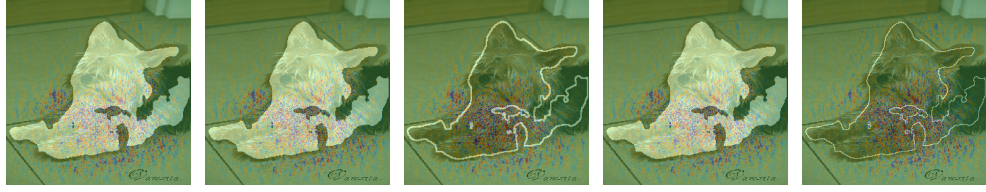
(a) Modelo com BPCAPooling.



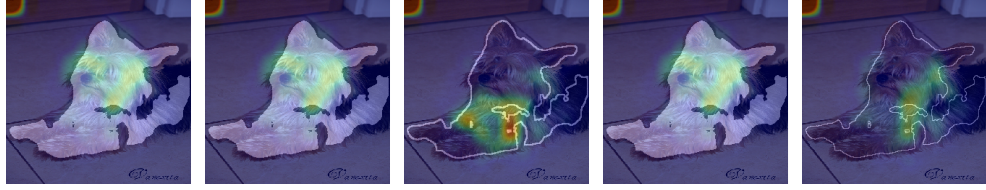
(b) Modelo com *Max Pooling*.

Fonte: do próprio autor.

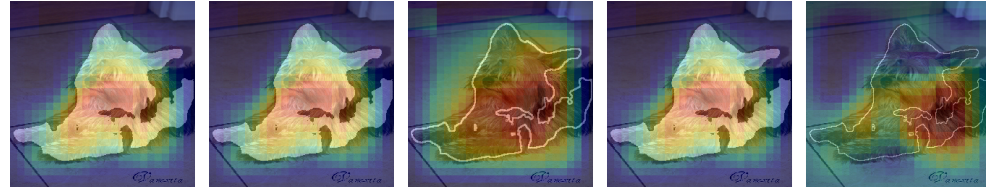
Figura 7.34: Métodos de atribuição destaque na U-Net com BPCAPooling baseada em mIoU para *Deletion*, *Mu Fidelity* e *Insertion*, respectivamente.



(a) *Gradient Input*.



(b) *Sobol*.



(c) *Occlusion*.

Fonte: do próprio autor.

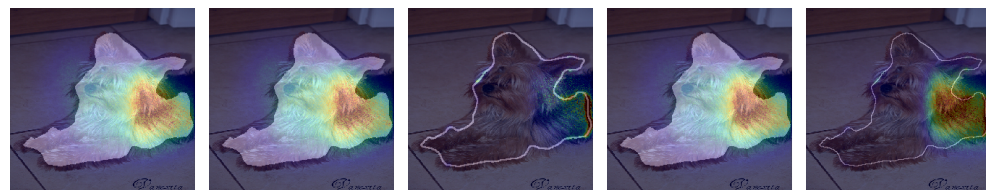
Figura 7.35: Métodos de atribuição destaque na U-Net com *Max Pooling* baseada em mIoU para *Deletion*, *Mu Fidelity* e *Insertion*, respectivamente.



(a) *Smooth Grad*.



(b) *Sobol*.



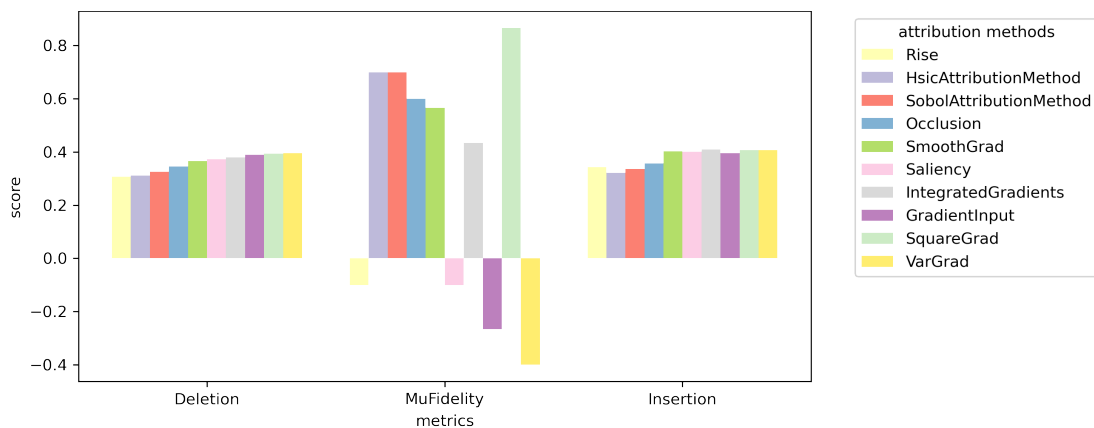
(c) *Squared Gradients*.

Fonte: do próprio autor.

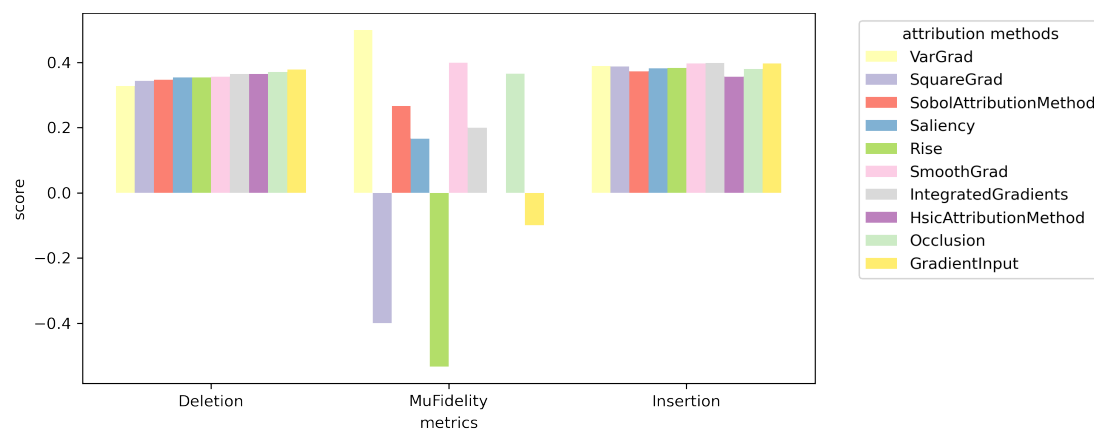
Uma observação relevante para essa combinação de modelo é que, diferentemente do modelo anterior, o uso do BPCAPooling, ao empregar o método *Occlusion* como métrica de *Insertion*, selecionou corretamente as partes centrais do animal de estimação como pontos de foco. Por outro lado, a combinação com o *Max Pooling* fez uma seleção adequada das bordas do animal, identificando-as como pontos importantes, particularmente com o método *Smooth Grad* definido como *Deletion*.

Ao analisar o desempenho das U-Net-Likes, primeiramente com base na métrica de acurácia, os escores podem ser visualizados na Figura 7.36. Observa-se um destaque significativo para os métodos *Rise*, *Squared Gradients* e *Integrated Gradients*, conforme representado na Figura 7.37, quando se utiliza o BPCAPooling. No entanto, ao utilizar o *Max Pooling*, os destaques recaem duas vezes no método *VarGrad* e, por fim, no *Integrated Gradients*.

Figura 7.36: Gráfico com os escores dos métodos de atribuição nas U-Net-Likes baseada em acurácia.



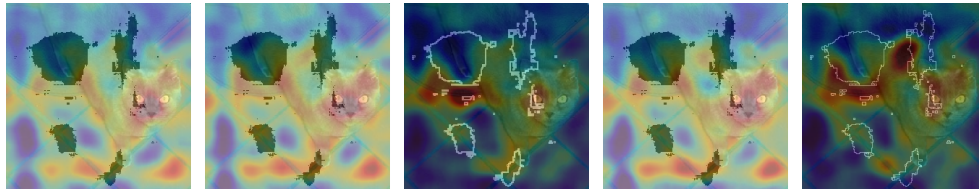
(a) Modelo com BPCAPooling.



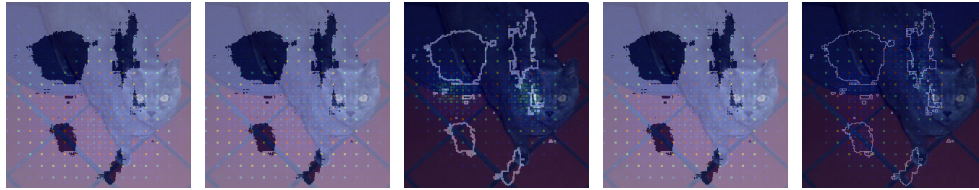
(b) Modelo com *Max Pooling*.

Fonte: do próprio autor.

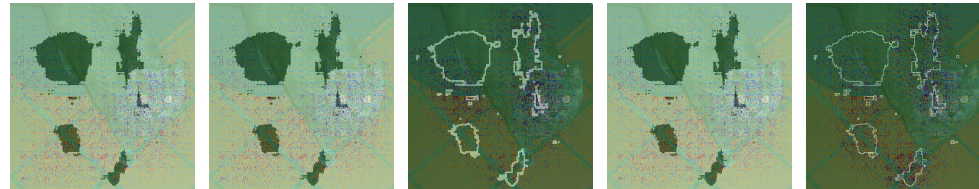
Figura 7.37: Métodos de atribuição destaque na U-Net-Like com BPCAPooling baseada em acurácia para *Deletion*, *Mu Fidelity* e *Insertion*, respectivamente.



(a) *Rise*.



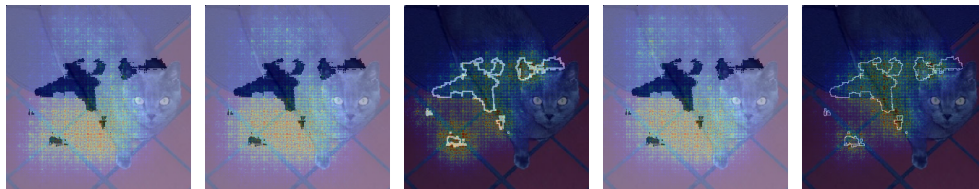
(b) *Squared Gradients*.



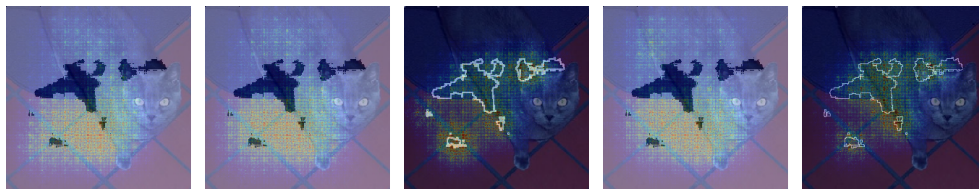
(c) *Integrated Gradients*.

Fonte: do próprio autor.

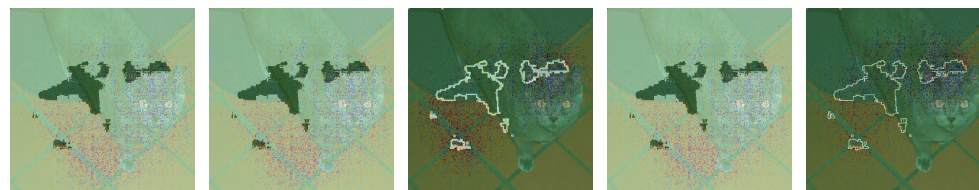
Figura 7.38: Métodos de atribuição destaque na U-Net-Like com *Max Pooling* baseada em acurácia para *Deletion*, *Mu Fidelity* e *Insertion*, respectivamente.



(a) *VarGrad*.



(b) *VarGrad*.



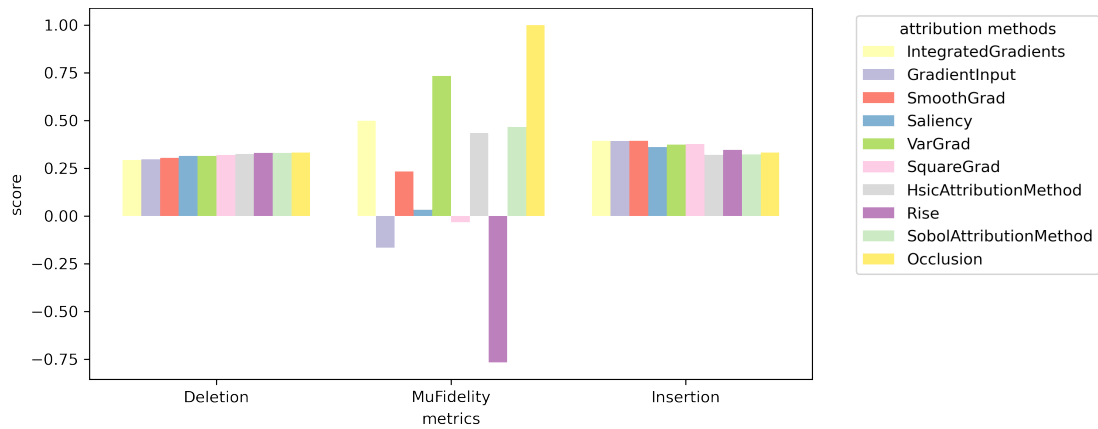
(c) *Integrated Gradients*.

Fonte: do próprio autor.

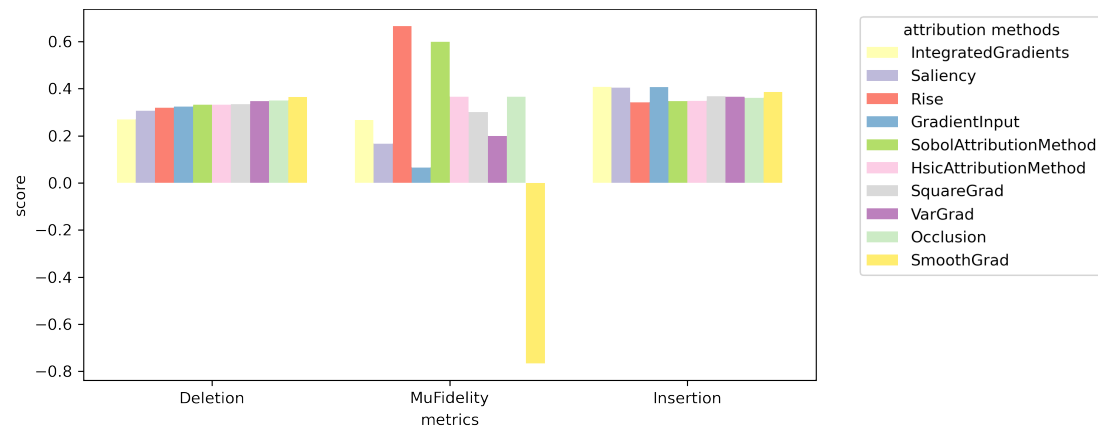
Nessa análise das U-Net-Likes, os resultados revelaram más adaptações de segmentação para ambas as arquiteturas. Notavelmente, ao utilizar o BPCAPooling, a segmentação não corresponde à forma real do gato na amostra, evidenciando uma correlação inadequada entre um subconjunto aleatório de *pixels* e a pontuação de atribuição, especialmente com a métrica *Squared Gradients*, que se destacou como a melhor para *Mu Fidelity*. Além disso, ao empregar o *Max Pooling*, o método *VarGrad* não selecionou uma grande parte do animal de interesse, demonstrando falhas na identificação das partes importantes (*Deletion*).

Na análise da arquitetura U-Net-Like, quando a métrica principal utilizada é a mIoU, observa-se uma distribuição mais ampla dos mapas de calor ao empregar ambos os métodos de *pooling*, particularmente ao visualizar a métrica *Mu Fidelity* em comparação com os métodos *Rise* e *Occlusion*. O *ranking* das métricas para ambas as estratégias de *pooling* é apresentado na Figura 7.39. Notavelmente, para o BPCAPooling, os métodos de destaque são *Integrated Gradients*, *Occlusion* e novamente *Integrated Gradients*. Por outro lado, para o método *Max Pooling*, *Integrated Gradients* se destaca duas vezes, juntamente com o método *Rise*.

Figura 7.39: Gráfico com os escores dos métodos de atribuição nas U-Net-*Likes* baseada em mIoU.



(a) Modelo com BPCAPooling.



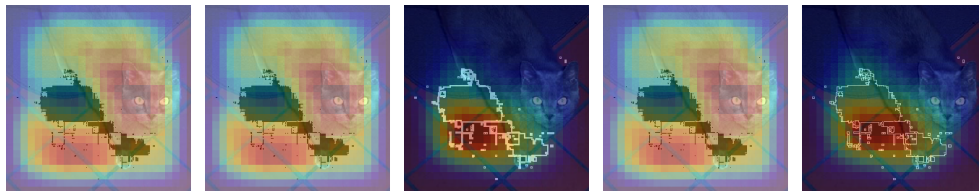
(b) Modelo com *Max Pooling*.

Fonte: do próprio autor.

Figura 7.40: Métodos de atribuição destaque na U-Net-Like com BPCAPooling baseada em mIoU para *Deletion*, *Mu Fidelity* e *Insertion*, respectivamente.



(a) *Integrated Gradients*.



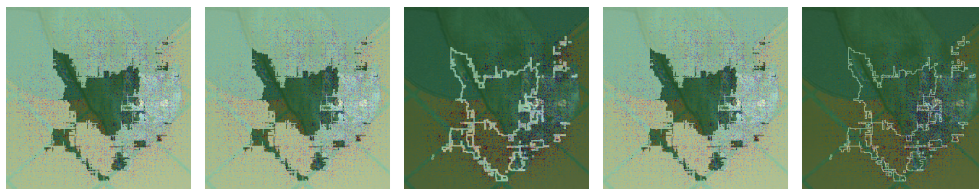
(b) *Occlusion*.



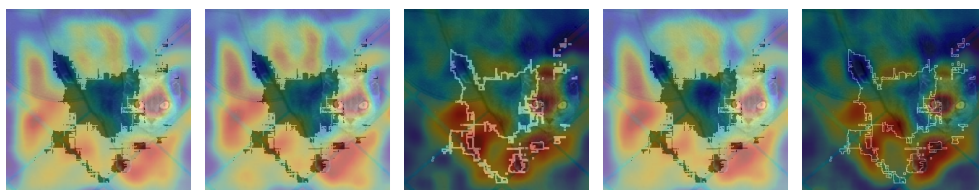
(c) *Integrated Gradients*.

Fonte: do próprio autor.

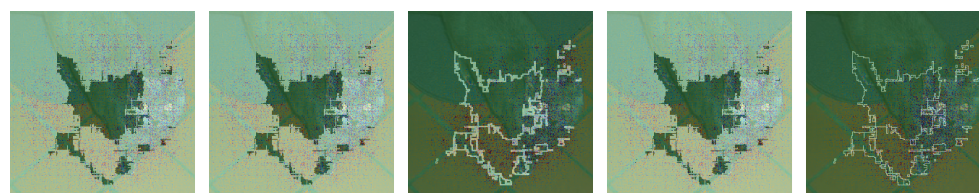
Figura 7.41: Métodos de atribuição destaque na U-Net-Like com *Max Pooling* baseada em mIoU para *Deletion*, *Mu Fidelity* e *Insertion*, respectivamente.



(a) *Integrated Gradients*.



(b) *Rise*.

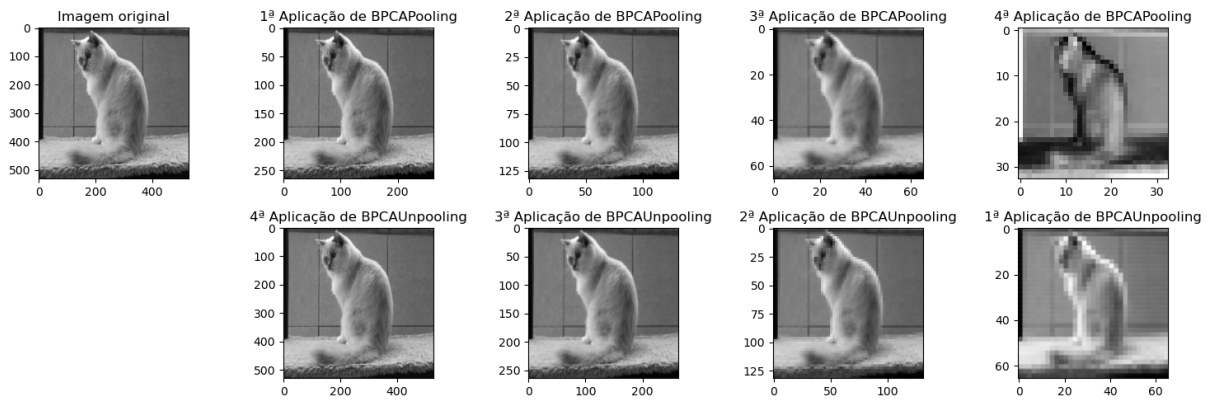


(c) *Integrated Gradients*.

Fonte: do próprio autor.

Além disso, iniciou-se o desenvolvimento do método BPCAUnpooling, disponível no repositório do projeto ⁷, mas a implementação com o *framework* atual representou uma barreira como será comentado na Seção 8.1. Esse método seria uma abordagem alternativa de *upsampling*, contrapartida ao BPCAPooling, revertendo o processo proposto por este último, preservando também a informação espacial com a consideração do contexto global no processo de reconstrução realizado pelas U-Nets e similares, como ilustrado na Figura 7.42.

Figura 7.42: Processo de redução e reconstrução de dimensionalidade com BPCAPooling e BPCAUnpooling, respectivamente.



Fonte: do próprio autor.

A técnica BPCAUnpooling, em contraste com o procedimento BPCAPooling, requisita a utilização de camadas densamente conectadas para obter um efeito similar ao oferecido por camadas convolucionais transpostas. A operação deste método é expressa na Equação 7.1, nela $X'_{i,j,k}$ simboliza o mapa de características resultante, com suas dimensões correspondendo, respectivamente, à altura (i), largura (j) e canais (k).

$$\begin{aligned} X'_{i,j,k} &= \text{BPCAUnpooling}(Y_{i',j',k'}) \\ &= \text{reshape}(U \cdot (\text{reshape}(S) \cdot \text{Dense}(\text{pad}(Y)))) \cdot \sigma + \mu. \end{aligned} \quad (7.1)$$

A equação expressa o processo de restauração da dimensionalidade pela aplicação do método BPCAUnpooling. Este processo envolve operações de preenchimento, remodelagem e reconstrução dos dados desagrupados com base na decomposição PCA. Na equação, o operador $\text{BPCAUnpooling}(Y_{i',j',k'})$ retrata a execução do procedimento BPCAUnpooling ao mapa de características de entrada Y .

⁷Repositório Github do projeto - <https://github.com/Lucs1590/USes-BPCA>

A função $\text{pad}(\mathbf{Y})$ preenche o mapa de características de entrada $\mathbf{Y}$ com zeros, adaptando o tamanho para a subsequente atividade de *unpooling*. O termo matemático $\text{Dense}(\text{pad}(\mathbf{Y}))$ é empregado na aprendizagem dos componentes decorrentes da PCA, por meio da camada densa com função de ativação.

A função *reshape* reorganiza seus argumentos para a estrutura adequada, onde S simboliza os valores singulares oriundos da decomposição PCA em alinhamento com os componentes da PCA. A expressão $U \cdot (\text{reshape}(S) \cdot \text{Dense}(\text{pad}(\mathbf{Y})))$ reconduz os dados desagrupados com o auxílio dos componentes e valores singulares fornecidos pela PCA.

Finalmente, a operação de multiplicação pelo desvio padrão σ e acréscimo da média μ refaz a normalização inicialmente feita antes da transformação PCA.

No contexto desta equação, U denota a matriz de componentes principais adquirida através da fórmula BPCAPooling, responsável por codificar a estrutura intrínseca dos dados. Esta matriz direciona a reestruturação dos dados para a dimensionalidade original.

7.3 Considerações Finais do Capítulo

Neste capítulo, foram apresentados os resultados dos experimentos realizados, abordando tanto o processo de classificação (Seção 7.1) quanto os passos em direção ao objetivo final de empregar o BPCAPooling como uma proposta de camada de *pooling* para segmentações semânticas, como explorado e detalhado na Seção 7.2.

8 CONCLUSÕES

As técnicas de segmentação representam uma ferramenta valiosa para a identificação de regiões de interesse em imagens, abordando desde situações mais simples, abordadas pelas técnicas de segmentação tradicionais (Capítulo 4), até cenários mais complexos, resolvidos pela segmentação semântica (Capítulo 5). Este último tipo de segmentação, em particular, apresenta semelhanças com a compreensão humana de cenários (Mohan e Valada, 2020), uma vez que classifica cada *pixel* em uma cena, atribuindo-os a diversas classes. Tal capacidade só foi alcançada graças à evolução das arquiteturas de redes neurais profundas (Capítulo 3), que apresentam habilidades melhoradas de aprendizado e adaptação em resposta aos problemas fornecidos.

Em uma análise mais específica, focada no campo do aprendizado de imagens, as redes neurais convolucionais ganharam destaque (Capítulo 3). Esta evolução entre as redes neurais profundas, as convolucionais e a segmentação semântica direcionou o estudo para o teste de um método de *pooling*, introduzido no contexto das CNNs, que preservasse a espacialidade das amostras, mantendo seus valores fundamentais, como explorado no Capítulo 6. O desenvolvimento desse método respondeu à primeira questão levantada nos objetivos deste trabalho, delineados na introdução (Capítulo 1).

A aplicação do método *Block-based Principal Component Analysis Pooling*, ou BPCA-Pooling, acompanhou a evolução das redes, iniciando sua aplicação em uma arquitetura comumente utilizada para as CNNs, a VGG-16. O objetivo era coletar insumos e avaliar a sua aplicabilidade como substituto dos métodos convencionais. Os resultados e discussões desses experimentos, detalhados no Capítulo 7, revelaram que os métodos convencionais - *Max Pooling* e *Avg Pooling* - ainda são superiores nos dois conjuntos de dados testados em problemas de classificação de imagens.

O melhor modelo utilizando *Max Pooling* alcançou uma acurácia de 46,071% e uma *loss* de 1,4862, no primeiro conjunto de dados, enquanto obteve uma acurácia de 50,080% e uma *loss* de 2,3745 no segundo conjunto. Já o uso do BPCA-Pooling resultou em apenas 17,543% de acurácia e 3,5121% de *loss* para o primeiro conjunto, e 4,4130% de acurácia e 4,4404 de *loss* para o segundo conjunto. Portanto, os resultados indicam que o método desenvolvido não é recomendado como uma alternativa viável para substituir os métodos convencionais até o momento.

Apesar dos valores de acurácia e *loss* relativamente baixos, o método apresentou diferen-

ças visuais consideráveis em relação ao método tradicional. Essas diferenças foram evidentes ao utilizar ferramentas de explicabilidade de modelos e ao examinar características das saídas das redes, conforme mencionado na Seção 7.1.1.

No contexto das segmentações semânticas, os resultados obtidos (Capítulo 7) revelaram comportamento diferente dos casos de classificação ao aplicar o método proposto nas arquiteturas. Os modelos U-Net com o método proposto apresentaram uma mIoU de 0,3333, acurácia de 86,77% e uma *loss* de 0,6659, enquanto o modelo com uso de *Max Pooling* obteve uma mIoU de 0,3367 e acurácia de 88,61%, mas com uma *loss* ligeiramente superior (0,6668). No entanto, todos os resultados mostram diferenças mínimas tanto em métricas quanto em qualidade visual das segmentações, sugerindo o BPCAPooling como uma alternativa válida ao método convencional, não fosse pela complexidade de código agregada, o que aumenta significativamente o tempo de execução durante as fases de treinamento e validação. Entretanto, este estudo abre uma possibilidade para que métodos tradicionais de extração de atributos e *manifold learning* possam ser investigados como uma camada de *pooling*, ou ainda, uma combinação de *poolings*.

8.1 Trabalhos Futuros

Para futuros desenvolvimentos, sugere-se: 1) a otimização do código desenvolvido, considerando a troca de *frameworks*, visando a redução da complexidade do código; 2) testes em conjuntos de dados adicionais apropriados para a tarefa de segmentação semântica; 3) o desenvolvimento do método BPCAUnpooling para preservar a espacialidade das amostras na fase de decodificação das U-Nets; e 4) avaliar o uso da técnica BPCAPooling em outras tarefas de processamento de imagens, como para filtragem de ruído, considerando que as redes neurais profundas normalmente eliminam a camada de *Max Pooling* para essa tarefa, visto que esse método acaba amplificando o ruído (Zhang et al., 2017) ou 5) introduzir a aplicação de ruído gaussiano como técnica de aumento de dados prévio para as etapas de treinamento e teste, visando avaliar o desempenho do BPCAPooling em relação aos métodos convencionais quando submetidos a essa técnica de aumento de dados, visto que o PCA é conhecido por separar imagem de ruído (Yi Zhou e Omar, 2011). Além disso, ainda nessa seção, serão discutidos detalhes que podem contribuir para trabalhos futuros.

Como mencionado na Seção 6.1.5, os experimentos de classificação empregaram os princípios de transferência de aprendizado, o que comumente oferece vantagens devido ao reaproveitamento de pesos de arquiteturas treinadas em grandes conjuntos de dados (Pan e Yang, 2010).

No entanto, embora as camadas de *pooling* não tenham parâmetros treináveis diretamente afetados pela retropropagação, elas influenciam diretamente as camadas de convolução subsequentes. Isso levanta a questão: “Será que o uso de transferência de aprendizado realmente contribuiu para a aplicação do método BPCAPooling nos experimentos realizados?” Uma vez que as camadas da rede passam a ficar com os pesos aquecidos, mas com o viés da utilização de um redutor de dimensionalidade que não olha para a informação espacial de um modo global, mas sim local, como é o caso do *Max Pooling* e *Avg Pooling*.

Além disso, outra questão pendente diz respeito a quais camadas precisam ser descongeladas para obter os melhores resultados possíveis com o método, como comentado na Seção 7.1.1, que demonstra maior progresso no quarto bloco convolucional.

Vale citar que a complexidade do método proposto é uma limitação. Um teste realizado com a arquitetura EfficientNetB0 (Tan e Le, 2019), que possui 17 camadas de *pooling*, substituindo os métodos convencionais por BPCAPooling resultou em um tempo estimado de treinamento da fase de aquecimento de aproximadamente 2.100 horas, de acordo com os padrões de desempenho comentados na Seção 7.1.

Para futuros trabalhos que visam aplicar o método BPCAPooling como camada de *pooling* para arquiteturas de classificação, sugere-se a não utilização de transferência de aprendizado para o método, mas sim a realização de fases de treinamento e validação mais extensas, além do uso de conjuntos de dados equivalentes ao ImageNet, o que é um desafio devido às necessidades de *hardware* que surgiriam. Uma sugestão adicional seria propor alternativas para otimizar a complexidade do código associado ao método BPCAPooling. Por fim, uma terceira ideia para trabalhos futuros, que também exigiria considerações sobre o *hardware* necessário para os experimentos, seria a aplicação de mais épocas no descongelamento de determinados blocos convolucionais durante o processo de *fine-tuning*, visando permitir a exploração mais profunda dos mapas de características e potencializar os aspectos de preservação espacial.

Um dos grandes desafios encontrados residiu na combinação dos *frameworks* TensorFlow e Keras. Dado que a abordagem proposta consistia no desenvolvimento de uma camada personalizada de *pooling*, erros de desenvolvimento eram comuns, afetando significativamente o tempo de implementação e outros aspectos mencionados nesta seção. Portanto, uma proposta para futuros trabalhos seria a exploração de diferentes *frameworks*, como PyTorch (Paszke et al., 2019). Além de potencialmente melhorar o desempenho do método, conforme discutido anteriormente.

A dificuldade enfrentada com os *frameworks* também prejudicou a aplicação do método em mais de um conjunto de dados para a tarefa de segmentação semântica. Uma excelente proposta futura seria estender os testes para um segundo conjunto de dados, preferencialmente com múltiplos objetos de interesse em cada exemplo. Conjuntos como *Cityscape* (Cordts et al., 2016) e *Lost and Found* (Pinggera et al., 2016) oferecem condições ideais para avaliar as capacidades de preservação de espacialidade do BPCAPooling em cenários mais complexos.

Um dos aspectos a serem explorados em pesquisas futuras diz respeito à análise da distribuição de energia dos componentes principais em relação aos autovalores. Em outras palavras, considerando que componentes principais com valores maiores indicam maior importância, seria interessante avaliar o quanto de informação está sendo preservado. Em alguns casos, pode ser vantajoso reorganizar e subdividir a informação em múltiplos componentes, garantindo assim que a quantidade de informação preservada seja significativa e relevante para a aplicação em questão.

8.2 Considerações Finais do Capítulo

Vale citar que a partir desse trabalho foi possível submeter um artigo intitulado *Exploring BPCA Pooling Layer in VGG-16 Network: A Comparative Study with Conventional Pooling Methods* na SIBGRAPI – *Conference on Graphics, Patterns and Images*, tendo todo o código utilizado documentado no Github⁸, alvejando aumentar o nível de contribuição com a comunidade científica.

Por fim, todo o código fonte utilizado neste trabalho está documentado e disponível como *open-source*⁹, buscando fomentar contribuições e incentivar a comunidade científica a avançar no tema abordado, facilitando futuras pesquisas nesta área a partir de um ponto inicial avançado e com algumas dificuldades mitigadas.

⁸Repositório Github do trabalho da conferência - https://github.com/Lucs1590/vgg_bpca

⁹Repositório Github do projeto - <https://github.com/Lucs1590/USEs-BPCA>

REFERÊNCIAS BIBLIOGRÁFICAS

- ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Disponível em: <<https://www.tensorflow.org/>>. Acessado em: 29 de nov. de 2023.
- ACHANTA, R.; SHAJI, A.; SMITH, K.; LUCCHI, A.; FUA, P.; SÜSTRUNK, S. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 34, n. 11, p. 2274–2281, 2012.
- ALAHMARI, S. S.; GOLDFOF, D. B.; MOUTON, P. R.; HALL, L. O. Challenges for the Repeatability of Deep Learning Models. *IEEE Access*, v. 8, p. 211860–211868, 2020.
- ALOM, M. Z.; YAKOPCIC, C.; HASAN, M.; TAHA, T. M.; ASARI, V. K. Recurrent residual U-Net for medical image segmentation. *Journal of medical imaging (Bellingham, Wash.)*, v. 6, n. 1, p. 1, 2019.
- ANGELOV, P. P.; SOARES, E. A.; JIANG, R.; ARNOLD, N. I.; ATKINSON, P. M. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, v. 11, n. 5, p. e1424, 2021.
- ANTONELLI, L.; DE SIMONE, V.; DI SERAFINO, D. A view of computational models for image segmentation. *Annali Dell'Universita'Di Ferrara*, v. 68, n. 2, p. 277–294, 2022.
- ARBELAEZ, P.; HARIHARAN, B.; GU, C.; GUPTA, S.; BOURDEV, L. D.; MALIK, J. Semantic segmentation using regions and parts. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, IEEE Computer Society, 2012, p. 3378–3385.
- ASGARI TAGHANAKI, S.; ABHISHEK, K.; COHEN, J. P.; COHEN-ADAD, J.; HAMARNEH, G. Deep semantic segmentation of natural and medical images: a review. *Artificial Intelligence Review*, v. 54, n. 1, p. 137–178, 2021.
- ATTA, M. A.; IMTIAZ, M.; HASSAN, A.; SAQIB, S. Image Segmentation by Using Threshold Techniques. *Lahore Garrison University Research Journal of Computer Science and Information Technology*, v. 2, n. 2, p. 1–6, 2018.

- BADRINARAYANAN, V.; KENDALL, A.; CIPOLLA, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 39, n. 12, p. 2481–2495, 2017.
- BARRETT, W. A.; MORTENSEN, E. N. Interactive live-wire boundary extraction. *Medical Image Analysis*, v. 1, n. 4, p. 331–341, 1997.
- BAYRAKDAR, I. S.; ORHAN, K.; AKARSU, S.; CELIK, Ö.; ATASOY, S.; PEKINCE, A.; YASA, Y.; BILGIR, E.; SAĞLAM, H.; ASLAN, A. F.; ODABAŞ, A. Deep-learning approach for caries detection and segmentation on dental bitewing radiographs. *Oral Radiology*, v. 1, p. 1–12, 2021.
- BENYAHIA, S.; MEFTAH, B.; LÉZORAY, O. Multi-features extraction based on deep learning for skin lesion classification. *Tissue and Cell*, v. 74, p. 101701, 2022.
- BISONG, E. Google Colaboratory. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, Berkeley, CA: Apress, p. 59–64, 2019.
- BOSSARD, L.; GUILLAUMIN, M.; VAN GOOL, L. Food-101—mining discriminative components with random forests. In: *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, Springer, 2014, p. 446–461.
- BOUREAU, Y.; PONCE, J.; LECUN, Y. A Theoretical Analysis of Feature Pooling in Visual Recognition. In: FÜRNKRANZ, J.; JOACHIMS, T., eds. *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, Omnipress, 2010, p. 111–118.
- BOYKOV, Y.; VEKSLER, O.; ZABIH, R. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 23, n. 11, p. 1222–1239, 2001.
- BRADSKI, G. The openCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, v. 25, n. 11, p. 120–123, 2000.
- BROOKS, J. COCO Annotator, 2019. Disponível em: <<https://github.com/jsbroks/coco-annotator/>>. Acessado em: 29 de nov. de 2023.

- CAESAR, H.; UIJLINGS, J. R. R.; FERRARI, V. COCO-Stuff: Thing and Stuff Classes in Context. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, IEEE Computer Society, 2018, p. 1209–1218.
- CARNEIRO, A. L. C.; SILVA, L. B.; SALVADEO, D. H. P. Efficient sign language recognition system and dataset creation method based on deep learning and image processing. *Thirteenth International Conference on Digital Image Processing (ICDIP 2021)*, v. 11878, p. 1187803, 2021a.
- CARNEIRO, A. L. C.; SILVA, L. D. B.; FAULIN, M. S. A. R. Artificial intelligence for detection and quantification of rust and leaf miner in coffee crop. *arXiv preprint arXiv:2103.11241*, 2021b.
- CAUCHY, A. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, v. 25, n. 1847, p. 536–538, 1847.
- CHEN, L.-C.; PAPANDREOU, G.; KOKKINOS, I.; MURPHY, K.; YUILLE, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 40, n. 4, p. 834–848, 2017.
- CHEN, L. C.; PAPANDREOU, G.; KOKKINOS, I.; MURPHY, K.; YUILLE, A. L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 40, n. 04, p. 834–848, 2018.
- CHOLLET, F. Keras, 2015. Disponível em: <<https://github.com/keras-team/keras>>. Acessado em: 29 de nov. de 2023.
- CHOLLET, F. *Deep learning with Python*. Simon and Schuster, 2021.
- CHRISTLEIN, V.; SPRANGER, L.; SEURET, M.; NICOLAOU, A.; KRAL, P.; MAIER, A. Deep generalized max pooling. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, p. 1090–1096, 2019.

- CHUNG, M.; LEE, M.; HONG, J.; PARK, S.; LEE, J.; LEE, J.; YANG, I. H.; LEE, J.; SHIN, Y. G. Pose-aware instance segmentation framework from cone beam CT images for tooth segmentation. *Computers in Biology and Medicine*, v. 120, p. 103720, 2020.
- CORDTS, M.; OMRAN, M.; RAMOS, S.; REHFELD, T.; ENZWEILER, M.; BENENSON, R.; FRANKE, U.; ROTH, S.; SCHIELE, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, p. 3213–3223.
- CSURKA, G.; LARLUS, D.; PERRONNIN, F. What is a good evaluation measure for semantic segmentation? In: BURGHARDT, T.; DAMEN, D.; MAYOL-CUEVAS, W. W.; MIRMEHDI, M., eds. *British Machine Vision Conference, BMVC 2013, Bristol, UK, September 9-13, 2013*, BMVA Press, 2013.
- DAHL, G. E.; SAINATH, T. N.; HINTON, G. E. Improving deep neural networks for LVCSR using rectified linear units and dropout. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2013, p. 8609–8613.
- DENG, J.; DONG, W.; SOCHER, R.; LI, L.; LI, K.; LI, F. ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, IEEE Computer Society, 2009, p. 248–255.
- DEVI, S. G.; SELVAM, K.; RAJAGOPALAN, S. An abstract to calculate big o factors of time and space complexity of machine code. *IET Conference Publications*, v. 2011, p. 844–847, 2011.
- DOMINGOS, P. A few useful things to know about machine learning. *Communications of the ACM*, v. 55, n. 10, p. 78–87, 2012.
- DU, G.; CAO, X.; LIANG, J.; CHEN, X.; ZHAN, Y.; CAO, X.; LIANG, J.; CHEN, X.; ZHAN, Y. Medical Image Segmentation based on U-Net: A Review. *Journal of Imaging Science and Technology*, v. 64, n. 2, p. 1–12, 2020.
- DUNHAM, M. H. *Data mining: Introductory and advanced topics*. Pearson Education India, 2006.

- EVERINGHAM, M.; VAN GOOL, L.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, v. 88, n. 2, p. 303–338, 2010.
- FAJAR, A.; SARNO, R.; FATICHAH, C.; SUSILO, R. I.; PANGESTU, G. Cyclical Learning Rate Optimization on Deep Learning Model for Brain Tumor Segmentation. *IEEE Access*, 2023.
- FALCAO, A. X.; STOLFI, J.; DE ALENCAR LOTUFO, R. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26, n. 1, p. 19–29, 2004.
- FALCÃO, A. X. *Paradigmas de Segmentacao de Imagens Guiada pelo Usuario: Live Wire, Live-Lane e 3D-Live-Wire*. Tese de Doutorado, Institute of Computing-IC State University of Campinas-UNICAMP, 1997.
- FEL, T.; HERVIER, L.; VIGOUROUX, D.; POCHE, A.; PLAKOO, J.; CADENE, R.; CHALVIDAL, M.; COLIN, J.; BOISSIN, T.; BETHUNE, L.; PICARD, A.; CLAIRE, N.; GARDES, L.; FLANDIN, G.; SERRE, T. Xplique: A Deep Learning Explainability Toolbox. *Workshop on Explainable Artificial Intelligence for Computer Vision (CVPR)*, 2022.
- FELZENSZWALB, P. F.; HUTTENLOCHER, D. P. Efficient graph-based image segmentation. *International Journal of Computer Vision*, v. 59, n. 2, p. 167–181, 2004.
- FISCHER, V.; KUMAR, M. C.; METZEN, J. H.; BROX, T. Adversarial Examples for Semantic Image Segmentation. *ArXiv preprint*, v. abs/1703.01101, 2017.
- FLEYEH, H. Color detection and segmentation for road and traffic signs. In: *2004 IEEE Conference on Cybernetics and Intelligent Systems*, 2004, p. 808–813.
- GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. 1 ed. O'Reilly Media, 574 p., 2017.
- GHOSH, S.; DAS, N.; DAS, I.; MAULIK, U. Understanding Deep Learning Techniques for Image Segmentation. *ArXiv preprint*, v. abs/1907.06119, 2019.

- GIBSON, E.; GIGANTI, F.; HU, Y.; BONMATI, E.; BANDULA, S.; GURUSAMY, K.; DAVIDSON, B.; PEREIRA, S. P.; CLARKSON, M. J.; BARRATT, D. C. Automatic Multi-Organ Segmentation on Abdominal CT with Dense V-Networks. *IEEE Transactions on Medical Imaging*, v. 37, n. 8, p. 1822–1834, 2018.
- GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings*, 2011, p. 315–323.
- GONZALEZ, R. C.; WOODS, R. C. *Processamento digital de imagens*, v. 3. Pearson Education, 624 p., 2009.
- GONÇALVES, D. N.; DE MOARES WEBER, V. A.; PISTORI, J. G. B.; DA COSTA GOMES, R.; DE ARAUJO, A. V.; PEREIRA, M. F.; GONÇALVES, W. N.; PISTORI, H. Carcass image segmentation using CNN-based methods. *Information Processing in Agriculture*, v. 8, n. 4, p. 560–572, 2021.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. MIT press, 2016.
- GUNNING, D.; STEFIK, M.; CHOI, J.; MILLER, T.; STUMPF, S.; YANG, G. Z. XAI—Explainable artificial intelligence. *Science Robotics*, v. 4, n. 37, 2019.
- GUO, Y.; LIU, Y.; GEORGIOU, T.; LEW, M. S. A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, v. 7, n. 2, p. 87–93, 2018.
- HAHNIOSE, R. H.; SARPESHKAR, R.; MAHOWALD, M. A.; DOUGLAS, R. J.; SEUNG, H. S. Digital selection and analogue amplification coexist in a cortex- inspired silicon circuit. *Nature*, v. 405, n. 6789, p. 947–951, 2000.
- HANS-HERMANN, B. Origins and extensions of the k-means algorithm in cluster analysis. *Journal Electronique d’Histoire des Probabilités et de la Statistique Electronic Journal for History of Probability and Statistics*, v. 4, n. 2, 2008.
- HAO, S.; ZHOU, Y.; GUO, Y. A Brief Survey on Semantic Segmentation with Deep Learning. *Neurocomputing*, v. 406, p. 302–321, 2020.

- HARALICK, R. M.; SHAPIRO, L. G. Image segmentation techniques. *Computer Vision, Graphics, & Image Processing*, v. 29, n. 1, p. 100–132, 1985.
- HARRIS, C. R.; MILLMAN, K. J.; DER WALT, S. J. v.; GOMMERS, R.; VIRTANEN, P.; COURNAPEAU, D.; WIESER, E.; TAYLOR, J.; BERG, S.; SMITH, N. J.; KERN, R.; HOYER, M. P.; STEPHAN; VAN KERKWIJK, M. H.; BRETT, M.; HALDANE, A.; DEL RÍO, J. F.; WIEBE, M.; PETERSON, P.; GÉRARD-MARCHANT, P.; SHEPPARD, K.; REDDY, T.; WECKESSER, W.; ABBASI, H.; GOHLKE, C.; OLIPHANT, T. E. Array programming with NumPy. *Nature*, v. 585, n. 7825, p. 357–362, 2020.
- HE, K.; ZHANG, X.; REN, S.; SUN, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 37, n. 9, p. 1904–1916, 2015.
- HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, p. 770–778.
- HESAMIAN, M. H.; JIA, W.; HE, X.; KENNEDY, P. Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges. *Journal of Digital Imaging*, v. 32, n. 4, p. 582–596, 2019.
- HINTON, G.; SRIVASTAVA, N.; SWERSKY, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *University of Toronto*, v. 14, n. 8, p. 2, 2012.
- HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M.; ADAM, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *ArXiv preprint*, v. abs/1704.04861, 2017.
- HUANG, G.; LIU, Z.; VAN DER MAATEN, L.; WEINBERGER, K. Q. Densely Connected Convolutional Networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, p. 2261–2269.
- HYTTINEN, J.; FÄLT, P.; JÄSBERG, H.; KULLAA, A.; HAUTA-KASARI, M. Oral and Dental Spectral Image Database—ODSI-DB. *Applied Sciences 2020, Vol. 10, Page 7246*, v. 10, n. 20, p. 7246, 2020.

IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: BACH, F. R.; BLEI, D. M., eds. *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, JMLR.org, 2015, p. 448–456 (*JMLR Workshop and Conference Proceedings*, v.37).

JADER, G.; FONTINELI, J.; RUIZ, M.; ABDALLA, K.; PITHON, M.; OLIVEIRA, L. Deep Instance Segmentation of Teeth in Panoramic X-Ray Images. *Proceedings - 31st Conference on Graphics, Patterns and Images, SIBGRAPI 2018*, p. 400–407, 2019.

KANDEL, I.; CASTELLI, M. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, v. 6, n. 4, p. 312–315, 2020.

KAYMAK, C.; UCAR, A. Semantic image segmentation for autonomous driving using fully convolutional networks. In: *2019 International Conference on Artificial Intelligence and Data Processing Symposium, IDAP 2019*, IEEE, Institute of Electrical and Electronics Engineers Inc., 2019, p. 1–8.

KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. *CoRR*, v. abs/1412.6980, 2014.

KIRILLOV, A.; HE, K.; GIRSHICK, R. B.; ROTHER, C.; DOLLÁR, P. Panoptic Segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, p. 9404–9413.

KOTU, V.; DESHPANDE, B. *Data science: concepts and practice*. Morgan Kaufmann, 2018.

KRIZHEVSKY, A.; HINTON, G. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.

KRIZHEVSKY, A.; NAIR, V.; HINTON, G. The CIFAR-10 dataset. *Department of Computer Science of, University of Toronto*, v. 55, n. 5, 2014.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: BARTLETT, P. L.; PEREIRA, F. C. N.; BURGESS, C.

J. C.; BOTTOU, L.; WEINBERGER, K. Q., eds. *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, 2012, p. 1106–1114.

KUGELMAN, J.; ALLMAN, J.; READ, S. A.; VINCENT, S. J.; TONG, J.; KALLONIATIS, M.; CHEN, F. K.; COLLINS, M. J.; ALONSO-CANEIRO, D. A comparison of deep learning U-Net architectures for posterior segment OCT retinal layer segmentation. *Scientific Reports* 2022 12:1, v. 12, n. 1, p. 1–14, 2022.

KUMAR, R. L.; KAKARLA, J.; ISUNURI, B. V.; SINGH, M. Multi-class brain tumor classification using residual network and global average pooling. *Multimedia Tools and Applications*, v. 80, n. 9, p. 13429–13438, 2021.

KUNCHEVA, L. I.; FAITHFULL, W. J. PCA feature extraction for change detection in multi-dimensional unlabeled data. *IEEE Transactions on Neural Networks and Learning Systems*, v. 25, n. 1, p. 69–80, 2014.

LAI, M. Deep Learning for Medical Image Segmentation. *ArXiv preprint*, v. abs/1505.02000, 2015.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2323, 1998.

LECUN, Y.; CORTES, C.; BURGESS, C. J. MNIST handwritten digit database. *ATT Labs.*, v. 2, disponível em: <<http://yann.lecun.com/exdb/mnist>>. Acessado em: 29 de nov. de 2023, 2010.

LECUN, Y.; HAFFNER, P.; BOTTOU, L.; BENGIO, Y. Object recognition with gradient-based learning. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 1999, p. 319–345.

LEE, H. S.; KIM, K. Simultaneous traffic sign detection and boundary estimation using convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, v. 19, n. 5, p. 1652–1663, 2018.

- LEE, J. S. Digital image smoothing and the sigma filter. *Computer Vision, Graphics, and Image Processing*, v. 24, n. 2, p. 255–269, 1983.
- LIANG, H.; SUN, X.; SUN, Y.; GAO, Y. Text feature extraction based on deep learning: a review. *Eurasip Journal on Wireless Communications and Networking*, v. 2017, n. 1, p. 1–12, 2017.
- LIU, C.; YUEN, J.; TORRALBA, A. SIFT flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 33, n. 5, p. 978–994, 2011.
- LIU, P.; ZHANG, H.; LIAN, W.; ZUO, W. Multi-Level Wavelet Convolutional Neural Networks. *IEEE Access*, v. 7, p. 74973–74985, 2019.
- LIU, W.; RABINOVICH, A.; BERG, A. C. ParseNet: Looking Wider to See Better. *ArXiv preprint*, v. abs/1506.04579, 2015.
- LIU, X.; HAN, Y.; BAI, S.; GE, Y.; WANG, T.; HAN, X.; LI, S.; YOU, J.; LU, J. Importance-Aware Semantic Segmentation in Self-Driving with Discrete Wasserstein Training. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press, 2020, p. 11629–11636.
- LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, IEEE Computer Society, 2015, p. 3431–3440.
- MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, p. 281–297.
- MAHDIANPARI, M.; SALEHI, B.; REZAEI, M.; MOHAMMADIMANESH, F.; ZHANG, Y. Very Deep Convolutional Neural Networks for Complex Land Cover Mapping Using Multispectral Remote Sensing Imagery. *Remote Sensing*, v. 10, n. 7, p. 1119, 2018.
- MAHMUD, M. S.; RAHMAN, M. M.; AKHTAR, M. N. Improvement of K-means clustering algorithm with better initial centroids based on weighted average. In: *2012 7th*

International Conference on Electrical and Computer Engineering, ICECE 2012, 2012, p. 647–650.

MALKANTHI, C.; DISSANAYAKE, M. B. Brain Tumor Boundary Segmentation of MR Imaging using Spatial Domain Image Processing. *International Journal for Innovation Education and Research*, v. 5, n. 10, p. 1–9, 2017.

MARTINEZ, A.; BENAVENTE, R. *The AR Face Database: CVC Technical Report, 24*. Computer Vision Center, 1998.

McCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, v. 5, n. 4, p. 115–133, 1943.

MILLMAN, K. J.; AIVAZIS, M. Python for scientists and engineers. *Computing in Science and Engineering*, v. 13, n. 2, p. 9–12, 2011.

MINAEI, S.; BOYKOV, Y.; PORIKLI, F.; PLAZA, A.; KEHTARNAVAZ, N.; TERZOPOULOS, D. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 44, n. 7, p. 3523–3542, 2021a.

MINAEI, S.; KALCHBRENNER, N.; CAMBRIA, E.; NIKZAD, N.; CHENAGHLU, M.; GAO, J. Deep Learning–based Text Classification. *ACM Computing Surveys*, v. 54, n. 3, p. 1–40, 2021b.

MOHAN, R.; VALADA, A. EfficientPS: Efficient Panoptic Segmentation. *ArXiv preprint*, v. abs/2004.02307, 2020.

MONTESINOS LÓPEZ, O. A.; MONTESINOS LÓPEZ, A.; CROSSA, J. Artificial Neural Networks and Deep Learning for Genomic Prediction of Continuous Outcomes. *Multivariate Statistical Machine Learning Methods for Genomic Prediction*, p. 427–476, 2022.

MÜLLER, D.; SOTO-REY, I.; KRAMER, F. Towards a guideline for evaluation metrics in medical image segmentation. *BMC Research Notes*, v. 15, n. 1, p. 1–8, 2022.

MUTHUKRISHNAN, R.; RADHA, M. Edge detection techniques for image segmentation. *International Journal of Computer Science & Information Technology*, v. 3, n. 6, p. 259, 2011.

- NATHAN SILBERMAN DEREK HOIEM, P. K.; FERGUS, R. Indoor Segmentation and Support Inference from RGBD Images. In: *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, Springer, 2012, p. 746–760.
- NEUHOLD, G.; OLLMANN, T.; BULÒ, S. R.; KONTSCHIEDER, P. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, IEEE Computer Society, 2017, p. 5000–5009.
- NGUYEN, T. T.; LARRIVÉE, N.; LEE, A.; BILANIUK, O.; DURAND, R. Use of artificial intelligence in dentistry: current clinical trends and research advances. *J Can Dent Assoc*, v. 87, n. 17, p. 1488–2159, 2021.
- NIRKIN, Y.; MASI, I.; TUAN, A. T.; HASSNER, T.; MEDIONI, G. On face segmentation, face swapping, and face perception. In: *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, IEEE, 2018, p. 98–105.
- OLIPHANT, T. E. Python for scientific computing. *Computing in Science and Engineering*, v. 9, n. 3, p. 10–20, 2007.
- OTSU, N. Threshold Selection Method From Gray-Level Histograms. *IEEE Trans Syst Man Cybern*, v. SMC-9, n. 1, p. 62–66, 1979.
- ÖZDEMİR, C. Avg-topk: A new pooling method for convolutional neural networks. *Expert Systems with Applications*, v. 223, p. 119892, 2023.
- OZTURK, O.; SARITÜRK, B.; SEKER, D. Z. Comparison of Fully Convolutional Networks (FCN) and U-Net for Road Segmentation from High Resolution Imageries. *International Journal of Environment and Geoinformatics*, v. 7, n. 3, p. 272–279, 2020.
- PADILLA, R.; PASSOS, W. L.; DIAS, T. L.; NETTO, S. L.; DA SILVA, E. A. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, v. 10, n. 3, p. 279, 2021.
- PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, v. 22, n. 10, p. 1345–1359, 2010.

- PAN, Z.; EMARU, T.; RAVANKAR, A.; KOBAYASHI, Y. Applying Semantic Segmentation to Autonomous Cars in the Snowy Environment. *ArXiv preprint*, v. abs/2007.12869, 2020.
- PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A. Deep Face Recognition. In: XIE, X.; JONES, M. W.; TAM, G. K. L., eds. *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, BMVA Press, 2015, p. 41.1–41.12.
- PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A.; JAWAHAR, C. V. Cats and dogs. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, IEEE Computer Society, 2012, p. 3498–3505.
- PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DeVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; LERER, A. Automatic differentiation in PyTorch. *Neural Information Processing Systems*, p. 8024–8035, 2019.
- PAUL, A.; CHAKI, N. Dimensionality Reduction of Hyperspectral Images Using Pooling. *Pattern Recognition and Image Analysis*, v. 29, n. 1, p. 72–78, 2019.
- PEARSON, K. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, v. 2, n. 11, p. 559–572, 1901.
- PEDRINI, H.; SCHWARTZ, W. R. *Análise de imagens digitais: princípios, algoritmos e aplicações*. Thomson Learning, 2008.
- PERKEL, J. M. Programming: Pick up Python. *Nature*, v. 518, n. 7537, p. 125–126, 2015.
- PFISTER, M.; SCHÜTZENBERGER, K.; PFEIFFENBERGER, U.; MESSNER, A.; CHEN, Z.; ARANHA, V.; SANTOS, D.; PUCHNER, S.; GARHÖFER, G.; SCHMETTERER, L.; GRÖSCHL, M.; WERKMEISTER, R. M. Automated segmentation of dermal fillers in OCT images of mice using convolutional neural networks. *Biomedical Optics Express*, Vol. 10, Issue 3, pp. 1315–1328, v. 10, n. 3, p. 1315–1328, 2019.
- PINGGERA, P.; RAMOS, S.; GEHRIG, S. K.; FRANKE, U.; ROTHER, C.; MESTER, R. Lost and Found: detecting small road hazards for self-driving vehicles. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 1099–1106, 2016.

- POLYAK, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, v. 4, n. 5, p. 1–17, 1964.
- PONTI, M. A.; DA COSTA, G. B. P. Como funciona o deep learning. *ArXiv preprint*, v. abs/1806.07908, 2018.
- PRAVITASARI, A. A.; IRIAWAN, N.; ALMUHAYAR, M.; AZMI, T.; IRHAMAH, I.; FITHRIASARI, K.; PURNAMI, S. W.; FERRIASTUTI, W. UNet-VGG16 with transfer learning for MRI-based brain tumor segmentation. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, v. 18, n. 3, p. 1310–1318, 2020.
- PRECHELT, L. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, v. 11, n. 4, p. 761–767, 1998a.
- PRECHELT, L. Early Stopping - But When? *Neural Networks: Tricks of the Trade*, p. 55–69, 1998b.
- PRIYA, B.; KUMAR, J. A.; KHUSHBOO, A.; NEHA, G.; VANSHIKA, B.; ABHISHEK, K. AI Revolutionizing Health Care: A Descriptive Review. *Journal of Advanced Scientific Research*, v. 12, n. 2, p. 22–34, 2021.
- REN, X.; MALIK, J. Learning a classification model for segmentation. *Proceedings of the IEEE International Conference on Computer Vision*, v. 1, p. 10–17, 2003.
- RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: KRISHNAPURAM, B.; SHAH, M.; SMOLA, A. J.; AGGARWAL, C. C.; SHEN, D.; RASTOGI, R., eds. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, ACM, 2016, p. 1135–1144.
- RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 9351, p. 234–241, 2015.
- ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958.

- ROTHER, C.; KOLMOGOROV, V.; BLAKE, A. "GrabCut- interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, v. 23, n. 3, p. 309–314, 2004.
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986.
- SABRI, N.; HAMED, H. N. A.; IBRAHIM, Z.; IBRAHIM, K. A comparison between average and max-pooling in convolutional neural network for scoliosis classification. *International Journal of Advanced Trends in Computer Science and Engineering*, v. 9, 2020.
- SALVADEO, D. H.; MASCARENHAS, N. D.; MOREIRA, J.; LEVADA, A. L.; CORRÊA, D. C. Improving face recognition performance using RBPCA MaxLike and information fusion. *Computing in Science and Engineering*, v. 13, n. 3, p. 14–21, 2011.
- SANG, S.; ZHOU, Y.; ISLAM, M. T.; XING, L. Small-Object Sensitive Segmentation Using Across Feature Map Attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 45, n. 5, p. 6289–6306, 2023.
- SCHEUER, C.; BOOT, E.; CARSE, N.; CLARDY, A.; GALLAGHER, J.; HECK, S.; MARRON, S.; MARTINEZ-ALVAREZ, L.; MASARYKOVA, D.; MCMILLAN, P.; MURPHY, F.; STEEL, E.; EKDOM, H. V.; VECCHIONE, H. Multi-level Context Gating of Embedded Collective Knowledge for Medical Image Segmentation. *Physical Education and Sport for Children and Youth with Special Needs Researches – Best Practices – Situation*, p. 343–354, 2020.
- SCHNEIDER, M. J.; BELLON, O. R. P.; ARAKI, H. Experimentos em fusão de imagens de alta resolução. *Boletim de Ciências Geodésicas*, v. 9, n. 1, 2003.
- SCHOPPE, O.; PAN, C.; CORONEL, J.; MAI, H.; RONG, Z.; TODOROV, M. I.; MÜSKES, A.; NAVARRO, F.; LI, H.; ERTÜRK, A.; MENZE, B. H. Deep learning-enabled multi-organ segmentation in whole-body mouse scans. *Nature Communications*, v. 11, n. 1, p. 1–14, 2020.
- SCHWENDICKE, F.; SAMEK, W.; KROIS, J. Artificial Intelligence in Dentistry: Chances and Challenges. *Journal of Dental Research*, v. 99, n. 7, p. 769–774, 2020.

- SEAL, A.; DAS, A.; SEN, P. Watershed: an image segmentation approach. *International Journal of Computer Science and Information Technologies*, v. 6, n. 3, p. 2295–2297, 2015.
- SENIOR, A.; HEIGOLD, G.; RANZATO, M.; YANG, K. An empirical study of learning rates in deep neural networks for speech recognition. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, p. 6724–6728, 2013.
- SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, v. 6, n. 1, p. 1–48, 2019.
- SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: BENGIO, Y.; LECUN, Y., eds. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- SMITH, L. N. Cyclical learning rates for training neural networks. *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, p. 464–472, 2017.
- SONG, F.; GUO, Z.; MEI, D. Feature selection using principal component analysis. *Proceedings - 2010 International Conference on System Science, Engineering Design and Manufacturing Informatization, ICSEM 2010*, v. 1, p. 27–30, 2010.
- STEVENS, C. F. The Neuron. *Scientific American*, v. 241, n. 3, p. 54–65, 1979.
- STEWART, G. W. On the Early History of the Singular Value Decomposition. *Society for Industrial and Applied Mathematics*, v. 35, n. 4, p. 551–566, 2006.
- SU, X.; LIANG, H.; YAO, G.; LI, H.; LI, S. Small-scale Image Semantic Segmentation Method Based on Multilevel Superposition and Enhancement Fusion. *Proceedings of the 33rd Chinese Control and Decision Conference, CCDC 2021*, p. 1502–1507, 2021.
- SULTANA, F.; SUFIAN, A.; DUTTA, P. Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey. *Knowledge-Based Systems*, v. 201-202, p. 106062, 2020.

- SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S. E.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, IEEE Computer Society, 2015, p. 1–9.
- TAN, M.; LE, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: CHAUDHURI, K.; SALAKHUTDINOV, R., eds. *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, PMLR, 2019, p. 6105–6114 (*Proceedings of Machine Learning Research*, v.97).
- TANG, S.; ZHU, Y.; YUAN, S. An improved convolutional neural network with an adaptable learning rate towards multi-signal fault diagnosis of hydraulic piston pump. *Advanced Engineering Informatics*, v. 50, p. 101406, 2021.
- TEICHMANN, M.; WEBER, M.; ZOELLNER, M.; CIPOLLA, R.; URTASUN, R. Multinet: Real-time joint semantic reasoning for autonomous driving. In: *IEEE Intelligent Vehicles Symposium*, IEEE, 2018, p. 1013–1020.
- THE APACHE SOFTWARE FOUNDATION Apache License 2.0, 2004. Disponível em: <<https://www.apache.org/licenses/>>. Acessado em: 29 de nov. de 2023.
- VAILLANT, R. Original approach for the localisation of objects in images. *IEE Proceedings - Vision, Image, and Signal Processing*, v. 141, n. 4, p. 245, 1994.
- VAN DER WALT, S.; COLBERT, S. C.; VAROQUAUX, G. The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, v. 13, n. 2, p. 22–30, 2011.
- VAN ROSSUM, G.; DRAKE, F. L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.
- VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISSER, Ł.; POLOSUKHIN, I. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017.

- VEDALDI, A.; SOATTO, S. Quick shift and kernel methods for mode seeking. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 5305 LNCS, n. PART 4, p. 705–718, 2008.
- VIITANIEMI, V.; LAAKSONEN, J. Techniques for image classification, object detection and object segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 5188 LNCS, p. 231–234, 2008.
- VILARES FERRO, M.; DOVAL MOSQUERA, Y.; RIBADAS PENA, F. J.; DARRIBA BILBAO, V. M. Early stopping by correlating online indicators in neural networks. *Neural Networks*, v. 159, p. 109–124, 2023.
- WANG, P.; CHEN, P.; YUAN, Y.; LIU, D.; HUANG, Z.; HOU, X.; COTTRELL, G. Understanding Convolution for Semantic Segmentation. *ArXiv preprint*, v. abs/1702.08502, 2017.
- WANG, Z.; BOVIK, A. C.; SHEIKH, H. R.; SIMONCELLI, E. P. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, v. 13, n. 4, p. 600–612, 2004.
- WEISS, K.; KHOSHGOFTAAR, T. M.; WANG, D. D. A survey of transfer learning. *Journal of Big Data*, v. 3, n. 1, p. 1–40, 2016.
- WERBOS, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Tese de Doutorado, Harvard University, 1974.
- WIDROW, B.; LEHR, M. A. 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. *Proc. IEEE*, v. 78, p. 1415–1442, 1990.
- WITHEY, D. J.; KOLES, Z. J. A review of medical image segmentation: methods and available software. *International Journal of Bioelectromagnetism*, v. 10, n. 3, p. 125–148, 2008.
- WOLD, S.; ESBENSEN, K.; GELADI, P. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, v. 2, n. 1-3, p. 37–52, 1987.
- YANG, S.; JIN, G.; WANG, R.; KE, Y. Interactive Tooth Segmentation Method of Dental Model based on Geodesic. *2nd International Conference on Mechanical, Electronic and Computer Engineering*, p. 422–426, 2016.

- YANOWITZ, S. D.; BRUCKSTEIN, A. M. A new method for image segmentation. *Computer Vision, Graphics, & Image Processing*, v. 46, n. 1, p. 82–95, 1989.
- YI, F.; MOON, I. Image segmentation: A survey of graph-cut methods. In: *2012 International Conference on Systems and Informatics, ICSAI 2012*, 2012, p. 1936–1941.
- YI ZHOU, A. M.; OMAR, M. A. Principal Component Analysis-Based Image Fusion Routine with Application to Automotive Stamping Split Detection. *Research in Nondestructive Evaluation*, v. 22, n. 2, p. 76–91, 2011.
- YOSINSKI, J.; CLUNE, J.; BENGIO, Y.; LIPSON, H. How transferable are features in deep neural networks? In: GHAHRAMANI, Z.; WELLING, M.; CORTES, C.; LAWRENCE, N. D.; WEINBERGER, K. Q., eds. *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014, p. 3320–3328.
- YUHENG, S.; HAO, Y. Image Segmentation Algorithms Overview. *ArXiv preprint*, v. abs/1707.02051, 2017.
- ZAFAR, A.; AAMIR, M.; MOHD NAWI, N.; ARSHAD, A.; RIAZ, S.; ALRUBAN, A.; DUTTA, A. K.; ALMOTAIRI, S. A Comparison of Pooling Methods for Convolutional Neural Networks. *Applied Sciences 2022, Vol. 12, Page 8643*, v. 12, n. 17, p. 8643, 2022.
- ZANJANI, F. G.; POURTAHERIAN, A.; ZINGER, S.; MOIN, D. A.; CLAESSEN, F.; CHERICI, T.; PARINUSSA, S.; DE WITH, P. H. Mask-MCNet: Tooth instance segmentation in 3D point clouds of intra-oral scans. *Neurocomputing*, v. 453, p. 286–298, 2021.
- ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. *Computer Vision–European Conference on Computer Vision 2014*, p. 818–833, 2014.
- ZHANG, H.; DANA, K. J.; SHI, J.; ZHANG, Z.; WANG, X.; TYAGI, A.; AGRAWAL, A. Context Encoding for Semantic Segmentation. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, IEEE Computer Society, 2018, p. 7151–7160.
- ZHANG, K.; ZUO, W.; CHEN, Y.; MENG, D.; ZHANG, L. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, v. 26, n. 7, p. 3142–3155, 2017.

ZHANG, S.; NIU, Y. LcmUNet: A Lightweight Network Combining CNN and MLP for Real-Time Medical Image Segmentation. *Bioengineering* 2023, Vol. 10, Page 712, v. 10, n. 6, p. 712, 2023.

ZHAO, R.; NIU, X.; LUK, W. Automatic Optimising CNN with Depthwise Separable Convolution on FPGA. *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, p. 285–285, 2018.

ZHENG, G.; NOLTE, L. P.; FERGUSON, S. J. Scaled, patient-specific 3D vertebral model reconstruction based on 2D lateral fluoroscopy. *International Journal of Computer Assisted Radiology and Surgery*, v. 6, n. 3, p. 351–366, 2011.