Identificação de Espécies de Animais do Pantanal usando Imagens Térmicas e Redes Neurais Convolucionais

Mauro dos Santos de Arruda

Universidade Federal de Mato Grosso do Sul
Faculdade de Computação

Mauro dos Santos de Arruda

Orientador: *Prof. Dr. Wesley Nunes Gonçalves*
Co-orientador: *Prof. Dr. Bruno Brandoli Machado*

Dissertação de Mestrado apresentada ao Curso de Pós Graduação em Ciência da Computação da Universidade Federal de Mato Grosso do Sul, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

UFMS - Ponta Porã
Setembro/2018

Agradecimentos

A Deus por ter me dado força e saúde para superar todos os desafios.

Ao meu orientador e coorientador, Prof. Wesley Nunes Gonçalves e, Prof. Bruno Brandoli Machado, pelo suporte, correções e incentivos.

Ao apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao Centro Tecnológico de Eletrônica e Informática (CTEI) pela disponibilização da câmera térmica, a NVidia pela doação da placa gráfica, e ao Centro de Reabilitação de Animais Silvestres (CRAS) e Instituto de Meio Ambiente de Mato Grosso do Sul (IMASUL) pela disponibilização do local de captura das imagens.

À minha família, pelo apoio e incentivo durante minha formação. E a todos amigos e familiares que, direta ou indiretamente, fizeram parte dessa jornada, o meu muito obrigado.

Abstract

Pantanal is one of the most important biomes of the world, with a large number of wild animals species, some of them are in extinction. The identification of these animals is extremely important for estimation of species population within the Pantanal. However, digital processing techniques for automatic identification of the animals have faced a great challenge due to changes related to lighting and pose conditions present in the images. In this work, we propose a new approach based on convolutional neural networks using the architecture of the technique *Fast R-CNN*. For the detection of regions of interest, our approach uses the segmentation of thermal and RGB images by means of the SLIC algorithm. Such regions are then projected onto the network feature map, and then a maximum pooling is performed to adjust the final size of the features with the fully-connected layer. Experimental results and statistics tests showed a gain of $5,5\%$ on the average performance when compared to the implementations of the *Fast R-CNN* technique. Our approach has proved to be effective for a large image dataset with 8 classes.

Keywords: Identification of Pantanal animal species, Convolutional neural networks and Thermal images.

Resumo

O Pantanal é um dos biomas mais importantes do mundo, com diversas espécies de animais silvestres, algumas delas ameaçadas de extinção. A identificação desses animais é de extrema importância para estimativa populacional por espécie dentro do Pantanal. Entretanto, as técnicas de imagens digitais para identificação de espécies de animais têm enfrentado um grande desafio devido às mudanças de iluminação e poses presentes nas imagens capturadas. Nesta dissertação de mestrado, nós propomos uma nova abordagem baseada em redes neurais convolucionais usando a arquitetura da técnica *Fast R-CNN*. Para a detecção de regiões de interesse, a nossa abordagem usa a segmentação de imagens térmicas e RGB usando o algoritmo SLIC. Estas regiões são então projetadas no mapa de características da rede neural convolucional, e depois uma subamostragem máxima é realizada para ajustar o tamanho final das características com a camada totalmente conectada. Resultados experimentais e teste estatísticos mostraram que a nossa abordagem obteve um ganho de $5,5\%$ no desempenho médio, quando comparado à técnica *Fast R-CNN*. A nossa abordagem mostrou-se ser eficaz para um conjunto de imagens com 8 classes.

Palavras-chave: Identificação de espécies animais do Pantanal, Redes neurais convolucionais e Imagens térmicas.

Sumário

Sumário xii
Lista de Figuras xiv
Lista de Tabelas xv

1 Introdução **1**
1.1 Contexto e Motivação 1
1.2 Objetivos 3
1.3 Organização do Texto 4

2 Trabalhos Correlatos **5**
2.1 Reconhecimento de Espécies de Animais 5
2.2 Redes Neurais Convolucionais (*Convolutional Neural Networks - CNN*) 7
2.3 Imagens Térmicas 8

3 Referencial Teórico **11**
3.1 Segmentação de Imagens usando Superpixels 11
3.2 Agrupamento de Pixels usando Componentes Conectados 14
3.3 Redes Neurais Convolucionais 16
3.3.1 Camada Convolucional 17
3.3.2 Camada de Normalização 21
3.3.3 Camada de Subamostragem 22
3.3.4 Camada Totalmente Conectada 23
3.3.5 Classificador Softmax 25
3.4 Arquiteturas de Redes Neurais Convolucionais 26
3.4.1 Rede AlexNet 26
3.4.2 Rede VGGNet 28
3.5 Redes Neurais Convolucionais Baseadas em Regiões de Interesse 30
3.5.1 Detecção de Regiões de Interesse 30

3.5.2 Algoritmos de Redes Neurais Convolucionais Baseadas em Regiões de Interesse 33

4 Uma Nova Abordagem para Identificação de Animais Silvestres 37
4.1 Detecção de Animais Usando Imagens Térmicas 37
4.2 Classificação dos Animais com Redes Neurais Convolucionais . . 42

5 Experimentos e Resultados 47
5.1 Conjunto de Imagens 47
5.1.1 Conjunto de Treinamento 47
5.1.2 Conjunto de Teste 48
5.2 Métricas de avaliação 49
5.3 Experimentos 52
5.3.1 Resultados dos Experimentos 54
5.3.2 Testes Estatísticos 56
5.3.3 Discussão dos Resultados 59
5.3.4 Avaliação Qualitativa 65

6 Conclusão 69

Referências 77

Lista de Figuras

1.1 Exemplo de imagem térmica e visual. 3

2.1 Exemplos de imagens térmicas usadas no monitoramento de animais. 9

3.1 Exemplos de monitoramento de animais usando a técnica SLIC. . 14
3.2 Representação da vizinhança de 4 ou 8 pixels. 15
3.3 Exemplo do funcionamento do algoritmo de componentes conectados com vizinhança de 8 pixels. 15
3.4 Estrutura da rede neural convolucional. 17
3.5 Representação dos principais componentes da camada convolucional. 18
3.6 Exemplo numérico do processo de convolução. 19
3.7 Exemplo visual da saída da função sigmoide e tangente hiperbólica. 21
3.8 Exemplo numérico da operação de subamostragem máxima. . . . 23
3.9 Exemplo da execução da subamostragem em uma imagem de duas araras-canindé. 24
3.10 Exemplo do mapeamento realizado pela camada totalmente conectada. 25
3.11 Exemplo numérico da execução do classificador *softmax*. 26
3.12 Estrutura da rede neural convolucional AlexNet. 27
3.13 Estrutura da rede neural convolucional VGGNet de 16 camadas. 29
3.14 Exemplo de regiões de interesse obtidas com o algoritmo de pesquisa seletiva. 33
3.15 Diagrama da proposta do algoritmo *R-CNN*. 34
3.16 Exemplo de uma regressão linear feita sobre uma região de interesse. 35

4.1 Etapa de detecção de regiões de interesse usando as imagens térmicas. 38

4.2 Seleção de superpixels com a temperatura média e incremento fixo. 40
4.3 Seleção dos superpixels com incremento fixo em imagens com temperaturas de regiões de fundo próxima a do animal. 40
4.4 Seleção de superpixels baseado no desvio padrão e média. 41
4.5 Exemplo de uma região de interesse detectada. 42
4.6 Etapa de identificação das espécies animais usando rede neural convolucional. 43
4.7 Projeção de uma região de interesse no mapa de características. . 44
4.8 Redimensionamento das características extraídas do mapa de características. 45

5.1 Exemplo do conjunto de imagens de treinamento. 48
5.2 Exemplo do conjunto de imagens de teste. 50
5.3 Áreas da intersecção sobre união de duas regiões. 51
5.4 Exemplos de resultados da intersecção sobre união. 51
5.5 Utilização do IoU para a verificação da precisão de uma região predita com dois *ground-truth*. 52
5.6 Exemplo do gráfico gerado pela precisão média. 53
5.7 Exemplo da estrutura dos diagramas de caixas e bigode. 54
5.8 Resultados da precisão média na arquitetura VGGNet de 16 camadas. 55
5.9 Resultados da precisão média na arquitetura VGGNet de 19 camadas. 56
5.10 Diagrama de caixas para as quatro abordagens avaliadas na arquitetura VGGNet de 16 camadas. 60
5.11 Diagrama de caixas para as quatro abordagens avaliadas na arquitetura VGGNet de 19 camadas. 61
5.12 Diagrama de caixas dos resultados da precisão média para as oito classes analisadas. 62
5.13 Diagrama de caixas dos resultados da medida-F para as oito classes analisadas. 63
5.14 Matriz de confusão para as regiões propostas na arquitetura VGGNet de 16 camadas. 64
5.15 Matriz de confusão para as regiões propostas na arquitetura VGGNet de 19 camadas. 64
5.16 Comparação visual da detecção de regiões de interesse propostas pelas abordagens. 66

Lista de Tabelas

5.1 Resultados das abordagens na arquitetura VGGNet com 16 camadas. 57
5.2 Resultados das abordagens na arquitetura VGGNet com 19 camadas. 57
5.3 Resultados do pós-teste de Tukey na arquitetura VGGNet de 16 camadas. 58
5.4 Resultados do pós-teste de Tukey na arquitetura VGGNet de 19 camadas. 59

CAPÍTULO

1

Introdução

1.1 Contexto e Motivação

O Pantanal é uma região natural localizada principalmente dentro dos estados brasileiros de Mato Grosso e Mato Grosso do Sul, se estendendo até a Bolívia e Paraguai. Esta região é a maior área úmida tropical do mundo, com área de aproximadamente 138.183 km^2. O nome Pantanal vem da palavra "Pântano", que significa zonas úmidas, porque 80% das planícies do Pantanal estão submersas durante as estações chuvosas [da Silva and de Moura Abdon, 1998].

O Pantanal é um dos ecossistemas mais ricos do mundo, abrigando um grande número de animais que vivem em equilíbrio ecológico. Junk et al. [2006] catalogaram 263 espécies de peixes, 96 espécies de répteis, 40 espécies de anfíbios, 390 espécies de aves e 130 espécies de mamíferos. O número de espécies diverge em vários trabalhos, por exemplo, as aves geralmente são dadas como algo entre 600 e 700 espécies [Swarts, 2000], embora apenas 390 foram confirmadas [Junk et al., 2006].

Como resultado do uso insustentável da terra, o Pantanal sofre muita perda de biodiversidade e dos seus habitats naturais [Alho and Sabino, 2011]. Nessa região existem muitas espécies raras e ameaçadas de extinção, como o tatu-canastra (*Priodontes maximus*), cervo-pantanal (*Blastocerus dichotomus*), tamanduá-bandeira (*Myrmecophaga tridactyla*), cachorro-do-mato (*Speothos venaticus*), entre outras. O grande desafio dos protetores desse bioma é conciliar a crescente demanda de desenvolvimento social e econômico (por exemplo, pecuária, gado, agricultura e turismo) com a conservação da biodiversidade

[Alho and Sabino, 2011].

Desta forma, conhecer a biodiversidade é essencial para os esforços dos investigadores que trabalham para proteger o Pantanal e as espécies ameaçadas de extinção. A identificação de espécies de animais no Pantanal leva tempo devido ao ambiente de zonas úmidas e aos hábitos alimentares dos animais, o que torna difícil estimar de forma eficaz as espécies e suas densidades por km^2 [Yu et al., 2013, Alho and Sabino, 2011]. Para acompanhar e monitorar a população, os pesquisadores passam dias no Pantanal observando as espécies com a ajuda de tecnologias, como as armadilhas fotográficas e imagens aéreas. O processo de identificação atual é extremamente demorado, requer treinamento especial e depende de uma inspeção manual das imagens [Fegraus et al., 2011, Yu et al., 2013].

Alguns métodos automáticos de reconhecimento de espécies foram propostos, como o de Chen et al. [2014] que usaram um algoritmo de segmentação automática de imagens obtidas por armadilhas fotográficas e fizeram o reconhecimento de 20 espécies de animais com redes neurais convolucionais, mas obtiveram apenas $38,31\%$ de precisão. Por outro lado, Yu et al. [2013] fez o recorte e seleção da imagens manualmente para obter o corpo inteiro do animal, isto permitiu que eles conseguissem 82% de precisão na classificação de 18 espécies de animais. Assim, temos que o problema de reconhecimento automático das espécies de animais ainda permanece não resolvido devido as condições de imagem muito desafiadoras [Gomez et al., 2016].

Para superar esses desafios, foi desenvolvido nesse projeto um método para o reconhecimento automático das espécies de animais do Pantanal baseado em redes neurais convolucionais e imagens térmicas, buscando deixar o processo de identificação mais rápido, diminuindo o custo com treinamento e pessoal. Diferente das abordagens já propostas na literatura, o nosso método localiza os animais na cena através de imagens térmicas, usando a diferença de calor entre o corpo do animal e o seu ambiente. A Figura 1.1 mostra um exemplo do uso de imagem térmica no monitoramento de animais silvestres, evidenciando a diferença de calor entre o corpo do animal e o ambiente.

Em seguida, a nossa abordagem usa as regiões de interesse na imagem RGB para fazer a identificação das espécies animais com as redes neurais convolucionais. Com isso, ela é capaz de classificar múltiplos animais nas imagens, sendo mais robusta que os métodos da literatura que classificam a imagem inteira. Os resultados mostraram que o uso das imagens térmicas na proposição de regiões para a classificação tem um grande potencial, alcançando um desempenho superior as abordagens comparadas, e conseguindo delimitar melhor os animais nas imagens.

Por fim, a ferramenta proposta pode auxiliar os pesquisadores na estima-

(a) Visual (b) Termografia

Figura 1.1: Exemplos de imagens térmicas usadas no monitoramento de uma onça-parda. Na coluna (a) é mostrada a imagem RGB e, na coluna (b), é mostrada a respectiva imagem térmica.

ção da população e densidade por km^2 dos animais, permitindo uma avaliação mais eficaz do impacto gerado pelo desenvolvimento econômico sobre esses animais, para que as empresas possam traçar metas e medidas que auxiliem no desenvolvimento sustentável do Pantanal.

1.2 Objetivos

O objetivo deste trabalho é propor uma abordagem automática para o reconhecimento de espécies de animais do Pantanal, empregando as redes neurais convolucionais [LeCun et al., 1989, Krizhevsky et al., 2012] e imagens térmicas. As imagens térmicas auxiliam na obtenção, por meio da temperatura, de regiões da imagem contendo algum animal que são posteriormente classificadas em uma das espécies usando redes neurais convolucionais. Para o desenvolvimento dessa abordagem alguns objetivos específicos foram definidos:

- Construir um banco de imagens térmicas e RGB das principais espécies animais do Pantanal.
- Avaliar a proposta do uso de imagens térmicas para obter as regiões de entrada das redes neurais convolucionais.
- Avaliar a aplicação da abordagem proposta em duas arquiteturas de redes neurais convolucionais com diferentes números de camadas.
- Desenvolver uma ferramenta automática de reconhecimento dessas espécies para utilização dos pesquisadores desse bioma.

1.3 Organização do Texto

O texto está organizado da seguinte forma:

- **Capítulo 2**: é realizada uma revisão de literatura sobre os desafios no reconhecimento de espécies animais e algumas abordagens já desenvolvidas, em seguida são apresentadas algumas aplicações que usam redes neurais convolucionais e imagens térmicas.
- **Capítulo 3**: neste Capítulo são descritas as teorias e técnicas usadas neste trabalho, no caso, o algoritmo de segmentação baseado em superpixel (SLIC), o algoritmo de componentes conectados para agrupamento de pixels, e as redes neurais convolucionais.
- **Capítulo 4**: são apresentadas as abordagens propostas para a identificação da espécie-animal usando imagens térmicas e redes neurais convolucionais.
- **Capítulo 5**: neste Capítulo são discutidos a validação dos experimentos e a comparação de resultados com outra abordagem de identificação de animais.
- **Capítulo 6**: são apresentadas as considerações finais.

CAPÍTULO

2

Trabalhos Correlatos

Neste capítulo é apresentada a revisão de literatura, realizada até o ano de 2016, dos principais métodos aplicados no projeto. Inicialmente, na Seção 2.1, é apresentado o problema do reconhecimento de espécies animais no Pantanal e algumas aplicações desenvolvidas para esse tipo de problema na área de visão computacional. Na Seção 2.2 são apresentadas algumas aplicações de Redes Neurais Convolucionais, e por fim na Seção 2.3 são apresentados os usos de imagens térmicas nos últimos anos.

2.1 Reconhecimento de Espécies de Animais

O grande desafio do Pantanal é conciliar o desenvolvimento social e econômico com a conservação de sua biodiversidade [Alho and Sabino, 2011]. Atualmente pelo uso insustentável da terra através do desmatamento, expansão da agricultura, caça e pesca ilegal, essa região vem acumulando grandes perdas de biodiversidade e habitats naturais [Alho and Sabino, 2011]. Para fazer a proteção do Pantanal e dessas espécies é muito importante que os órgãos de proteção e pesquisadores tenham um controle dessa biodiversidade, sendo necessário fazer a identificação das espécies presentes nesse bioma e a estimativa de suas populações.

Para conseguir fazer a identificação das espécies animais os pesquisadores normalmente passam vários dias em campo observando e capturando imagens com ajuda de armadilhas fotográficas, imagens aéreas, etc. O processo de identificação das espécies ocorre pela inspeção manual das imagens capturadas, o que demanda muito tempo e requer treinamento especial [Fegraus et al., 2011, Yu et al., 2013]. Para superar esses desafios, é necessário um

método que permita fazer o reconhecimento automático das espécies animais. Entretanto, isso é um grande desafio na área de visão computacional, principalmente pela característica do ambiente em que o animal se encontra e pela sua movimentação constante.

Algumas abordagens têm sido propostas para superar esses desafios. Um método de captura muito usado são as armadilhas fotográficas que gravam as imagens dos animais que se movem através de seu campo de visão [Aguzzi et al., 2009, Yu et al., 2013, Kays et al., 2010]. As armadilhas fotográficas também são usadas em conjunto com métodos de visão computacional para a identificação das espécies animais [Aguzzi et al., 2009, Yu et al., 2013]. Aguzzi et al. [2009] usou as armadilhas fotográficas na captura de imagens de espécies marinhas nas margens e no fundo do oceano, e aplicou o descritor de Fourier juntamente com o método de k vizinhos mais próximos. Eles fizeram a identificação de três espécies de animais através dos seus contornos na imagem, e obtiveram uma precisão de 77.92% para a classificação de peixes, 42.96% para caracóis, e de 63.02% para caranguejos.

Yu et al. [2013] propôs uma identificação automatizada de espécies de animais para as imagens capturadas por armadilhas fotográficas usando SIFT (transformada de características invariante a escala) e pirâmide espacial. Eles testaram o método em um banco de imagens com 18 espécies de animais terrestres, alcançando uma precisão média de 82%. Arruda et al. [2015] usaram um método de segmentação usando superpixels e componentes conectados em imagens térmicas para obter regiões da imagem contendo animais silvestres. Eles analisaram cinco classes de mamíferos de grande porte, e alcançaram uma taxa de acerto de 85%.

Algumas aplicações mais relacionadas a este trabalho foram de Chen et al. [2014] e Gomez et al. [2016], que propuseram métodos de identificação de espécies de animais usando redes neurais convolucionais. Chen et al. [2014] propôs um método para a identificação de espécies animais utilizando uma rede neural convolucional de seis camadas, sendo três convolucionais e três de subamostragem. Seu conjunto de dados foi capturado com armadilhas fotográficas e incluiu 20 espécies de animais comuns na América do Norte. Eles alcançaram uma precisão de 38.31% no reconhecimento das espécies. Os autores atribuíram o baixo desempenho às características desafiadoras das imagens de armadilhas fotográficas, e enfatizaram a capacidade de aprendizagem das redes neurais convolucionais e a possibilidade de melhoria do desempenho com a captura de mais imagens.

Por outro lado, Gomez et al. [2016] testaram seis arquiteturas CNNs, duas AlexNet [Krizhevsky et al., 2012], uma VGGNet [Simonyan and Zisserman, 2014], duas GoogLeNet [Szegedy et al., 2015] e três ResNet [He et al., 2015], e

quatro tipos de base de imagens com 26 espécies animais da Tanzânia. A sua melhor abordagem usou uma rede neural convolucional ResNet de 101 camadas e um conjunto de imagens segmentadas manualmente, obtendo uma precisão de 88.90% e 98.10% para as avaliações top 1 e top 5 do desafio IMAGENet [Russakovsky et al., 2015].

Apesar da nossa abordagem também usar redes neurais convolucionais como Chen et al. [2014] e Gomez et al. [2016], ela se diferencia por realizar a classificação sobre regiões das imagens. Nesses dois trabalhos a classificação pela rede neural convolucional é realizada sobre a imagem inteira, não conseguindo classificar múltiplos animais na mesma imagem. Além disso, a nossa abordagem também se diferencia do trabalho de Gomez et al. [2016], por realizar todo o processo de classificação de forma automática, uma vez que eles segmentam as imagens manualmente.

2.2 Redes Neurais Convolucionais (Convolutional Neural Networks - CNN)

As redes neurais convolucionais consistem em camadas convolucionais, na qual é incorporado o algoritmo de *Backpropagation* que aprende os parâmetros de cada camada. Desde a sua criação, a CNN tem sido caracterizada por três propriedades básicas: as conexões locais, o compartilhamento de peso e a subamostragem local. As duas primeiras propriedades permitem que o modelo aprenda os padrões visuais locais importantes com menos parâmetros ajustáveis que o modelo totalmente conectado, e a terceira propriedade prepara a CNN para possuir invariância à translação.

Uma das primeiras aplicações de redes neurais convolucionais é a LeNet-5 proposta por LeCun et al. [1989] para o reconhecimento óptico de caracteres. Comparado as redes neurais convolucionais atuais, ela possuía uma arquitetura menor devido aos recursos computacionais limitados da época e aos desafios do treinamento de algoritmos para redes maiores, sendo composta por apenas 7 camadas e imagem de entrada de tamanho 32×32 pixels. Embora as redes neurais convolucionais tivessem muito potencial, só recentemente elas se tornaram predominantes. Isso aconteceu pelo aumento do poder computacional atual, da quantidade de dados para treinamento disponíveis na Internet e da necessidade de desenvolvimento de métodos mais eficazes para a formação de tais modelos [Krizhevsky et al., 2012, Simonyan and Zisserman, 2014, He et al., 2015, Szegedy et al., 2015].

Um exemplo recente e notável do uso de redes convolucionais profundas para classificação de imagens é o desafio Imagenet [Russakovsky et al., 2015] em que uma CNN [Krizhevsky et al., 2012] obteve um erro consideravelmente

menor comparado com o erro das abordagens tradicionais de visão computacional (usando SIFT e Máquinas de Vetores de Suporte). As redes neurais convolucionais também obtiveram recentemente sucesso para diferentes aplicações, incluindo estimação de pose humana [Toshev and Szegedy, 2014], análise de rosto [Luo et al., 2012], detecção de pontos-chave facial [Sun et al., 2013], reconhecimento de voz [Graves et al., 2013] e classificação de ação [Karpathy et al., 2014].

2.3 Imagens Térmicas

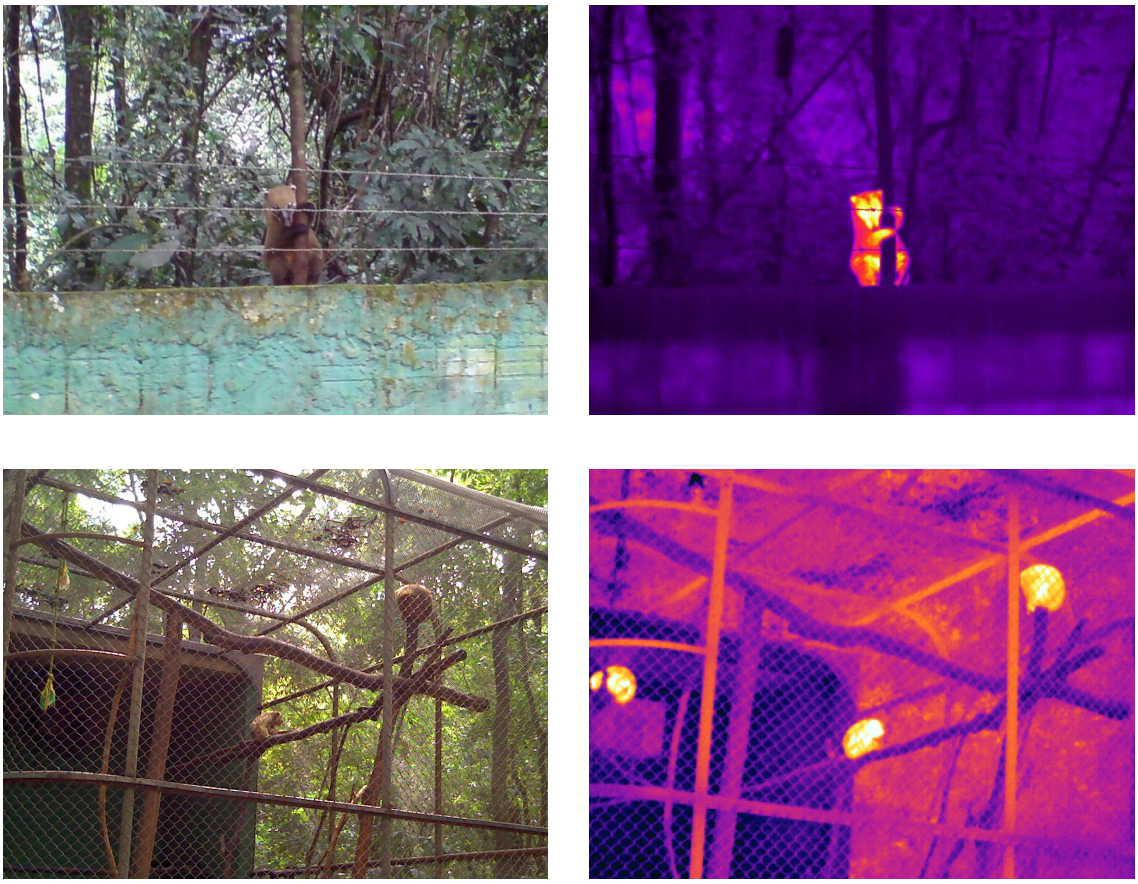
As imagens térmicas registram a radiação infravermelha emitida pelos objetos com uma temperatura acima do zero absoluto. Conforme a intensidade emitida por um objeto aumenta, maior o contraste dele na imagem térmica. Assim, câmeras térmicas podem ser usadas durante o dia e a noite. O uso de imagens térmicas possui algumas vantagens sobre problemas clássicos da visão computacional (condições de iluminação, características de textura do ambiente e do objeto). As imagens capturadas por câmeras visuais dependem da fonte de energia visível no espectro eletromagnético, tornando-se bastante sensíveis às condições de iluminação e sombras. Por outro lado, imagens térmicas tem seus próprios desafios, incluindo as condições do clima, a distância entre o objeto de interesse e a câmera e, o efeito *halo*[1] que aparece ao redor de objetos muito quentes ou muito frios.

Na Figura 2.1 são mostradas duas imagens para a mesma cena no monitoramento de animais, uma delas capturada com o sensor CCD visível e a outra com o sensor térmico. As propriedades térmicas do animal em relação ao ambiente (Fundo) são bastante diferentes. Nota-se que na segunda linha, dificilmente o animal na sombra poderia ser identificado usando abordagens visuais tradicionais.

As câmeras térmicas foram desenvolvidas inicialmente para o uso da indústria militar em tarefas que exigiam visão noturna [Kastek et al., 2010, Siegel, 2002]. Entretanto, nas últimas décadas as imagens térmicas têm sido usadas em uma vasta gama de aplicações, tais como inspeção da qualidade de alimentos [R. Vadivambal, 2011], detecção por calor em processos industriais [Ng et al., 2007] e detecção de vazamento de gás ou focos de incêndios [Hwang et al., 2010].

Outras aplicações de imagens térmicas que têm crescido nos últimos anos são para a detecção, reconhecimento e monitoramento de animais. Tatter-

[1] É uma distorção que ocorre principalmente em câmeras térmicas equipadas com sensores ferroelétricos, fazendo com que as regiões que rodeiam um objeto brilhante na imagem se tornem mais escuras e a região em torno de objetos escuros se torne mais claras [Goodall et al., 2016].



(a) Visual (b) Termografia

Figura 2.1: Exemplos de imagens térmicas usadas no monitoramento de animais. Nas cenas da primeira linha é mostrado um quati, e da segunda linha são mostrados três macacos. Na coluna (a) são mostradas as imagens RGB e, na coluna (b), são mostradas as mesmas imagens térmicas.

sall and Cadena [2010] usaram as imagens térmicas para o monitoramento em estudos com animais, enquanto H. B. Graves [1972] comparou a eficácia da detecção de veados de cauda branca em imagens térmicas para diferentes níveis de sensibilidade em levantamentos aéreos. Collier et al. [2007] comparou a eficácia de dois métodos para a detecção de populações de veados de cauda branca: usando imagens térmicas (que detectou $92,3\%$ dos veados) e contagem por amostragem (que detectou somente de $54,4\%$ dos animais). Por fim, Garner et al. [1995] também avaliaram a eficiência da termografia no levantamento das populações de cervos, alces e perus selvagens.

CAPÍTULO

3

Referencial Teórico

Neste capítulo, são detalhados os conceitos aplicados nesse trabalho para desenvolver a abordagem proposta. Inicialmente, na Seção 3.1, é apresentada a abordagem de segmentação de imagens usando superpixels juntamente com a descrição da técnica SLIC. Na Seção 3.2 é descrito o funcionamento do algoritmo de componentes conectados para o agrupamento de pixels. Na Seção 3.3 são apresentadas as partes que compõe as redes neurais convolucionais e o seu funcionamento. Na Seção 3.4 são descritas algumas arquiteturas de redes neurais convolucionais importantes que foram propostas e, na Seção 3.5, são apresentadas as abordagens que usam redes neurais convolucionais baseadas em regiões de interesse.

3.1 Segmentação de Imagens usando Superpixels

As técnicas baseadas em superpixels se dividem em dois tipos: os algoritmos que recaem sobre grafos [Shi and Malik, 1997, Veksler et al., 2010, Moore et al., 2008] e os algoritmos baseados em gradiente [Vincent and Soille, 1991, Vedaldi and Soatto, 2008, Levinshtein et al., 2009, Achanta et al., 2012].

O SLIC (*Simple Linear Iterative Clustering*) [Achanta et al., 2012] é uma das técnicas baseadas em gradiente que divide a imagem em regiões com um número aproximado de pixels, também chamados de superpixels. O SLIC é inspirado no algoritmo *k-means* [Hartigan and Wong, 1979], usando a distância Euclidiana para definir as regiões.

A técnica possui dois parâmetros para determinar o número de pixels de cada região. O parâmetro k determina a quantidade desejada de superpixels na imagem, enquanto m controla o nível de compactação dos superpixels

definido pela relação entre a similaridade dos atributos de cor e a proximidade espacial. Basicamente, o algoritmo usa cinco dimensões representadas no vetor $\nu = [L, a, b, x, y]$, tal que $[L, a, b]$ representam os valores no espaço de cor CIELAB e $[x, y]$ são as coordenadas espaciais dos pixels. O Algoritmo 1 representa o funcionamento do algoritmo de segmentação de imagens por superpixels SLIC.

Algoritmo 1: Segmentação de Imagens SLIC

```
1  Inicializa os centros da imagem C_k = [l_k, a_k, b_k, x_k, y_k]^T por amostragem de
   uma grade regular com espaçamento S.
2  Mova os centros para a posição de gradiente mais baixa em uma
   vizinhança 3 × 3.
3  repita
4      Defina um label l(i) = −1 para cada pixel i.
5      Defina uma distância d(i) = ∞ para cada pixel i.
6      para cada centro C_k faça
7          para cada pixel i em uma região 2S × 2S ao redor de C_k faça
8              Computar a distância d entre C_k e i.
9              se d < d(i) então
10                 definir d(i) = d.
11                 definir l(i) = k.
12             fim
13         fim
14     fim
15     Computar os novos centros das regiões.
16     Computar o erro residual E.
17 até E ≤ limiar ;
```

O algoritmo SLIC inicia com k centros na imagem, dado por $C_k = [l_k, a_k, b_k, x_k, y_k]$, amostrados em uma grade regular igualmente espaçados em intervalos $S = \sqrt{N/k}$, tal que N é a quantidade de pixels na imagem (linha 1 do Algoritmo 1). Os centros são movidos para a posição com menor valor de gradiente resultante da vizinhança de 3×3 pixels, evitando assim pixels ruidosos ou borda (linha 2 do Algoritmo 1). Em seguida, o algoritmo calcula a distância Euclidiana de cada pixel i em relação aos centros C_k de uma região de busca de tamanho $2S \times 2S$ [1] (linha 6-14 do Algoritmo 1).

Para computar a distância d entre um pixel i e um centro C_k representado na linha 8 do Algoritmo 1, precisamos considerar as informações de cor e posição espacial de i e C_k. Porém, o uso da distância Euclidiana para agrupar

[1] Achanta et al. [2012] definiu a área de busca $2S \times 2S$ para aumentar a velocidade de processamento.

os pixels no espaço de atributos com cinco dimensões ($labxy$) se torna inconsistente. Dessa forma, as distâncias são calculadas separadamente para a proximidade de cor (Equação 3.1) e posição espacial (Equação 3.2).

$$d_c(i,k) = \sqrt{(L_k - L_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \tag{3.1}$$

$$d_s(i,k) = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \tag{3.2}$$

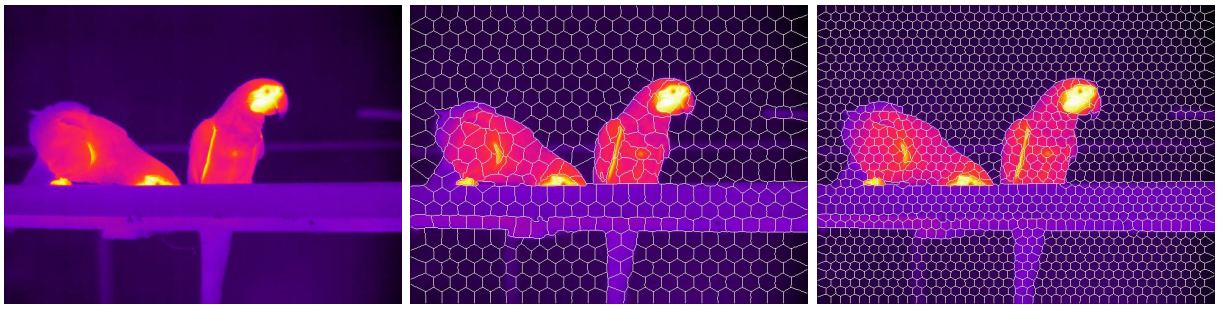
Em seguida, é necessário normalizar a proximidade de cor em relação à distância espacial, somando-se as distâncias Euclidianas para o espaço de cor d_c e para a posição espacial d_s dos pixels. A distância total d é então calculada pela Equação 3.3, dada por:

$$d = d_c + m\frac{d_s}{S} \tag{3.3}$$

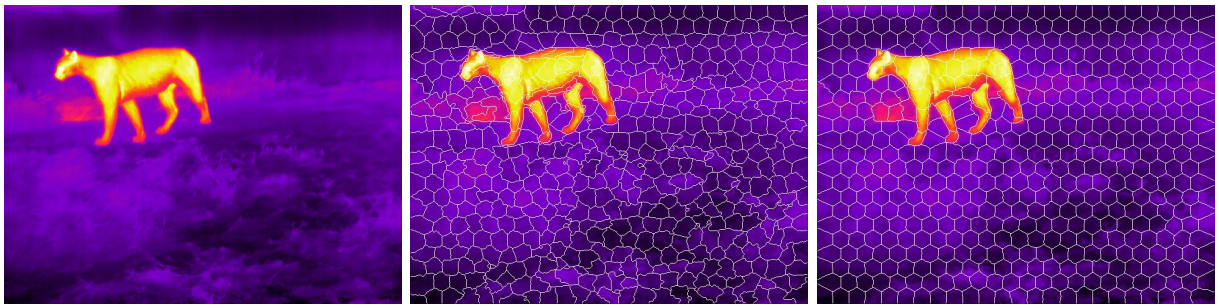
tal que m permite ponderar a relação da proximidade de cor e posição espacial dos pixels. Quanto maior é o valor de m, mais a proximidade espacial é privilegiada e resulta em superpixels mais compactos, enquanto que valores de m mais baixos, como $m = 1$, privilegiam a similaridade da cor dos pixels, criando superpixels não regulares em forma e tamanho.

Cada pixel i é então associado ao centro do grupo mais próximo (linha 6-14 do Algoritmo 1). Em seguida, o novo centro é atualizado por meio da média dos atributos de todos os pixels atribuídos à ele. O critério de convergência do algoritmo é determinado pelo erro residual entre a posição do centro da última iteração e a posição do centro atual (linha 15-17 do Algoritmo 1). Em alguns casos são usados apenas um número específico de iterações ao invés do erro.

A Figura 3.1 mostra exemplos do uso da técnica SLIC no monitoramento de animais em imagens térmicas. Na primeira linha são usados diferentes valores de k para ilustrar a geração das partições de superpixels nas imagens. Podemos observar que ao aumentar o valor de k obtemos uma maior divisão da imagem e superpixels menores. Na segunda linha da Figura 3.1 temos a variação do parâmetro m e a sua influência na geração dos superpixels. Podemos observar que ao aumentar o valor de m obtemos superpixels mais compactos, pois ele privilegia a distância espacial em relação a cor.



(a) Imagem de entrada (b) SLIC com $k = 500$ (c) SLIC com $k = 1500$



(d) Imagem de entrada (e) SLIC com $m = 1$ (f) SLIC com $m = 5$

Figura 3.1: Dois exemplos de monitoramento de animais usando a técnica SLIC com variação nos parâmetros k e m. Na primeira linha é mostrada a variação do parâmetro k em uma imagem de arara-canindé com k igual a 500 e 1500, respectivamente. Na segunda linha temos a variação do parâmetro m em uma imagem de onça-parda com m igual a 1 e 5, respectivamente.

3.2 Agrupamento de Pixels usando Componentes Conectados

O algoritmo de componentes conectados [Haralick and Shapiro, 1992, Rosenfeld, 1970, 1976] implementa o agrupamento de pixels de uma imagem representada por uma matriz de entrada. Basicamente, a matriz é percorrida linha por linha buscando os pixels que identificam algum objeto de interesse na imagem (com valor 1 no caso de imagens binárias). Em seguida, é atribuída uma etiqueta temporária, em uma matriz de labels, para cada pixel identificado. Para isso, o algoritmo analisa as etiquetas atribuídas a pixels da vizinhança de 4 ou 8 pixels, representados na Figura 3.2, salvando os conflitos de etiquetas em uma matriz de equivalência que é usada para a rotulação final dos componentes da imagem.

A Figura 3.3 ilustra o funcionamento do algoritmo de componentes conectados com vizinhança de 8 pixels em uma matriz de exemplo. O algoritmo percorre a matriz de entrada (Figura 3.3 (a)) de cima para baixo e da esquerda para a direita, atribuindo etiquetas temporárias para cada pixel, representado

Figura 3.2: Representação da vizinhança de 4 ou 8 pixels, representado pela cor verde, ao se analisar o pixel de cor vermelha. Na figura (a) temos a representação da vizinhança de 4 pixels, e na figura (b) temos a representação da vizinhança de 8 pixels.

pela cor vermelha, considerando as etiquetas dos vizinhos que já foram percorridos, representados pela cor verde. Quando encontramos o primeiro pixel que identificam um objeto e a matriz de labels está vazia, o pixel recebe o primeiro valor de etiqueta (1), mostrado na Figura 3.3 (c). Se existe apenas um pixel etiquetado na vizinhança, o algoritmo atribuí a mesma etiqueta ao pixel analisado como é mostrado na Figura 3.3 (e).

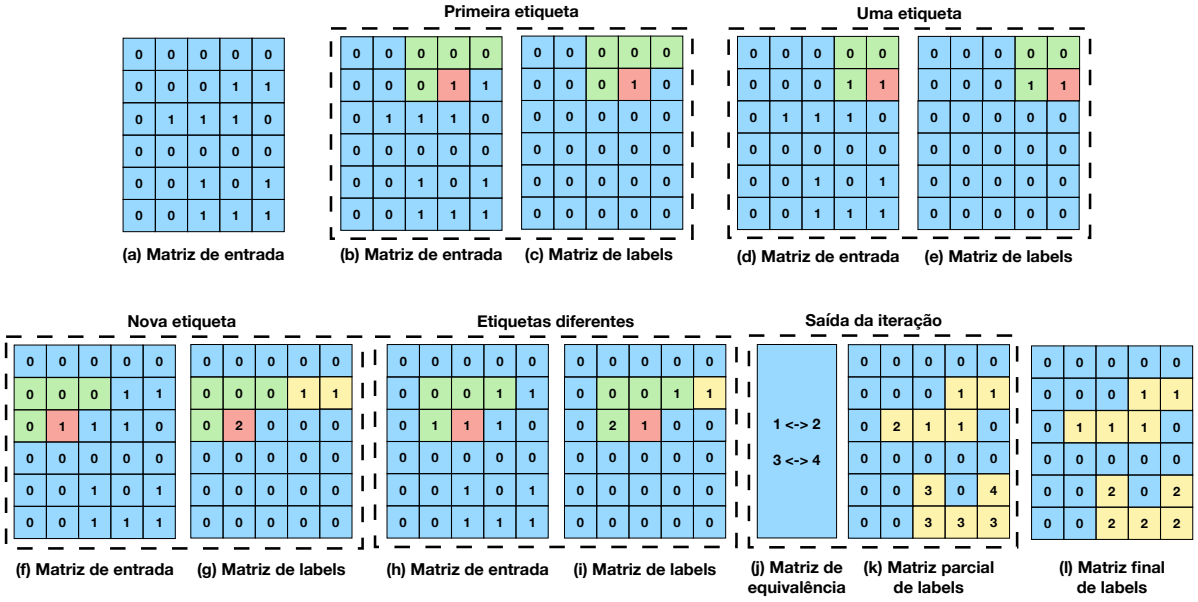


Figura 3.3: Exemplo do funcionamento do algoritmo de componentes conectados com vizinhança de 8 pixels. A Figura (a) representa a matriz de entrada que será processada pelo algoritmo. Nas figuras (b) e (c) temos a execução quando não existe etiquetas atribuídas na matriz de labels. Nas figuras (d) e (e) temos a execução quando existe uma etiqueta na vizinhança já analisada. Nas figuras (f) e (g) temos o processo de criação de uma nova etiqueta quando não temos etiquetas na vizinhança analisada mas temos na matriz de labels. Nas figuras (h) e (i) temos o processo quando a vizinhança contêm etiquetas diferentes. Nas figuras (j) e (k) temos a matriz de equivalência e a matriz de labels parcial obtida ao fim da primeira iteração sobre a matriz de entrada. Na figura (l) temos a matriz de labels final atualizada a partir da matriz de equivalência.

Quando o algoritmo não encontra pixels etiquetados na vizinhança, mas

existem etiquetas na matriz de labels, ele gera uma nova etiqueta com o próximo valor (Figura 3.3 (g)). Se o algoritmo encontrar etiquetas com valores diferentes na vizinhança (Figura 3.3 (i)), ele seleciona uma e registra na matriz de equivalência (Figura 3.3 (j)) que ambas etiquetas compõem a mesma componente. Por fim, ao terminar de percorrer a matriz de entrada o algoritmo percorre a matriz parcial de labels (Figura 3.3 (k)) atribuindo as etiquetas finais usando a matriz de equivalência. Assim, obtemos a matriz final de labels representado pela Figura 3.3 (l).

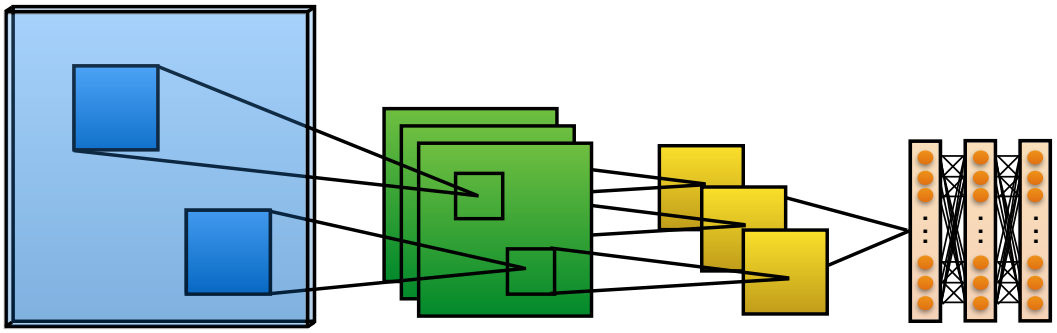
3.3 Redes Neurais Convolucionais

As redes neurais convolucionais são uma sequência de camadas, onde cada camada transforma um volume de ativações para outro através de uma função. Elas são uma variação das redes Perceptrons de Múltiplas Camadas (*Multilayer Perceptron - MLP*). Basicamente, a rede neural convolucional permitiu uma certa invariância a transformações de entradas, ao contrário das redes MLP que perdiam informações espaciais das entradas por possuírem todas as camadas totalmente conectadas.

Para superar esse desafio as redes neurais convolucionais se basearam em três ideias básicas: (i) a utilização de campos receptivos ao invés da conexão total dos neurônios, (ii) o compartilhamento de parâmetros que auxilia na utilização dos campos receptivos e na diminuição do número de parâmetros usados pela CNN e, (iii) o processo de subamostragem, que permite uma invariância espacial na CNN e faz a redução progressiva do tamanho da imagem para a classificação.

As redes neurais convolucionais são compostas por diferentes tipos de camadas, mas os três tipos principais são: a Camada Convolucional, a Camada de Subamostragem, e a Camada Totalmente Conectada. A Figura 3.4 mostra a estrutura de uma rede neural convolucional e suas camadas.

Nas próximas seções vamos apresentar a estrutura e o funcionamento dos principais tipos de camadas que compõem a rede neural convolucional. Na Seção 3.3.1 é descrito o funcionamento da camada convolucional na CNN, na Seção 3.3.2 é apresentada a camada de normalização, na Seção 3.3.3 é mostrado o funcionamento e as características de uma camada de subamostragem, na Seção 3.3.4 são apresentadas as características da camada totalmente conectada, e por fim, na Seção 3.3.5 é descrito o funcionamento do classificador *softmax*.



(a) Imagem de entrada (b) Camada convolucional (c) Camada de subamostragem (d) Camada totalmente conectada

Figura 3.4: Exemplo de uma rede neural convolucional com os três tipos principais de camadas: Camada convolucional, Camada de subamostragem e Camada totalmente conectada.

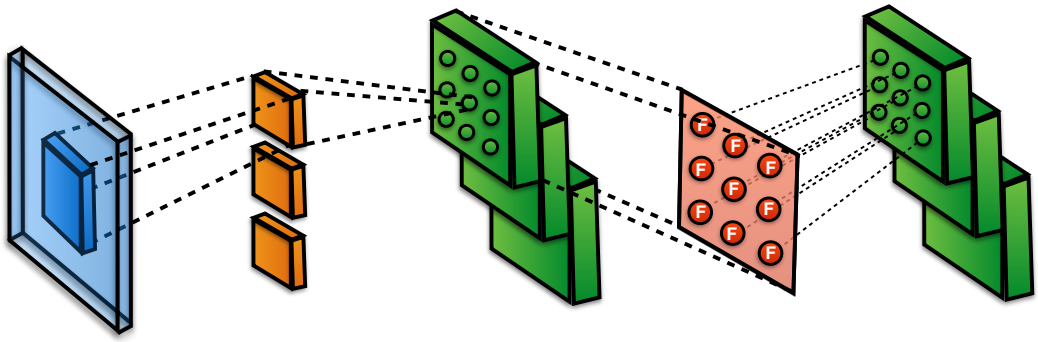
3.3.1 Camada Convolucional

A camada convolucional é um dos principais blocos da rede neural convolucional, pois ela faz grande parte do trabalho computacional da CNN. Ela conecta cada neurônio a uma região local da imagem que são chamadas de campos receptivos. Assim, cada cálculo executado por essa camada é basicamente um produto entre um conjunto de pesos, também chamados de filtros, e uma pequena região da imagem de entrada.

A Figura 3.5 apresenta a estrutura da camada convolucional. Basicamente, recebemos uma imagem de entrada (Figura 3.5 (a)) que é convoluída com um conjunto de filtros (Figura 3.5 (b)) por meio da varredura de um campo receptivo, representado pela área destacada na imagem de entrada. Em seguida, obtemos um volume de saída (Figura 3.5 (c)) onde cada posição é um neurônio que corresponde a multiplicação do campo receptivo por um filtro. Após esse processo temos a aplicação da função de ativação (Figura 3.5 (d)) que é responsável pela transformação não-linear dos dados. Por fim, obtemos o volume final (Figura 3.5 (e)) que será usado como entrada na próxima camada da CNN. As próximas seções explicam o funcionamento de cada etapa desse processo.

Processo de Convolução

O processo de convolução é o principal mecanismo da camada convolucional. A convolução é utilizada no processamento de imagens para a detecção de bordas, suavização da imagem, ou ainda para a extração de atributos dela, entre outras aplicações. Basicamente, o processo de convolução executa uma operação matemática entre um volume de entrada V, que pode ser a imagem de entrada da CNN ou a saída de outras camadas, e um conjunto de filtros F



(a) Imagem de entrada (b) Conjunto de filtros (c) Volume de saída (d) Aplicação da função de ativação (e) Entrada da próxima camada

Figura 3.5: Representação dos principais componentes da camada convolucional. Na figura (a) temos a imagem de entrada com o campo receptivo em destaque, na figura (b) temos o conjunto de filtros usados no processo de convolução. A figura (c) apresenta o volume de saída desse processo, e na figura (d) e (e) temos a aplicação da função de ativação sobre cada neurônio.

que percorre o volume de entrada. O processo de convolução é dado por:

$$V_s(x,y) = (V * f)(x,y) = b + \sum_{k=0}^{P_f-1} \sum_{i=0}^{L_f-1} \sum_{j=0}^{A_f-1} V(x \times D + i, y \times D + j) f(i,j,k) \qquad (3.4)$$

onde V possui tamanho $[L_V \times A_V \times P_V]$, que representam a largura, altura e profundidade do volume, e f é um filtro do conjunto de filtros F, e possui tamanho $[L_f \times A_f \times P_f]$, que representam a largura, altura e profundidade do filtro. Além disso, b representa o bias associado ao filtro f, e D representa o deslocamento que garante a convolução do filtro com todo o volume de entrada. Portanto, o processo de convolução pode ser interpretado como um somatório da multiplicação de cada elemento do volume de entrada e seus vizinhos locais, chamado de campo receptivo C_r, com todos os filtros do conjunto de filtros F.

A Figura 3.6 mostra um exemplo numérico do processo de convolução. Inicialmente recebemos um volume de entrada V (Figura 3.6 (a)) com dimensão $[4 \times 4 \times 3]$, um conjunto de filtros F (Figura 3.6 (b)) que possui três filtros de dimensão $[2 \times 2 \times 3]$, representados pelas cores verde, vermelho e amarelo, e um conjunto de bias B (Figura 3.6 (c)), que possui os bias de todos os filtros de F. Assim, realizamos o processo de convolução descrito na Equação 3.4, onde convoluímos cada filtro com o volume de entrada considerando um deslocamento $D = 1$. Por fim, obtemos o volume de saída V_s com dimensão $[3 \times 3 \times 3]$ (Figura 3.6 (d)), onde cada camada representa a convolução realizada por um filtro diferente.

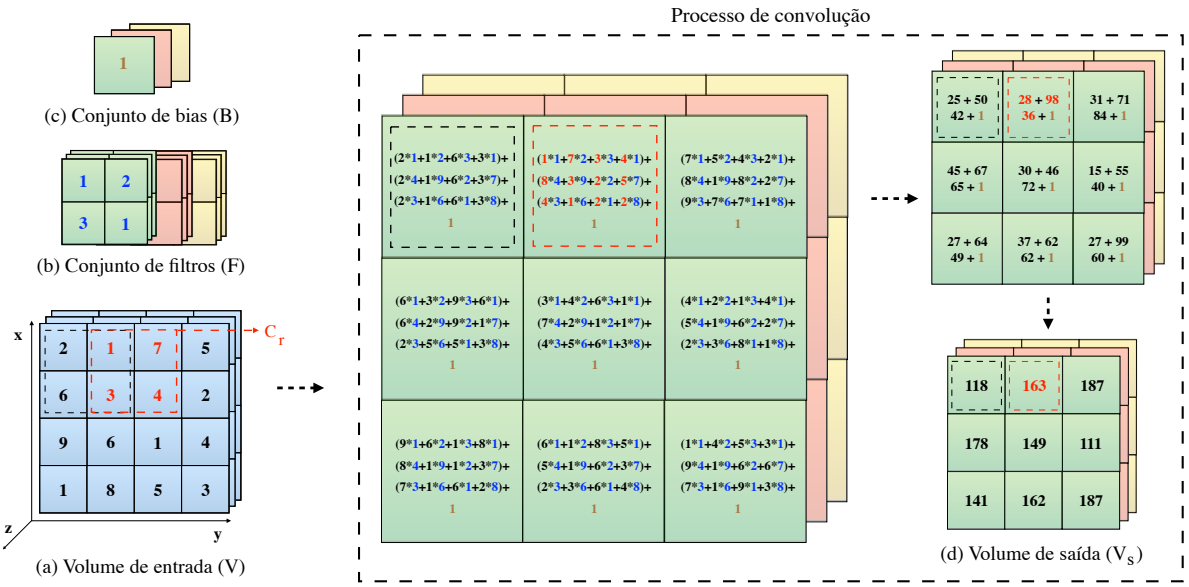


Figura 3.6: Exemplo numérico do processo de convolução realizado sobre um volume de entrada $[4 \times 4 \times 3]$, representado pela figura (a), usando um conjunto de filtros F com três filtros de tamanho $[2 \times 2 \times 3]$, representado pela figura (b), e um conjunto de bias B, representado pela figura (c). Nesse exemplo, o campo receptivo C_r é representado pelo quadrado tracejado em vermelho, e o deslocamento usado foi $D = 1$. A figura (d) mostra o volume de saída obtido pelo processo de convolução.

Controle da Dimensão da Camada Convolucional

Existem três hiperparâmetros importantes que são usados em conjunto com os filtros, e que ajudam a controlar o tamanho do volume de saída da camada convolucional, são eles: o número de filtros, o deslocamento, e o preenchimento com zeros.

O número de filtros N_F determina a quantidade de filtros presentes no conjunto F, onde cada filtro aprende uma característica diferente do volume de entrada. O deslocamento é usado para mover o filtro sobre o volume de entrada na convolução, se o deslocamento é 1 o filtro é movido para a próxima posição do volume, se é 2, o filtro se desloca duas posições. O valor do deslocamento é usado para mover o filtro em todas as direções no volume. Assim, quanto maior o valor do deslocamento, menor será o tamanho do volume produzido.

O preenchimento com zeros em torno do volume de entrada permite que o tamanho espacial do volume de saída seja controlado, podendo permitir por exemplo que a largura e altura do volume de saída sejam os mesmos que da entrada. O cálculo do tamanho espacial para o volume de saída V_s pode ser escrito como $L_{V_s} = (L_V - L_F + 2P_z)/D + 1$ onde L_{V_s} é a largura do volume de saída, L_V é a largura do volume de entrada, L_F é a largura dos filtros, P_z é a largura da borda usada para o preenchimento com zeros em torno do volume, e D

é o tamanho do deslocamento usado. No caso da Figura 3.6, temos $L_V = 4$, $L_F = 2$, $D = 1$ e $P_z = 0$. Assim, calculando a dimensão de V_s, temos que $L_{V_s} = (L_V - L_F + 2P_z)/D + 1 = (4 - 2 + 2 \times 0)/1 + 1 = (2 + 0)/1 + 1 = 2 + 1 = 3$, o que podemos confirmar na Figura 3.6 (e) pois $L_{V_s} = 3$.

Função de Ativação

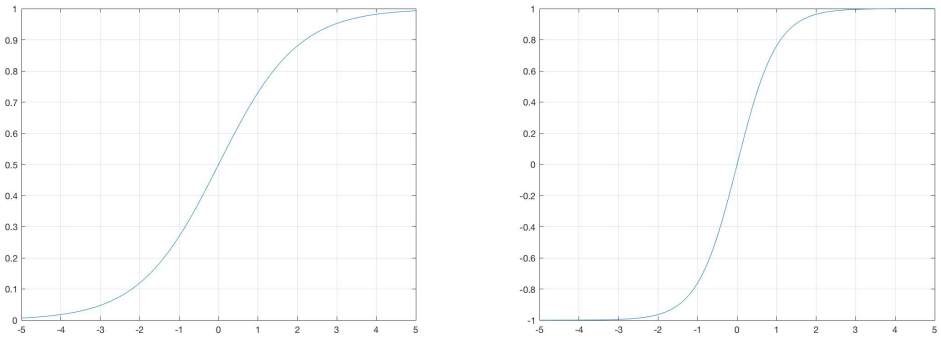
Normalmente após o processo de convolução é usada uma função de ativação que introduzem um componente não linear nas redes neurais convolucionais. Basicamente, ela recebe os valores de saída da camada convolucional e executa uma operação matemática fixa sobre eles. Assim, ela não permite que a saída da rede neural convolucional sejam geradas apenas por uma função linear sobre a entrada. Por exemplo, considerando uma rede neural convolucional com duas camadas convolucionais, temos que a saída das duas camadas podem ser representadas por:

$$\phi(\phi(V \times F_1 + B_1) \times F_2 + B_2) \tag{3.5}$$

onde V é a imagem de entrada da CNN, F_i e B_i são os filtros e bias aplicados em cada camada convolucional, e ϕ representa as funções de ativação usadas sobre a saída de cada camada convolucional. Podemos observar que sem as funções de ativação ϕ, a saída das duas camadas convolucionais seriam uma transformação linear k sobre a imagem de entrada V, onde $k = ((F_1 + B_1) \times F_2 + B_2)$. Além disso, poderíamos substituir as duas camadas convolucionais, por uma única camada que executa uma operação $V \times F' + B'$, onde $F' = F_1 \times F_2$ e $B' = (F_2 \times B_1) + B_2$. Assim, a saída da rede seria considerada um simples modelo de regressão linear, pois aplicaria apenas transformações lineares sobre a imagem de entrada V. Portanto, as funções de ativação são importantes para que a CNN aprenda e execute tarefas mais complexas, que não seriam possíveis apenas pelo uso de transformações lineares. Alguns exemplos de funções de ativação propostos são: a função sigmoide, a função tangente hiperbólica (tanh) e a função ReLu.

As funções sigmoide e tanh basicamente normalizam o valor de entrada para o intervalo de $[0, 1]$ e $[-1, 1]$, como é mostrado na Figura 3.7. Como podemos observar, uma limitação do uso dessas funções é quando as ativações dos neurônios saturam nos extremos 0 ou 1 para sigmoide e -1 ou 1 para tanh, temos que o aumento ou diminuição dessa entrada (eixo X) terá impacto mínimo na saída (eixo Y). Com isso, os gradientes nessas regiões serão muito pequenos e durante o Backpropagation não será possível fazer alterações significativas nos pesos desses neurônios, pois quase nenhum sinal fluirá através deles.

A função ReLu tornou-se muito popular nos últimos anos e calcula a fun-



(a) Sigmoide (b) Tangente hiperbólica

Figura 3.7: Exemplo visual da saída da função sigmoide, representado na figura (a), e tangente hiperbólica, representado pela figura (b).

ção $f(x) = max(0, x)$ onde basicamente aplica-se um limiar em zero. Apesar de ser uma estratégia simples, ela tem uma maior aceleração na convergência da descida do gradiente em comparação com as funções sigmoide e tanh[2], e um custo computacional menor pois ela simplesmente descarta as entradas menores que 0, enquanto que as funções sigmoide e tanh transformam todas as entradas para os intervalos de $[0, 1]$ e $[-1, 1]$, respectivamente.

3.3.2 Camada de Normalização

Dado que as saídas de cada camada de uma rede neural convolucional são as entradas da próxima, temos que as entradas de cada camada são afetadas pelos parâmetros de todas as anteriores. Assim, pequenas alterações nos parâmetros de uma camada são amplificadas conforme a CNN fica mais profunda. Isso se torna um problema no treinamento pois as camadas precisam se adaptar continuamente à nova distribuição de entrada, atrasando o processo de aprendizado da CNN. Essa dependência é chamada de mudança covariável [Shimodaira, 2000], e as camadas de normalização foram propostas para resolver esse tipo de problema.

A camada de normalização é aplicada nas camadas internas da CNN, como entre a camada convolucional e de subamostragem, e ela normaliza a saída de uma camada que será a entrada da próxima. Ela é indicada para redes neurais convolucionais com ativações muito altas, pois permite a detecção de características de alta frequência enquanto diminuem as respostas uniformemente altas de uma vizinhança. Basicamente, essa camada procura incentivar a competição de grandes atividades e respostas dos neurônios em uma região

[2]Krizhevsky et al. [2012] mostraram que a função ReLu alcança a convergência seis vezes mais rápido que a tanh.

local.

Foram propostos vários tipos de camadas de normalização, como a Normalização de Resposta Local, Normalização de Variância Média e Normalização em Lotes. A Normalização de Resposta Local (*Local Response Normalization*) executa a "inibição lateral" normalizando as regiões de entrada locais [Krizhevsky et al., 2012], e ficou conhecida por ter sido usada nas arquiteturas AlexNet e VGGNet. Por outro lado, a Normalização de Variância Média (*Mean Variance Normalization*) realiza normalização de contraste ou de instância [Bolstad et al., 2003] e a Normalização em Lotes (*Batch Normalization*), que ficou muito conhecida por ser usada em redes neurais convolucionais como Inception [Szegedy et al., 2017] e ResNet [He et al., 2015], que executa a normalização em mini-lotes [Ioffe and Szegedy, 2015].

3.3.3 Camada de Subamostragem

A camada de subamostragem (*subsampling layer* ou *pooling layer*) realiza uma operação de amostragem descendente ao longo das dimensões espaciais da imagem. Ela é normalmente inserida após a camada convolucional. A sua principal função é gerar invariância espacial para facilitar a detecção dos padrões esperados em determinadas regiões da imagem mesmo que ocorra pequenas mudanças nas posições desses padrões. Além disso, ela reduz progressivamente o tamanho espacial da imagem, reduzindo a quantidade de parâmetros e o custo computacional da CNN.

A camada de subamostragem recebe um volume de entrada V com tamanho $[L_V \times A_V \times P_V]$, onde L_V é a largura, A_V é a altura e P_V é a profundidade do volume. Em seguida, é definido dois hiperparâmetros, a largura do campo receptivo L_{C_r}, que gera um campo receptivo C_r com dimensão $[L_{C_r} \times L_{C_r}]$, e o deslocamento D. Por fim, ela faz o redimensionamento espacialmente do volume de entrada através de uma função de agrupamento. Existe várias funções de agrupamento para realizar a amostragem descendente, as três principais são: a subamostragem média, a subamostragem da norma $L2$ e a subamostragem máxima.

A subamostragem média e a subamostragem da norma $L2$ já foram bastante utilizados ao longo dos anos [LeCun et al., 1989, Lecun et al., 1998], mas recentemente caíram em desuso quando comparados com a subamostragem máxima [Scherer et al., 2010, Boureau et al., 2010], pois ele permite aplicar a amostragem descendente de uma forma mais simples e intuitiva, mantendo apenas as maiores ativações conforme reduz o tamanho do volume.

A Figura 3.8 mostra o funcionamento da camada de subamostragem máxima em um exemplo numérico com $L_{C_r} = 2$ e $D = 2$. Basicamente, recebemos um volume de entrada V (Figura 3.8 (a)) com dimensão $[4 \times 4 \times 3]$, e selecio-

namos o valor máximo, representado pelos valores destacados em vermelho, de cada campo receptivo. O campo receptivo C_r, representado pela área tracejada, foi movido por um deslocamento $D = 2$, representado pelas quatro cores no volume, resultando em um volume de saída V_s (Figura 3.8 (b)) com dimensão $[2 \times 2 \times 3]$.

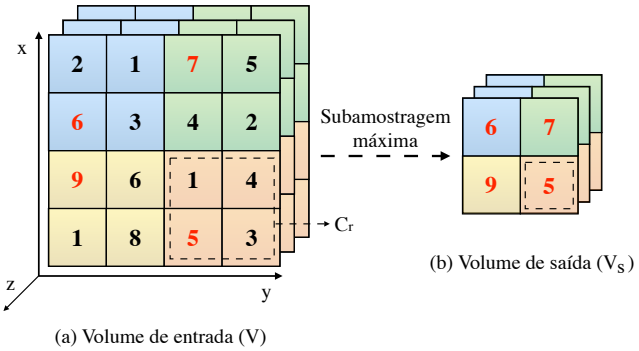


Figura 3.8: Exemplo numérico da operação de subamostragem máxima sobre um volume de entrada, com $L_{C_r} = 2$ e $D = 2$. Na figura (a) temos o volume de entrada V com dimensão $[4 \times 4 \times 3]$, e na figura (b) temos o volume de saída V_s com dimensão $[2 \times 2 \times 3]$.

Existem duas variações principais da subamostragem máxima. A mais usada emprega a largura do campo receptivo L_{C_r} e o deslocamento D igual à 2, e a segunda é chamada de subamostragem de sobreposição pois aplica $L_{C_r} = 3$ e $D = 2$, fazendo com que os campos receptivos se sobreponham. Não é recomendado usar tamanhos de L_{C_r} e D maiores, pois a execução irá descartar várias ativações devido a alta sobreposição das regiões durante o deslocamento do campo receptivo. Por fim, o volume de saída V_s com dimensão $[L_{V_s} \times A_{V_s} \times P_{V_s}]$, é dado por: $L_{V_s} = ((L_V - L_{C_r})/D + 1)$, $A_{V_s} = ((A_V - L_{C_r})/D + 1)$ e $P_{V_s} = P_V$.

A Figura 3.9 mostra um exemplo visual da aplicação da subamostragem na imagem de duas araras-canindé mostrando o volume de entrada e saída. Neste exemplo o volume de entrada tem valores de $L_V = 224$, $A_V = 224$ e $P_V = 64$, e definimos $L_{C_r} = 2$ e $D = 2$. Assim, obtemos o volume de saída V_s com $L_{V_s} = 112$, $A_{V_s} = 112$ e $P_{V_s} = 64$. Podemos observar que as figuras permaneceram similares, apesar da dimensão ter sido reduzida pela metade. Isso é importante pois mostra a preservação das informações da imagem quando aplicamos a subamostragem.

3.3.4 Camada Totalmente Conectada

A camada totalmente conectada calcula a pontuação de todas as classes para a imagem de entrada, basicamente ela faz o caminho de decisão a partir dos resultados das camadas anteriores para cada classe de saída. Nesse bloco,

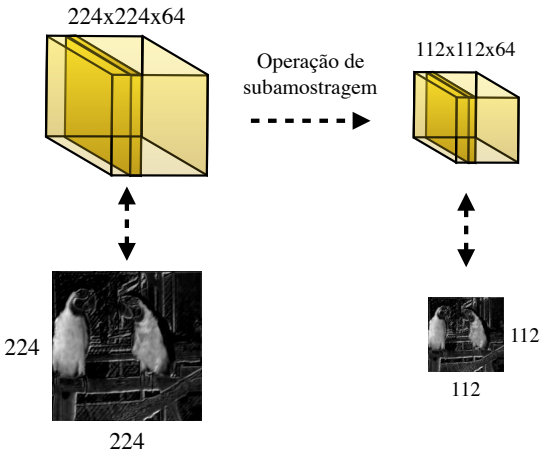


Figura 3.9: Exemplo da execução da subamostragem em uma imagem de duas araras-canindé. O volume de entrada possui dimensão $[224 \times 224 \times 64]$, e a subamostragem foi realizada usando $L_{C_r} = 2$ e $D = 2$.

cada neurônio é conectado com todos os neurônios da camada anterior, e as suas ativações podem ser calculadas com uma multiplicação de matrizes seguido de uma soma do bias. A função de mapeamento usada na camada totalmente conectada é definida pela Equação 3.6:

$$PT_f(i) = B_t(i) + \sum_{j=0}^{L_P} P(i,j) \times PT(i) \tag{3.6}$$

onde PT é o vetor de pontuações recebido da camada anterior, P e B_t são a matriz de pesos e o vetor com os bias de todas as classes analisadas. Vale ressaltar, que se a saída da camada anterior for uma matriz, como ocorre na camada convolucional e de subamostragem, o vetor de pontuações PT é obtido pela concatenação de cada linha da matriz em ordem crescente.

A Figura 3.10 ilustra um exemplo numérico do mapeamento realizado pela camada totalmente conectada. A matriz de pesos P (Figura 3.10 (a)) possui dimensão $[3 \times 3]$, e o vetor de pontuações PT (Figura 3.10 (b)) e o vetor de bias B_t (Figura 3.10 (c)) possuem dimensão $[3]$. Por fim, a Figura 3.10 (d) mostra o cálculo definido na Equação 3.6 que gera o vetor de pontuações de saída PT_s obtido pela camada totalmente conectada. As cores azul, amarelo e verde, representam as três classes analisadas. Assim, a área em azul em P e B_t correspondem aos pesos e bias da primeira classe, e a cor azul em PT_s corresponde a pontuação final para a primeira classe que a CNN gerou analisando o vetor de entrada PT.

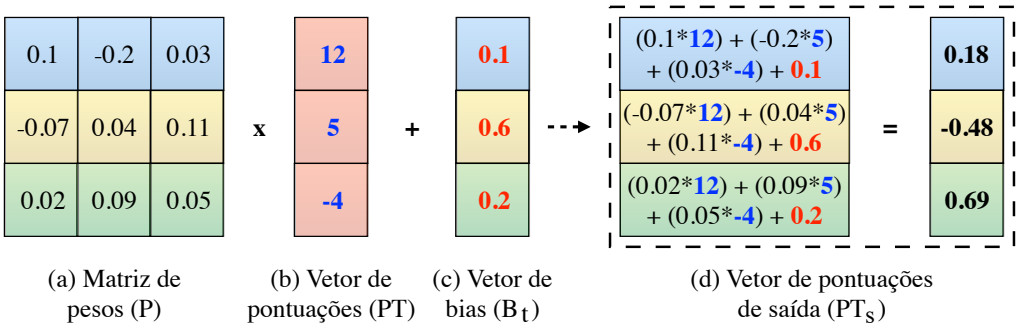


Figura 3.10: Exemplo do mapeamento realizado pela camada totalmente conectada. Na figura (a), (b) e (c) temos a matriz de pesos P, o vetor de pontuações de entrada PT e o vetor de bias B_t. Na figura (d) temos o vetor de pontuações de saída PT_s, gerado pela camada totalmente conectada.

3.3.5 Classificador Softmax

Os classificadores tem a função de mapear a imagem de entrada à uma pontuação para cada classe. Para isso, o classificador usa o vetor de pontuações obtido da camada totalmente conectada e aplica uma função de perda para avaliar a classificação obtida.

A função de perda influência diretamente na aprendizagem dos pesos e, consequentemente, no resultado da CNN. Basicamente, ela mede o quão ruim está sendo a classificação das imagens de treinamento, ou seja, ela mede o quão distante a classe predita está do rótulo real daquela imagem.

Assim, a função de perda será alta se as classes preditas forem diferentes dos rótulos reais, e será baixa se estivermos classificando corretamente as imagens. A partir desse valor podemos ajustar os pesos da CNN de modo a minimizar a perda através do algoritmo de *Backpropagation*.

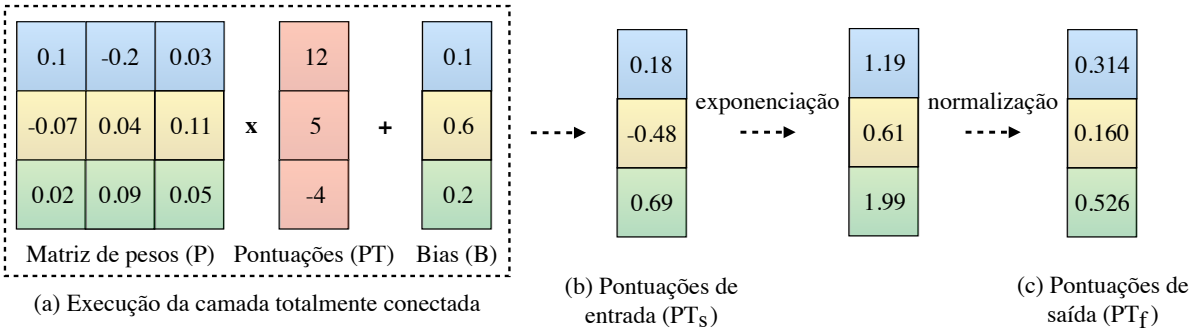
O classificador *softmax* é uma generalização para várias classes do classificador binário de regressão logística. Ele interpreta as pontuações obtidas como probabilidades logarítmicas não normalizadas, e aplica uma função de perda com entropia cruzada [Kline and Berardi, 2005]. Assim, obtém como saída as probabilidades das classes de serem corretas de forma normalizada. A função *softmax* é definida como:

$$f_j(PT_s) = \frac{e^{PT_{sj}}}{\sum_{k=0}^{|PT_s|-1} e^{PT_{sk}}} \tag{3.7}$$

onde $|PT_s|$ representa o tamanho do vetor PT_s. Assim, dado um vetor de pontuações de saída da camada totalmente conectada PT_s com valores reais, a função *softmax* normaliza PT_s para um vetor de valores entre 0 e 1 que juntos totalizam 1.

A Figura 3.11 mostra um exemplo numérico da aplicação do classificador

softmax. Na Figura 3.11 (a) temos a execução da camada totamente conectada que gera o vetor de pontuações PT_s (Figura 3.11 (b)). O *softmax* é aplicado seguindo a Equação 3.7 em cada posição j do vetor PT_s, representados pelas cores azul, amarelo e verde. Por fim, obtemos o vetor de pontuações de saída PT_f (Figura 3.11 (c)) com a probabilidade para cada classe, cuja soma totaliza 1.



(a) Execução da camada totalmente conectada

(b) Pontuações de entrada (PT_S)

(c) Pontuações de saída (PT_f)

Figura 3.11: Exemplo numérico da execução do classificador *softmax*. Na figura (a) temos a execução da camada totalmente conectada que gera o vetor de pontuações PT_s, representado na figura (b). Por fim, na figura (c) temos o vetor de pontuações de saída após aplicar o softmax.

3.4 Arquiteturas de Redes Neurais Convolucionais

Várias arquiteturas de redes neurais convolucionais têm sido propostas ao longo dos anos, nesta seção vamos apresentar duas arquiteturas de redes neurais convolucionais importantes mostrando as suas principais características, vantagens e desvantagens. Na Seção 3.4.1 é apresentada a rede neural convolucional AlexNet proposta por Krizhevsky et al. [2012], e na Seção 3.4.2 é descrita a rede neural convolucional VGGNet proposta por Simonyan and Zisserman [2014].

3.4.1 Rede AlexNet

A rede neural convolucional AlexNet proposta por [Krizhevsky et al., 2012] foi um dos trabalhos que popularizou as redes neurais convolucionais na área de visão computacional. A AlexNet foi submetida ao desafio IMAGEnet ILSVRC em 2012 e superou significativamente o segundo colocado com erro de 16% em relação à 26% do vice-campeão na avaliação top 5.

A AlexNet tem uma arquitetura muito semelhante a rede neural convolucional LeNet [LeCun et al., 1989], porém é maior e mais profunda. Os autores propuseram o conceito de camadas convolucionais empilhadas, o que era in-

comum na época, pois anteriormente era usado uma única camada convolucional seguida de uma camada de subamostragem.

A rede neural convolucional AlexNet contém oito camadas principais onde as cinco primeiras são camadas convolucionais e as três restantes são camadas totalmente conectadas. Além disso, entre essas camadas temos duas camadas de normalização e três camadas de subamostragem.

As camadas de normalização foram colocadas após a primeira e segunda camada convolucional, e as camadas de subamostragem máxima foram colocadas depois das duas camadas de normalização e da última camada convolucional. A função de ativação não-linear ReLu é usada após todas as camadas convolucionais e totalmente conectadas. A Figura 3.12 mostra a estrutura da rede neural convolucional AlexNet.

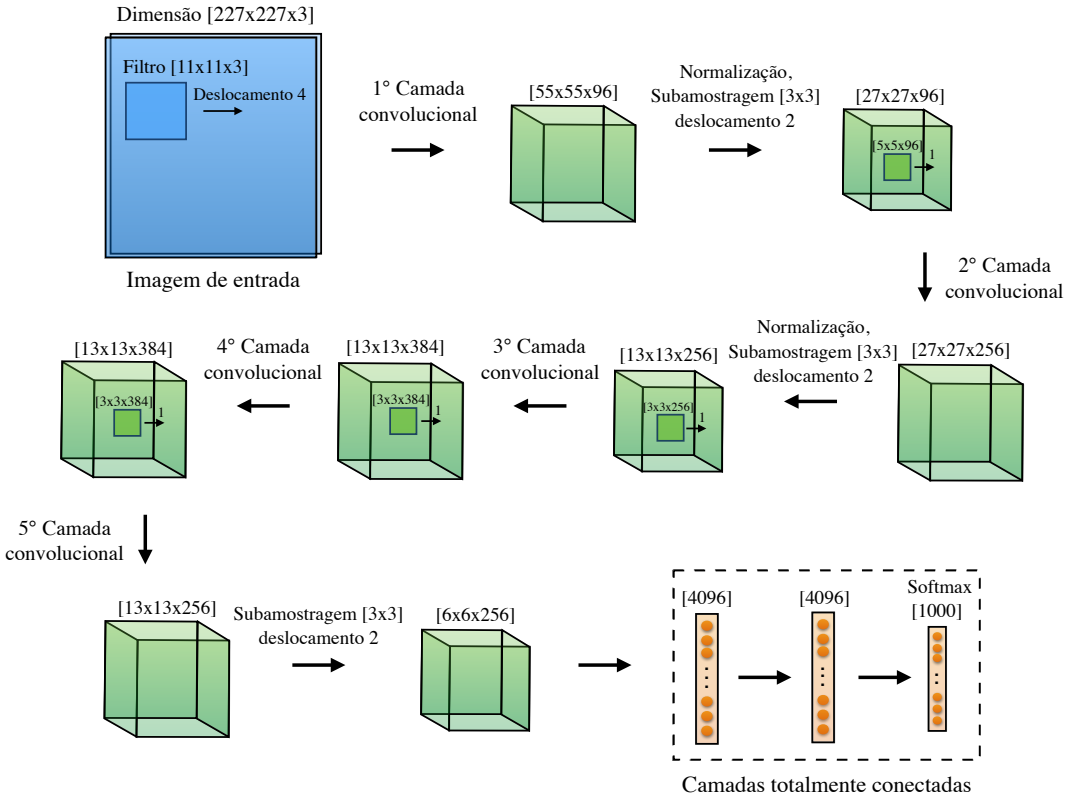


Figura 3.12: Imagem da arquitetura da rede neural convolucional AlexNet submetida ao desafio IMAGEnet ILSVRC em 2012.

De acordo com a Figura 3.12 pode-se observar que a AlexNet recebe como entrada uma imagem RGB com dimensão $[227 \times 227 \times 3]$, e a primeira camada convolucional aplica 96 filtros de tamanho $[11 \times 11 \times 3]$ sobre essa imagem, através de um deslocamento de 4 pixels, já a segunda camada usa 256 filtros de tamanho $[5 \times 5 \times 96]$ sobre a saída da primeira camada de subamostragem máxima.

A terceira camada convolucional aplica 384 filtros de tamanho $[3 \times 3 \times 256]$ na saída da segunda camada de subamostragem máxima, e a quarta e quinta

camada convolucional aplicam filtros de dimensão $[3 \times 3 \times 384]$ usando 384 e 256 filtros, respectivamente. Em seguida, a terceira camada de subamostragem máxima é realizada sobre a saída da quinta camada convolucional.

Por fim, a arquitetura é composta por três camadas totalmente conectadas, onde as duas primeiras possuem 4096 neurônios, e a última possui 1000 neurônios que é ligado a um classificador *softmax* que produz uma distribuição das 1000 classes do desafio IMAGEnet.

3.4.2 Rede VGGNet

A rede neural convolucional VGGNet foi proposta por Simonyan and Zisserman [2014] na ILSVRC 2014 e trouxe também uma importante contribuição ao desenvolvimento das redes neurais convolucionais na área de visão computacional. Ela mostrou que a profundidade da CNN é um componente crítico para o bom desempenho.

Os autores apresentaram seis arquiteturas da VGGNet: duas com 11, uma com 13, duas com 16, e uma com 19 camadas. Porém, a sua melhor CNN contém 16 camadas convolucionais e totalmente conectadas, e é uma arquitetura homogênea pois só executa convoluções com campo receptivo $[3 \times 3]$ e subamostragens com campo receptivo $[2 \times 2]$. A Figura 3.13 mostra a estrutura da rede neural convolucional VGGNet de 16 camadas.

A desvantagem da VGGNet é que quanto maior a profundidade mais caro fica para avaliar a CNN, e ela acaba usando mais memória e parâmetros. A maioria destes parâmetros estão na primeira camada totalmente conectada, e foi observado que esse tipo de camada pode ser removida sem perda de desempenho, reduzindo significativamente o número de parâmetros necessários.

A rede neural convolucional VGGNet recebe como entrada uma imagem RGB com dimensão $[224 \times 244 \times 3]$, que passa por um pré-processamento inicial no qual é realizada a subtração da cor média de todos os pixels da imagem. Esse processo é repetido para todas as imagens que serão usadas para o treinamento da CNN.

As imagens passam pelas camadas convolucionais que aplicam um filtro de tamanho $[3 \times 3]$. Os autores usaram esse tamanho de filtro por ser o menor tamanho capaz de aprender características das quatro direções e do centro durante o processo de convolução. Esse tamanho de filtro é usado em todas as arquiteturas, exceto em uma que aplica um filtro $[1 \times 1]$ que age como uma transformação linear das entradas.

O passo de deslocamento do filtro convolucional é de 1 pixel, e o preenchimento com zeros ao redor da imagem é realizado de forma a preservar a dimensão da imagem após a convolução, ou seja, para os filtros de tamanho $[3 \times 3]$ é usado um preenchimento com zeros de tamanho 1.

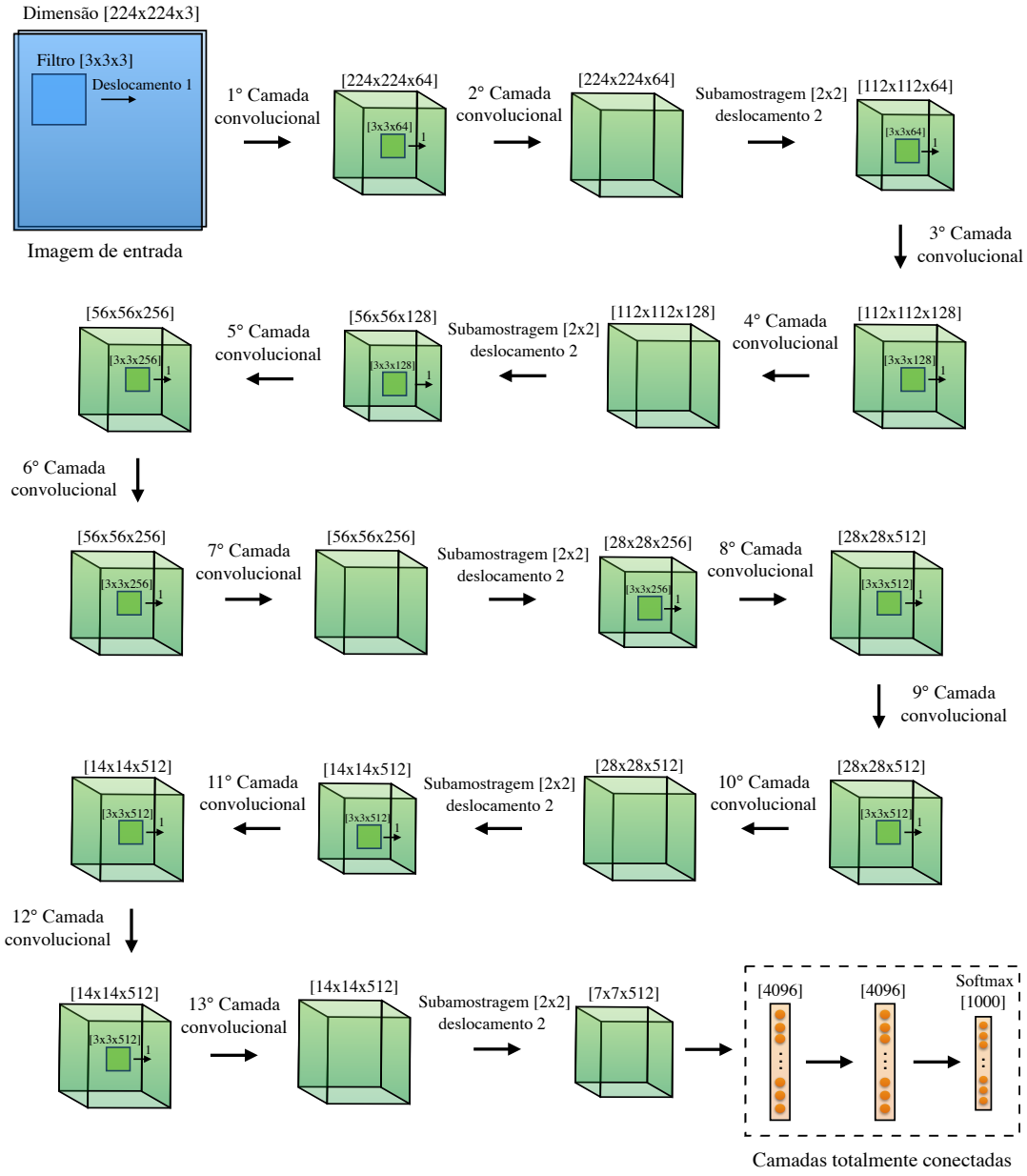


Figura 3.13: Imagem da estrutura usada na rede neural convolucional VGG-Net de 16 camadas submetida ao desafio IMAGEnet ILSVRC em 2014.

A amostragem descendente é feita através de cinco camadas de subamostragens distribuídas no corpo da CNN. Assim, nem todas camadas convolucionais são seguidas de camadas de subamostragens. Os autores usam uma camada de subamostragem máxima com campo receptivo $[2 \times 2]$ e deslocamento de 2 pixels.

As arquiteturas são compostas por três camadas totalmente conectadas, onde as duas primeiras possuem 4096 neurônios cada e a última contém 1000 neurônios ligados a um *softmax* que faz a distribuição das 1000 classes do desafio IMAGEnet.

Além disso, os autores empregaram após todas as camadas a função de ativação não-linear ReLu, e eles não usam camadas de normalização, exceto em uma das arquiteturas. Eles verificaram que as camadas de normalização

não melhoram o desempenho da CNN na classificação das imagens do IMAGEnet, ao contrário, ela só aumenta o consumo de memória e processamento. Na arquitetura onde eles aplicaram essa camada, eles usaram os mesmos parâmetros usados pela AlexNet.

3.5 Redes Neurais Convolucionais Baseadas em Regiões de Interesse

O principal desafio do uso de redes neurais convolucionais para a identificação de objetos de interesse é a necessidade de encontrar a sua localização na imagem. Isso acontece pois podemos ter muitos objetos na imagem, e cada um pode possuir uma classe diferente. Assim, precisamos primeiro fazer a localização dele na imagem para depois fazermos a identificação da sua classe.

A localização do objeto de interesse na imagem requer muito processamento, pois existe um número muito grande de posições e escalas que a região em que o animal está pode assumir na imagem. Para resolver esse problema, foi proposta a análise de apenas uma parte dessas possíveis regiões, que basicamente seleciona e analisa algumas regiões consideradas mais prováveis de se conter o objeto de interesse.

3.5.1 Detecção de Regiões de Interesse

O processo de identificação das possíveis regiões de interesse é chamado de detecção de regiões de interesse (ou proposição de regiões - *Region Proposals*), onde basicamente o algoritmo busca na imagem as regiões que provavelmente contêm um objeto de interesse. Essa tarefa é realizada sem se importar com a classe do objeto, identificando e definindo as regiões semelhantes na imagem.

Existem vários algoritmos que fazem esse trabalho, e eles se dividem em dois tipos de abordagem, as técnicas baseadas em agrupamento [Arbeláez et al., 2014, Uijlings et al., 2013] e as técnicas baseadas em pontuação de janela [Zitnick and Dollár, 2014, Alexe et al., 2012]. O primeiro tipo de abordagem se caracteriza pela geração de segmentos na imagem que podem corresponder aos objetos de interesse. O segundo tipo se caracteriza pela pontuação da janela candidata de acordo com a probabilidade de conter um objeto de interesse.

O algoritmo de pesquisa seletiva (*Selective Search*) proposto por Uijlings et al. [2013] faz parte das técnicas baseadas em agrupamento, sendo o algoritmo de detecção de regiões de interesse mais usado atualmente. Ele faz o agrupamento das regiões com uma abordagem de baixo para cima, ou seja, o algoritmo começa a partir de um pixel e vai agrupando os pixels próximos, que

possuam cores ou texturas similares. Inicialmente, o algoritmo de pesquisa seletiva emprega o algoritmo proposto por [Felzenszwalb and Huttenlocher, 2004] para dividir uma imagem I em um conjunto de regiões iniciais R.

O algoritmo proposto por [Felzenszwalb and Huttenlocher, 2004] gera as regiões usando uma abordagem baseada em grafos. Primeiro, ele define os pixels como vértices do grafo, e a diferença de intensidade dos pixels como os pesos das arestas, obtendo um grafo $G = \{V, A\}$, onde $V = \{v_1, v_2, ..., v_n\}$ é o conjunto de vértices, e $A = \{a_1, a_2, ..., a_m\}$ é o conjunto de arestas. Em seguida, o algoritmo considera todos os vértices como componentes iniciais, dado por $C = \{C_1, C_2, ..., C_k\}$, que serão usadas para definir as regiões de interesse. A partir disso, o algoritmo verifica duas componentes distintas C_i e C_j, e armazena a aresta de peso mínimo a_{min} que conecta as duas componentes, através da fórmula:

$$a_{min} = \min_{v_i \in C_i, v_j \in C_j, (v_i, v_j) \in A} (p(v_i, v_j)) \tag{3.8}$$

onde v_i e v_j são vértices do grafo G, e $p(v_i, v_j)$ é o peso da aresta que conecta os vértices v_i e v_j. Além disso, para cada componente C_i o algoritmo armazena a sua aresta de peso máximo a_{max_i} após aplicar uma árvore geradora mínima (*minimum spanning tree* - MST) através da fórmula $a_{max_i} = \max_{a \in MST(C_i, A)} (p(a))$. Assim, o algoritmo verifica se $a_{min} \leq min(a_{max_i}, a_{max_j})$, se a condição for satisfeita as componentes C_i e C_j serão unidas em uma nova componente, se não o algoritmo continua a execução verificando outras componentes. Assim, o processo é repetido até que o algoritmo não encontre mais nenhum par de componentes distintas que possuam arestas que satisfaçam essas condições, retornando as componentes como regiões de interesse.

Com o conjunto de regiões $R = \{r_1, r_2, ..., r_n\}$ geradas pelo algoritmo de Felzenszwalb and Huttenlocher [2004], o algoritmo de pesquisa seletiva calcula a similaridade para todo par $(r_i, r_j) \in R$, representada por $s(r_i, r_j)$, e armazena todas as similaridades obtidas em um conjunto S.

Os autores [Uijlings et al., 2013] definiram quatro similaridades para analisar as regiões geradas. A primeira é a similaridade de cor entre duas regiões, calculada através da análise de um histograma de 25 bins de cada canal da imagem, parâmetro definido pelos autores [Uijlings et al., 2013]. Basicamente, o algoritmo obtém um histograma de cor $C_i = \{c_i^1, c_i^2, ..., c_i^{d_c}\}$ para cada região r_i com dimensionalidade $d_c = 75$, e usa a interseção dos histogramas para obter a similaridade das regiões. Assim, a similaridade de cor é calculada por:

$$s_{cor}(r_i, r_j) = \sum_{k=1}^{d_c} min(c_i^k, c_j^k) \tag{3.9}$$

A segunda similaridade calculada é a de textura, obtida por meio da análise de um conjunto de histogramas de 10 bins cada, parâmetro definido pelos autores [Uijlings et al., 2013], obtido para cada direção e para cada canal da imagem. Basicamente, o algoritmo constrói um histograma de textura $T_i = \{t_i^1, t_i^2, ..., t_i^{d_t}\}$ para cada região r_i com dimensionalidade $d_t = 240$, e analisa a intersecção dos histogramas das regiões para obter a similaridade. Assim, a similaridade de textura é definida por:

$$s_{tex}(r_i, r_j) = \sum_{k=1}^{d_t} min(t_i^k, t_j^k) \tag{3.10}$$

A terceira similaridade que os autores analisam é a do tamanho das regiões, que procura agrupar as regiões menores com as regiões maiores, fazendo com que as regiões menores se unam mais rapidamente. Essa similaridade é calculada por:

$$s_{tam}(r_i, r_j) = 1 - \frac{tam(r_i) + tam(r_j)}{tam(I)} \tag{3.11}$$

onde $tam(I)$ é a quantidade de pixels da imagem I. Por fim, a quarta similaridade proposta é a de forma, que procura unir mais rapidamente as regiões que estão sobrepostas ou próximas. Para isso ela cria uma região BB_{ij} que delimita duas regiões r_i e r_j, e assim consegue avaliar o resultado gerado pela união das regiões. Essa similaridade é dada por:

$$s_{for}(r_i, r_j) = 1 - \frac{tam(BB_{ij}) - tam(r_i) - tam(r_j)}{tam(I)} \tag{3.12}$$

Em seguida, o algoritmo de pesquisa seletiva utiliza todas as similaridades em uma similaridade geral dada por:

$$s(r_i, r_j) = a_1 s_{cor}(r_i, r_j) + a_2 s_{tex}(r_i, r_j) + a_3 s_{tam}(r_i, r_j) + a_4 s_{for}(r_i, r_j) \tag{3.13}$$

onde podemos usar as variáveis (a_1, a_2, a_3 e a_4) para definir a importância de cada similaridade, permitindo uma maior flexibilidade para diferentes tipos de aplicações.

Após o cálculo das similaridades, o algoritmo extraí a maior similaridade $s(r_i, r_j)$ do conjunto S, e une as duas regiões correspondentes. Assim, se $s(r_i, r_j) = max(S)$ temos que uma nova região r_t será criada, onde $r_t = r_i \cup r_j$. Após unir as duas regiões que possuem maior similaridade o algoritmo pesquisa no conjunto S e remove todas as similaridades existentes com as regiões unidas, representada por $s(r_i, r_*)$ e $s(r_*, r_j)$. Em seguida, ele calcula a similaridade de r_t com os seus vizinhos, atualizando o conjunto de similaridades S, e armazena a nova região r_t no conjunto de regiões R.

Esse processo é repetido até que o conjunto de similaridades S esteja vazio, e toda a imagem seja formada por apenas uma região. Portanto, o conjunto R terá todas as regiões geradas durante a execução. Assim, o algoritmo de pesquisa seletiva termina a execução definindo todas as regiões do conjunto R como um região de interesse.

A Figura 3.14 mostra um exemplo das regiões de interesse obtidas pelo algoritmo de pesquisa seletiva sobre a imagem de uma arara-canindé. Na Figura 3.14 (a) temos um imagem de entrada com duas araras-canindé que foi aplicado o algoritmo de pesquisa seletiva, na Figura 3.14 (b) temos as regiões de interesse finais que foram detectadas pelo algoritmo, e na Figura 3.14 (c) temos os boxes plotados na imagem com base nas regiões propostas.

(a) Imagem de entrada (b) Regiões detectadas (c) Regiões de interesse

Figura 3.14: Exemplo de regiões de interesse obtidas com o algoritmo de pesquisa seletiva sobre uma imagem de duas araras-canindé. Na figura (a) temos a imagem de entrada, na figura (b) temos as regiões detectadas, e na figura (c) temos os boxes plotados.

3.5.2 *Algoritmos de Redes Neurais Convolucionais Baseadas em Regiões de Interesse*

O algoritmo *R-CNN* foi proposto por Girshick et al. [2014] com o intuito de unir a ideia de detecção de regiões de interesse com as redes neurais convolucionais. Basicamente, o algoritmo aplica a pesquisa seletiva nas imagens gerando em torno de duas mil regiões de interesse, e cada região é classificada

pela rede neural convolucional.

A Figura 3.15 mostra a abordagem *R-CNN*. No passo (1) é realizada a extração das regiões de interesse na imagem de entrada. No passo (2) as regiões obtidas são redimensionadas para o tamanho de entrada da CNN. Por fim, no passo (3) executamos a rede neural convolucional sobre cada região gerada, obtendo o desempenho da localização e classificação da região através de um classificador SVM e um regressor linear baseado em limite de caixa.

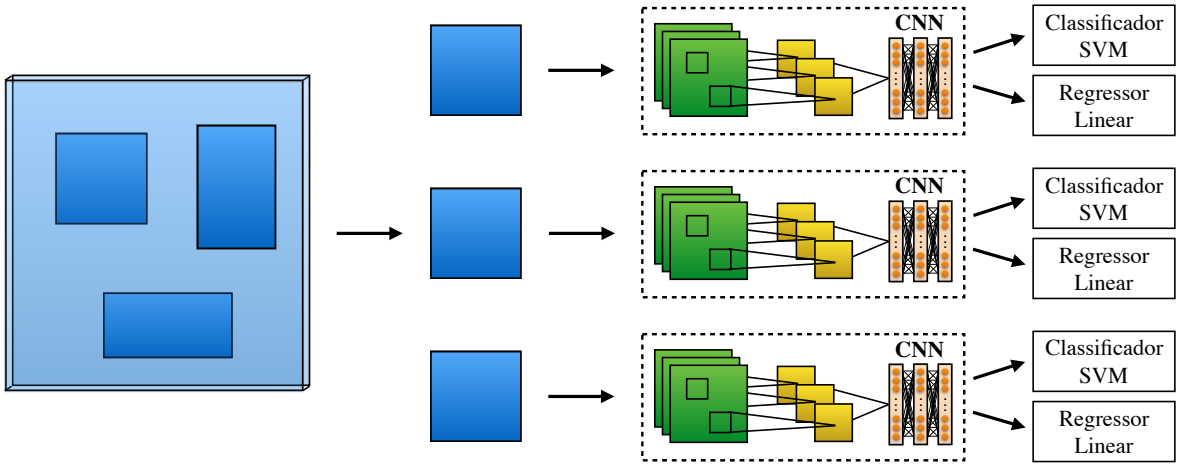


Figura 3.15: Diagrama da proposta do algoritmo *R-CNN*. No passo (1) é realizada a extração das regiões de interesse, no passo (2) as regiões são redimensionadas para serem processadas pela CNN, e no passo (3) executamos a rede neural convolucional sobre cada região gerada, para a classificação da região.

A regressão linear baseada em limite de caixa é usada para fazer um ajuste fino das regiões de interesse obtidas pela CNN, afim de conseguir uma melhor estimativa da posição do objeto de interesse. Esse tipo de ajuste é importante pois a região de interesse inicial, pode não conter completamente o objeto de interesse.

Uma vez aprendida a classe da região de interesse, a regressão linear procura melhorar o formato da região inicial r com base no aprendizado de outras regiões de interesse da mesma classe. Esse tipo de regressão possui um melhor desempenho quando comparados aos métodos de detecção de regiões de interesse simples, pois ele usa as características de aprendizado da rede neural convolucional para melhorar as regiões. A Figura 3.16 mostra um exemplo visual da etapa de regressão linear para uma região de interesse em uma imagem de uma onça-parda. Basicamente, temos uma região inicial r, representada pela região de cor verde na imagem, e aplicamos uma regressão linear $d(r)$, obtendo uma região g, representada pela região de cor amarela.

Apesar do algoritmo *R-CNN* ter trazido melhoras significativas na localização e classificação dos objetos, ele possui alguns problemas importantes que

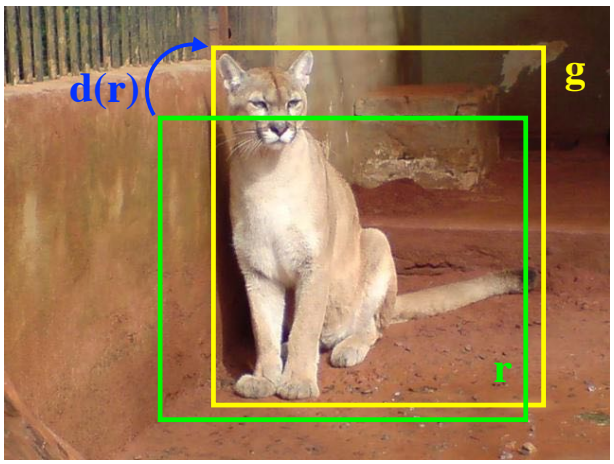


Figura 3.16: Exemplo de uma regressão linear feita sobre uma região de interesse em uma imagem de onça-parda. A área r destacada em verde representa a região proposta inicialmente, e a região g em amarelo representa a região em que foi aplicada uma regressão linear $d(r)$.

podem ser melhorados. O primeiro é o tempo de teste que é lento devido à necessidade de executar a CNN em cada região, e isso é muito custoso se considerarmos que são geradas por volta de duas mil regiões (Figura 3.14 (c)). O segundo é que a aplicação do SVM e da regressão linear na CNN são realizados de modo off-line, não permitindo a atualização dos pesos da CNN na mudança do processamento de uma região para a outra. O terceiro é a complexidade no processo de treinamento dessa CNN que possui vários estágios.

Para superar esses problemas Ren et al. [2015] propuseram o *Fast R-CNN* que também usa a detecção de regiões de interesse com a rede neural convolucional. Porém, eles mudaram a ordem em que é realizada a extração das regiões e a execução da rede neural convolucional, afim de fazer apenas um processamento da CNN.

Para isso, o algoritmo recebe como entrada a imagem inteira e as regiões de interesse, mas utiliza as regiões somente no meio da execução da CNN. O algoritmo obtêm primeiro um mapa de características gerado na última camada convolucional da CNN, e depois aplica as regiões de interesse sobre ele. Assim, o algoritmo faz a extração das características desse mapa que correspondem as regiões de interesse, e as usa para a classificação das regiões.

O *Fast R-CNN* resolve os três problemas do *R-CNN* de forma intuitiva. Para o problema do elevado tempo na etapa de teste o algoritmo usa a computação compartilhada das camadas convolucionais entre as regiões de interesse, melhorando consideravelmente o tempo de execução do teste. Para o problema de execução off-line sem atualização dos pesos da CNN e, da complexidade do processo de treinamento, o *Fast R-CNN* propõe o treinamento de todo o sistema de uma só vez sem a necessidade de treinar todas as regiões de interesse separadamente.

CAPÍTULO

4

Uma Nova Abordagem para Identificação de Animais Silvestres

Neste capítulo é apresentada a abordagem para o reconhecimento de espécies animais usando imagens térmicas. A abordagem proposta foi dividida em duas partes: a primeira é descrita na Seção 4.1 e apresenta o método usado para a detecção das regiões de interesse usando SLIC em imagens térmicas, e a segunda parte é apresentada na Seção 4.2 e descreve o processo de classificação das espécies usando redes neurais convolucionais.

4.1 Detecção de Animais Usando Imagens Térmicas

A abordagem proposta para a detecção de regiões de interesse é baseada no algoritmo de segmentação SLIC (*Simple Linear Iterative Clustering*), proposto por Achanta et al. [2012]. A ideia aqui consiste em agrupar os pixels da imagem em regiões que possuam similaridade baseada na emissão de calor dos corpos dos animais. Uma vez que o calor do corpo tende a ser maior que a temperatura média do ambiente, os atributos de cor são definidos com mais intensidade segundo uma matriz de calor da câmera térmica.

O superpixel permite usar as características locais dos pixels da imagem, ao contrário do algoritmo k-médias [Hartigan and Wong, 1979] que considera todo espaço de busca no agrupamento. A Figura 4.1 mostra o diagrama da nossa abordagem para a detecção de regiões de interesse usando as imagens térmicas.

Seja $I \in \Re^{l \times a}$ uma imagem RGB e $T \in \Re^{l \times a}$ a sua correspondente imagem térmica (Figura 4.1 (1)), onde l e a representam à largura e a altura das ima-

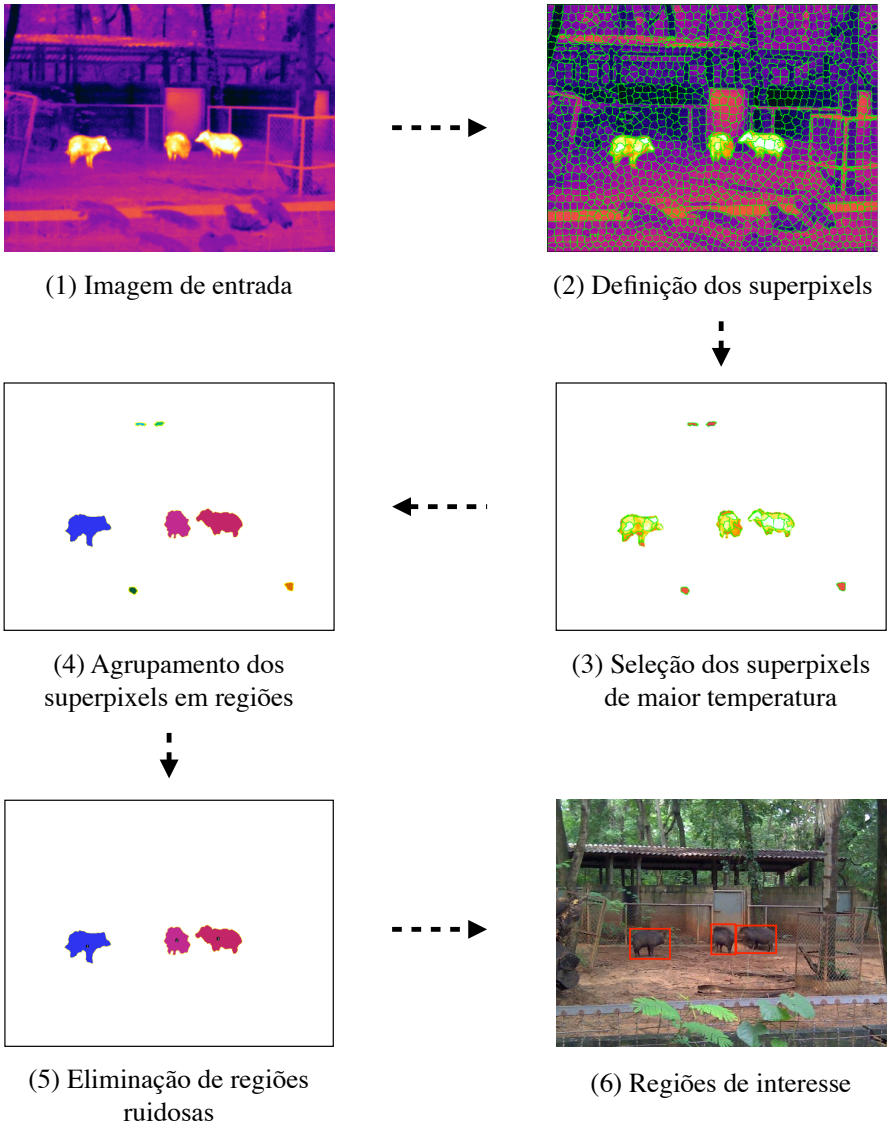


Figura 4.1: Etapa de detecção de regiões de interesse usando as imagens térmicas. No passo (1) é realizado captura das imagens, no passo (2) são gerados os superpixels com o SLIC, no passo (3) são selecionados os superpixels de maior temperatura, no passo (4) é realizada a segmentação dos animais usando componentes conectados, no passo (5) são eliminadas regiões ruidosas, e no passo (6) obtemos as regiões de interesse.

gens. Para cada posição de $T(x, y)$ a câmera térmica fornece a temperatura em graus °C. Primeiro, calculamos a temperatura média da imagem $Temp_m$, através da equação:

$$Temp_m = \frac{1}{l \times a} \sum_{x=1}^{l} \sum_{y=1}^{a} T(x, y) \tag{4.1}$$

Em seguida, aplicamos a técnica SLIC [Achanta et al., 2012] sobre a imagem térmica T e obtemos um conjunto de superpixels $S = \{s_1, s_2, ..., s_k\}$ (Figura 4.1 (2)), onde k é o número total de superpixels gerados. Com isso, podemos calcular também a temperatura média para cada superpixel s_i de acordo com a equação a seguir:

$$Temp_{s_i} = \frac{1}{N_{s_i}} \sum_{(x,y) \in s_i} T(x, y) \tag{4.2}$$

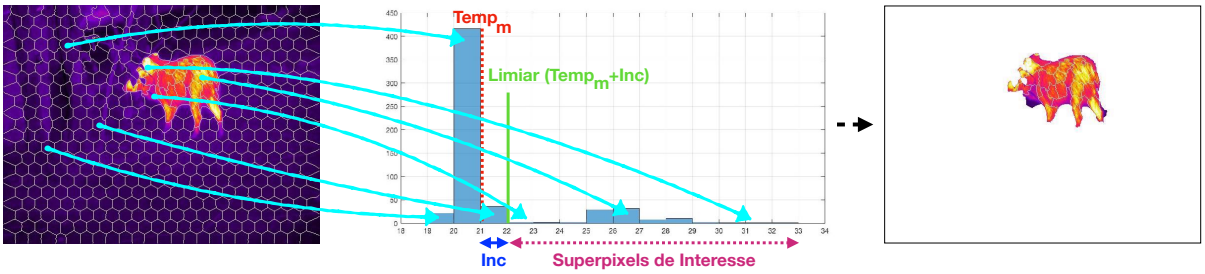
onde N_{s_i} é o número total de pixels que compõem o superpixel s_i. Afim de separar os superpixels que pertencem ao animal ou à região de fundo na imagem, a abordagem verifica se a temperatura de um superpixel s_i é maior que a temperatura ambiente média $Temp_m$, através da equação:

$$Temp_{s_i} > Temp_m + Inc \tag{4.3}$$

onde Inc é um incremento que auxilia na separação dos superpixels que pertencem ao animal ou a região de fundo. Assim, construímos um limiar adaptativo que considera a temperatura média da imagem e um incremento para a seleção dos superpixels de maior temperatura (do passo (2) para o passo (3) da Figura 4.1).

A Figura 4.2 mostra um exemplo da separação dos superpixels de maior temperatura considerando um histograma que apresenta a distribuição das temperaturas dos superpixels da imagem. Basicamente, recebemos uma imagem com os superpixels definidos (Figura 4.2 (a)), e eles são selecionados considerando a temperatura média da imagem $Temp_m$, representada pela linha vermelha, e um incremento Inc, representado pela seta de cor azul. Assim, geramos um limiar adaptativo, dado por $Temp_m + Inc$, que é representado pela linha de cor verde na Figura 4.2 (b). Por fim, obtemos os superpixels maior temperatura na imagem térmica (Figura 4.2 (c)).

Como podemos ter imagens com diferentes distribuições de temperaturas dos superpixels, um limiar gerado com incremento fixo Inc pode não delimitar



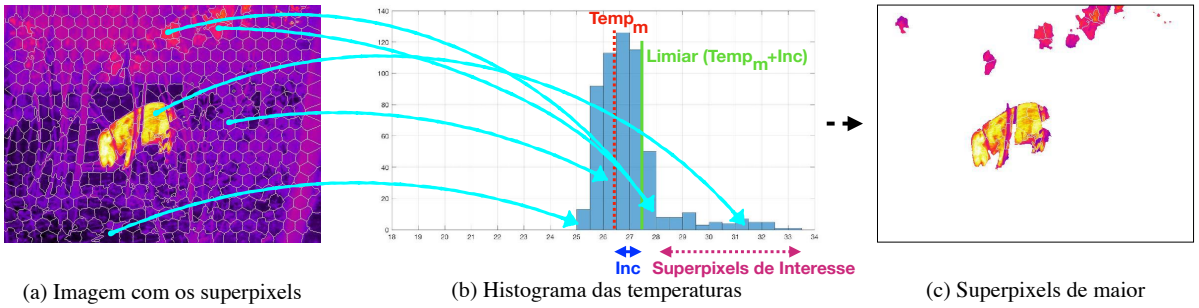
(a) Imagem com os superpixels

(b) Histograma das temperaturas dos superpixels

(c) Superpixels de maior temperatura selecionados

Figura 4.2: Ilustração do processo de seleção dos superpixels mais quentes considerando a temperatura média da imagem e um incremento $Inc = 1$. Na figura (a) temos a imagem térmica com os superpixels definidos, na figura (b) temos o histograma das temperaturas dos superpixels, e na figura (c) temos os superpixels selecionados.

corretamente os superpixels de interesse, mesmo se adaptando a temperatura média de cada imagem. A Figura 4.3 mostra um exemplo onde os superpixels que representam região de fundo possuem temperatura próxima a dos superpixels de interesse, que representam o animal. Nesse caso, se aplicarmos o mesmo incremento $Inc = 1$ usado no exemplo da Figura 4.2, não vamos separar os superpixels de interesse corretamente, pois a temperatura média nessa imagem é de $26,5°C$ e os superpixels que representam o animal estão na faixa de $28°C$ a $33,5°C$, logo, selecionaremos superpixels da faixa de $27,5°C$ a $28°C$ que correspondem a região de fundo (Figura 4.3 (c)).



(a) Imagem com os superpixels

(b) Histograma das temperaturas dos superpixels

(c) Superpixels de maior temperatura selecionados

Figura 4.3: Exemplo de distribuição de temperatura de superpixels de regiões de fundo próxima aos dos superpixels do animal e uma seleção dos superpixels considerando $Inc = 1$. Na figura (a) temos a imagem térmica com os superpixels definidos, na figura (b) temos o histograma das temperaturas dos superpixels, e na figura (c) temos os superpixels selecionados.

Por esse motivo, a nossa abordagem também permite que o valor de Inc seja determinado de forma fixa ou variável através de um parâmetro de seleção T_{Inc}, que define o tipo do incremento que será usado. Se T_{Inc} for definido como fixo, recebemos Inc por parâmetro e aplicamos a Equação 4.3. Caso T_{Inc}

seja definido como variável, criamos um conjunto de incrementos para serem aplicados na Equação 4.3.

A Figura 4.4 mostra um exemplo dos limiares gerados com os incrementos variáveis. Basicamente, criamos os limiares variáveis com a Equação 4.3, representados pelas linhas verdes, considerando um conjunto de incrementos Inc e a temperatura média da imagem, representada pela linha de cor vermelha. Os incrementos variam com uma taxa dis_f, representada pela seta de cor marrom, que é recebida por parâmetro e indica a distância desejada entre os limiares. Além disso, os incrementos Inc_i começam em 0 e variam até um limite $\sigma(C_{Temp})$, onde σ é o desvio padrão, representado pela seta de cor laranja, e C_{Temp} é o conjunto de temperaturas dos superpixels, dada por $C_{Temp} = \{Temp_{s_1}, Temp_{s_2}, ..., Temp_{s_k}\}$, onde k é o número total de superpixels. O desvio padrão foi adotado para limitar os incrementos pois o limiar máximo será a soma da média com o desvio padrão, e eliminará aproximadamente 85% dos superpixels da imagem.

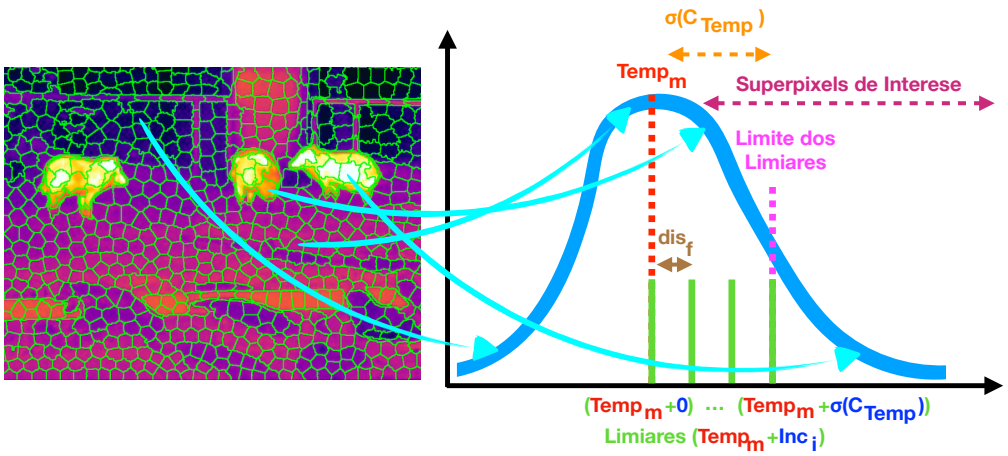


Figura 4.4: Exemplo de distribuição das temperaturas médias dos superpixels e definição de limiares variáveis baseado no desvio padrão e média.

Em seguida, nós agrupamos os superpixels que satisfazem a Equação 4.3 em regiões através do algoritmo de componentes conectados com vizinhança-de-8 pixels [Haralick and Shapiro, 1992, Rosenfeld, 1970], descrito na Seção 3.2. Assim, obtemos um conjunto de regiões $R = \{r_1, r_2, ..., r_z\}$ (Figura 4.1 (4)), onde z é o número total de regiões agrupadas, e cada região é definida por $r_i = \{s_1 \cup s_2 \cup ... \cup s_{N_{r_i}}\}$, onde N_{r_i} é o número total de superpixels da região r_i. Portanto, um superpixel s_i só pertencerá a uma região r_i, se para um dado j, temos que $s_j \in r_i$ e pelo menos um pixel de s_i é considerado vizinhança-de-8 pixels de um pixel de s_j e se respeita a Equação 4.3.

Por fim, consideramos as regiões de maior quantidade de pixels para eliminar pequenas regiões ruidosas que não representam nenhum animal na imagem (do passo (4) para o passo (5) da Figura 4.1), e.g. como pedras ou

formigueiros que podem estar mais quentes que o ambiente. Para isso, primeiro obtemos a região com maior quantidade de pixels do conjunto R com $M_r = \max_{i=1}^{z}(Tam_{r_i})$, onde Tam_{r_i} é o tamanho em pixels da região r_i. Em seguida, verificamos para toda região r_i, se $Tam_{r_i} > \frac{M_r}{p_{min}}$, onde p_{min} é uma variável que define a menor proporção aceita entre Tam_{r_i} e M_r. Por exemplo, se $p_{min} = 4$ todas as regiões r_i que tiverem tamanho Tam_{r_i} maior que 25% da região M_r serão definidas como uma região de interesse da imagem I (Figura 4.1 (5)).

Assim, o conjunto de regiões de interesse que localizam os animais é definido como $ROI = \{roi_1, roi_2, ..., roi_n\}$ (boxes vermelhos da Figura 4.1 (6)), onde cada região roi_i é uma região de interesse detectada pela abordagem, e é definida por $roi_i = \{x_{roi_i}, y_{roi_i}, l_{roi_i}, a_{roi_i}\}$ onde x_{roi_i} e y_{roi_i} representam as posições espaciais x e y da região de interesse roi_i na imagem I, sendo definidas no canto superior esquerdo da região e, l_{roi_i} e a_{roi_i} representam a largura e altura da região roi_i.

A Figura 4.5 ilustra uma região $roi_i = \{x_{roi_i}, y_{roi_i}, l_{roi_i}, a_{roi_i}\}$ detectada pela abordagem. A área destacada em vermelha representa a região detectada, o ponto de cor azul no canto superior esquerdo representa o ponto inicial da roi_i com as coordenadas (x_{roi_i}, y_{roi_i}). Já as setas de cor azul representam a largura l_{roi_i} e a altura a_{roi_i} da roi_i.

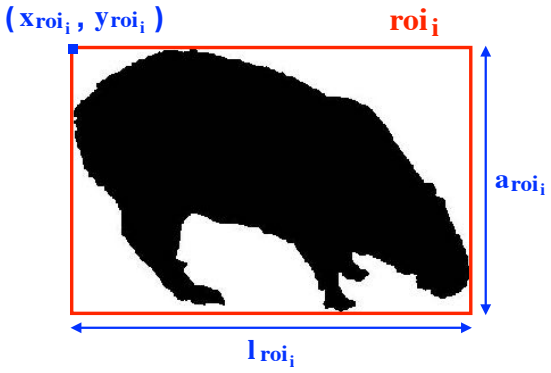


Figura 4.5: Exemplo das dimensões x_{roi_i}, y_{roi_i}, l_{roi_i} e a_{roi_i} de uma região roi_i detectada pela abordagem.

4.2 Classificação dos Animais com Redes Neurais Convolucionais

A abordagem proposta para a identificação das espécies de animais é realizada usando as redes neurais convolucionais. Nós escolhemos as redes neurais convolucionais devido ao recente desempenho em várias tarefas de visão computacional [Krizhevsky et al., 2012, Simonyan and Zisserman, 2014], obtidos principalmente pelas arquiteturas AlexNet [Krizhevsky et al., 2012],

VGGNet [Simonyan and Zisserman, 2014] e GoogLeNet [Szegedy et al., 2015].

A etapa de classificação dos animais com CNN é similar ao *Fast R-CNN* [Girshick, 2015], porém foram utilizadas as regiões de interesse obtidas pela saída do algoritmo SLIC sobre as imagens térmicas, conforme descrito na Seção 4.1. A Figura 4.6 mostra o diagrama da nossa abordagem para a identificação das espécies animais usando redes neurais convolucionais.

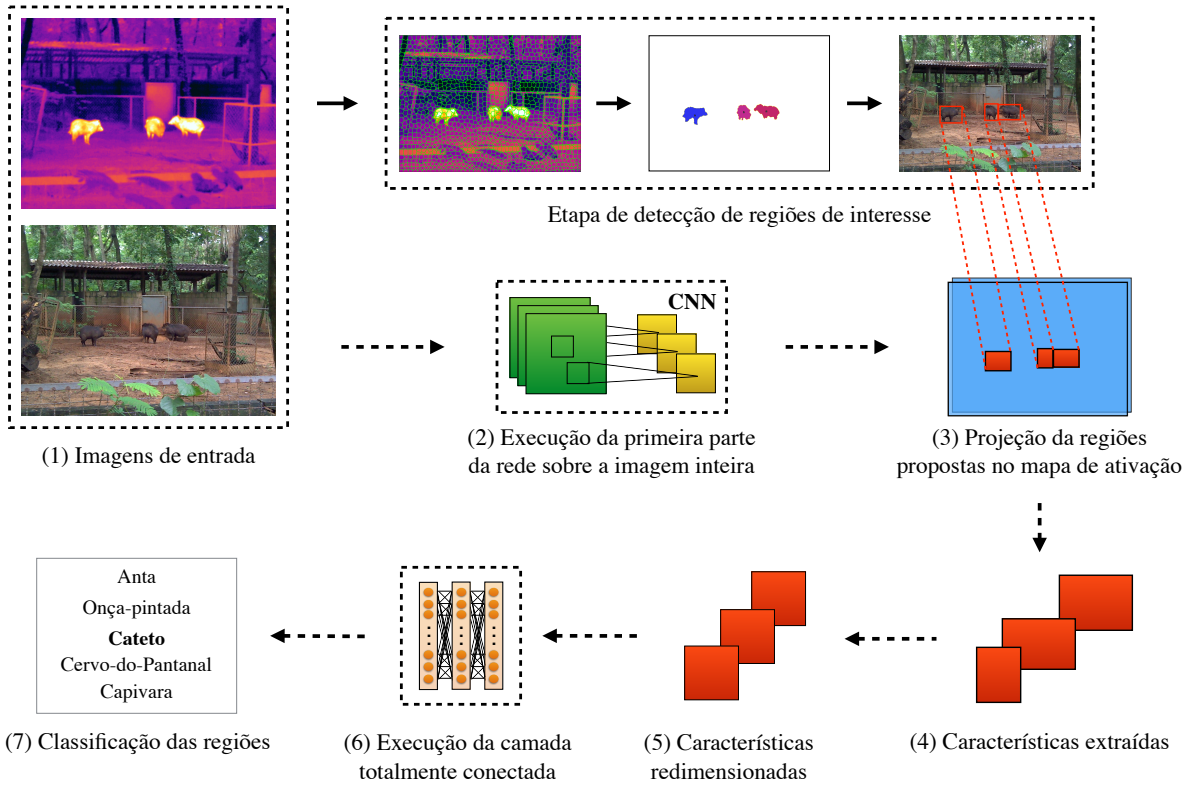


Figura 4.6: Etapa de identificação das espécies animais usando rede neural convolucional. No passo (1) recebemos como entrada as imagens RGB e térmica, no passo (2) executamos a primeira parte da CNN sobre a imagem RGB inteira para obtermos o mapa de características da última camada convolucional, no passo (3) fazemos a projeção das regiões propostas pela primeira etapa da abordagem sobre o mapa de características, no passo (4) e (5) realizamos a extração das características e fazemos o seu redimensionamento, no passo (6) executamos a camada totalmente conectada com as características redimensionadas e no passo (7) fazemos a classificação das regiões.

Na etapa de treinamento, é necessário realizar o redimensionamento de cada imagem I para o tamanho de entrada fixo da rede neural convolucional, tal que $[L \times A \times 3]$, onde L é a largura e A a altura da arquitetura da CNN. Por exemplo, a arquitetura VGGNet possui um volume de entrada fixado em $[224 \times 224 \times 3]$. Após a imagem ser redimensionada, outro passo realizado é a subtração da cor média em todos os pixels de cada canal RGB da imagem I. A subtração da cor média normaliza as imagens, retirando a influência das diferentes condições de iluminação no aprendizado da CNN. Essa etapa é importante pois podemos ter imagens mais iluminadas que outras de uma

mesma classe, porém, a CNN precisa aprender as características da classe independente das condições de captura. Essa operação é dada por:

$$\begin{aligned} I(x,y,R) &= I(x,y,R) - \mu_R \\ I(x,y,G) &= I(x,y,G) - \mu_G \\ I(x,y,B) &= I(x,y,B) - \mu_B \end{aligned} \tag{4.4}$$

onde $\mu_{\{.\}}$ é a cor média de todas as imagens usadas durante a etapa de treinamento da rede neural convolucional. A imagem pré-processada (Figura 4.6 (1)) é então executada pela primeira parte da rede neural convolucional (Figura 4.6 (2)), até obtermos o mapa de características MP_c da última camada convolucional da CNN.

Com o mapa de características MP_c e o conjunto de regiões de interesse ROI obtidas pela etapa de detecção de regiões (descrito na Seção 4.1), nós extraímos as características do mapa que correspondem a cada região roi_i (Figura 4.6 (4)). Porém, como as $ROIs$ foram detectadas na imagem de entrada I, e o mapa de características possui dimensão menor que a dimensão de I (e.g. a VGGNet possui imagem de entrada com dimensão de 224×244 pixels e mapa de características de 7×7 pixels), primeiro precisamos fazer a projeção da região identificada sobre o mapa de características (Figura 4.6 (3)).

Para isso, usamos as dimensões x_{roi_i}, y_{roi_i}, l_{roi_i} e a_{roi_i} de cada região roi_i para fazermos a projeção dessa região sobre o mapa de características MP_c. Assim, obtemos uma região projetada $roi_{proj} = \{x_{proj}, y_{proj}, l_{proj}, a_{proj}\}$, conforme é mostrado na Figura 4.7.

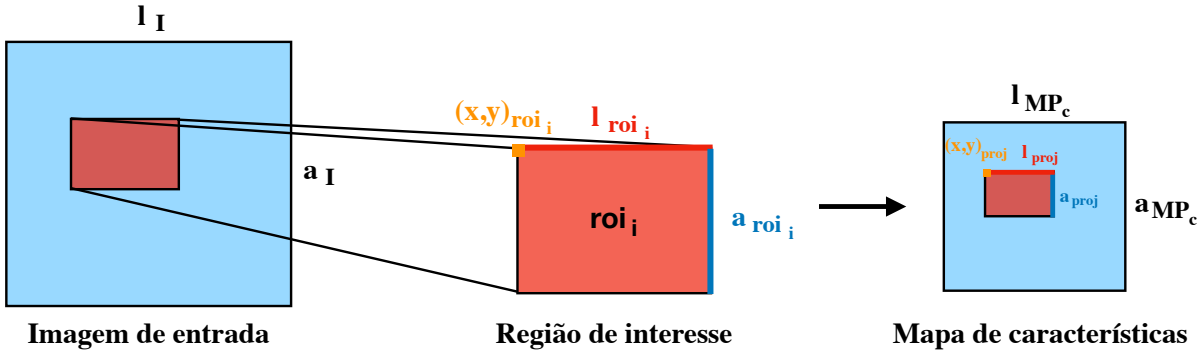
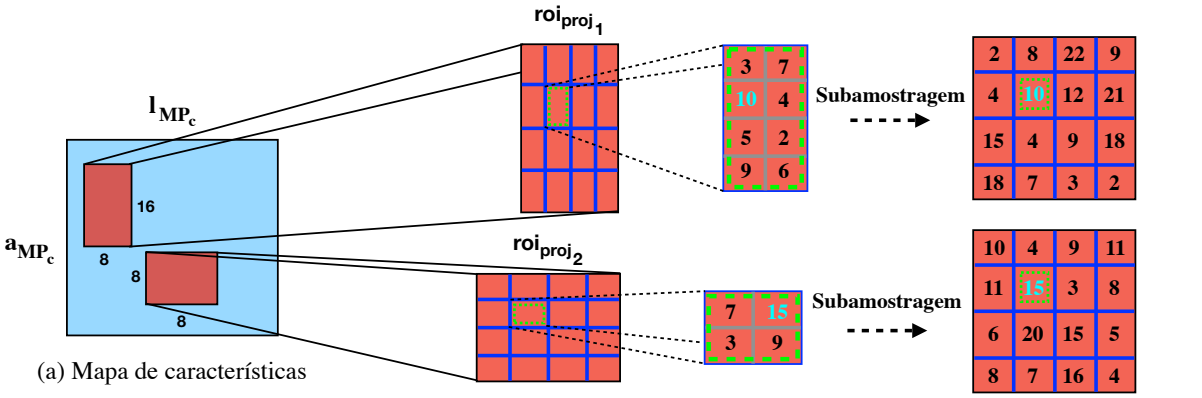


Figura 4.7: Exemplo da projeção de uma região de interesse identificada na imagem de entrada sobre um mapa de características.

Em seguida, as características extraídas são redimensionadas (do passo (4) para o passo (5)) para um tamanho fixo $w = [L_c \times A_c]$, onde L_c e A_c são hiperparâmetros da camada da CNN usada, e definem a largura e altura de entrada da camada totalmente conectada. Assim, a região projetada de tamanho $[l_{proj} \times a_{proj}]$ é dividida em uma matriz de tamanho w, onde cada célula terá

tamanho aproximado de $[\frac{l_{proj}}{L_c} \times \frac{a_{proj}}{A_c}]$. Por fim, uma subamostragem máxima é realizada em cada célula da matriz para obter as características redimensionadas com tamanho final w.

A Figura 4.8 mostra a etapa de redimensionamento das regiões para duas regiões projetadas no mapa de características, onde roi_{proj_1} possui dimensão $[8 \times 16]$ e roi_{proj_2} possui dimensão $[8 \times 8]$. Na Figura 4.8 (a) temos o mapa de características com as regiões projetadas. Na Figura 4.8 (b) realizamos a divisão das regiões por uma matriz $w = [4 \times 4]$, representada pela divisão em azul. Assim, obtemos em cada célula da divisão o tamanho $[2 \times 4]$ para a roi_{proj_1} e $[2 \times 2]$ para a roi_{proj_2}. Por fim, para cada célula realizamos uma subamostragem máxima, representado pela operação sobre a área tracejada em verde, e obtemos as regiões redimensionadas (Figura 4.8 (c)).



(a) Mapa de características

(b) Regiões de interesse

(c) Regiões redimensionadas

Figura 4.8: Exemplo do redimensionamento das características extraídas do mapa de características. Na figura (a) temos o mapa de característica com as regiões projetadas, na figura (b) temos a divisão das regiões $roi_{proj_1} = [8 \times 16]$ e $roi_{proj_2} = [8 \times 8]$ por $w = [4 \times 4]$, e na figura (c) temos o resultado da aplicação da subamostragem em cada célula da divisão.

Como a abordagem usa redes neurais convolucionais pré-treinadas para a execução, as camadas totalmente conectadas foram ajustadas para o número de K classes que seriam treinadas. Para isso, a última camada totalmente conectada e o seu classificador (e.g. *softmax* para a VGGNet) foram substituídas por uma camada totalmente conectada e um classificador *softmax* de $K + 1$ classes, além de um regressor linear baseado em limite de caixa. Assim, as características redimensionadas são usadas como entrada para as camadas totalmente conectadas da CNN (Figura 4.6 (6)), e terminamos a execução do algoritmo fazendo a classificação das regiões (Figura 4.6 (7)).

CAPÍTULO

5

Experimentos e Resultados

Neste capítulo são descritos os experimentos realizados para a avaliação da proposta, os conjuntos de imagens e métricas de avaliação usadas, e os resultados obtidos. Na Seção 5.1 são descritos os conjuntos de imagens para o treinamento e teste da CNN, na Seção 5.2, são apresentadas as métricas de avaliação empregadas durante os experimentos, na Seção 5.3 são descritas as configurações adotadas durante os experimentos e os resultados obtidos sobre o conjunto de imagens de animais silvestres do Pantanal.

5.1 Conjunto de Imagens

5.1.1 Conjunto de Treinamento

O conjunto de imagens de treinamento foi obtido da IMAGEnet [Russakovsky et al., 2015], que é uma base formada por imagens da internet. Ela contém mais de 100.000 classes com cerca de 1.000 imagens para cada classe. Por ser composto de imagens da internet não existe um controle sobre o dispositivo de captura, bem como a resolução das imagens dessa base.

Devido as diferentes quantidades de imagens por classe nessa base, realizamos a seleção aleatória de 600 imagens para cada classe, formando um conjunto de treinamento com 4.800 imagens e 8 classes. Essas classes são formadas por 8 espécies de animais do Pantanal: anta-brasileira (*Tapirus terrestris*), arara-canindé (*Ara ararauna*), capivara (*Hydrochoerus hydrochaeris*), cateto (*Pecari tajacu*), onça-parda (*Puma concolor*), papagaio-verdadeiro (*Amazona aestiva*), quati-de-cauda-anelada (*Nasua nasua*) e tamanduá-bandeira (*Myrmecophaga tridactyla*). A Figura 5.1 mostra exemplos dessa base de ima-

gens para cada classe.

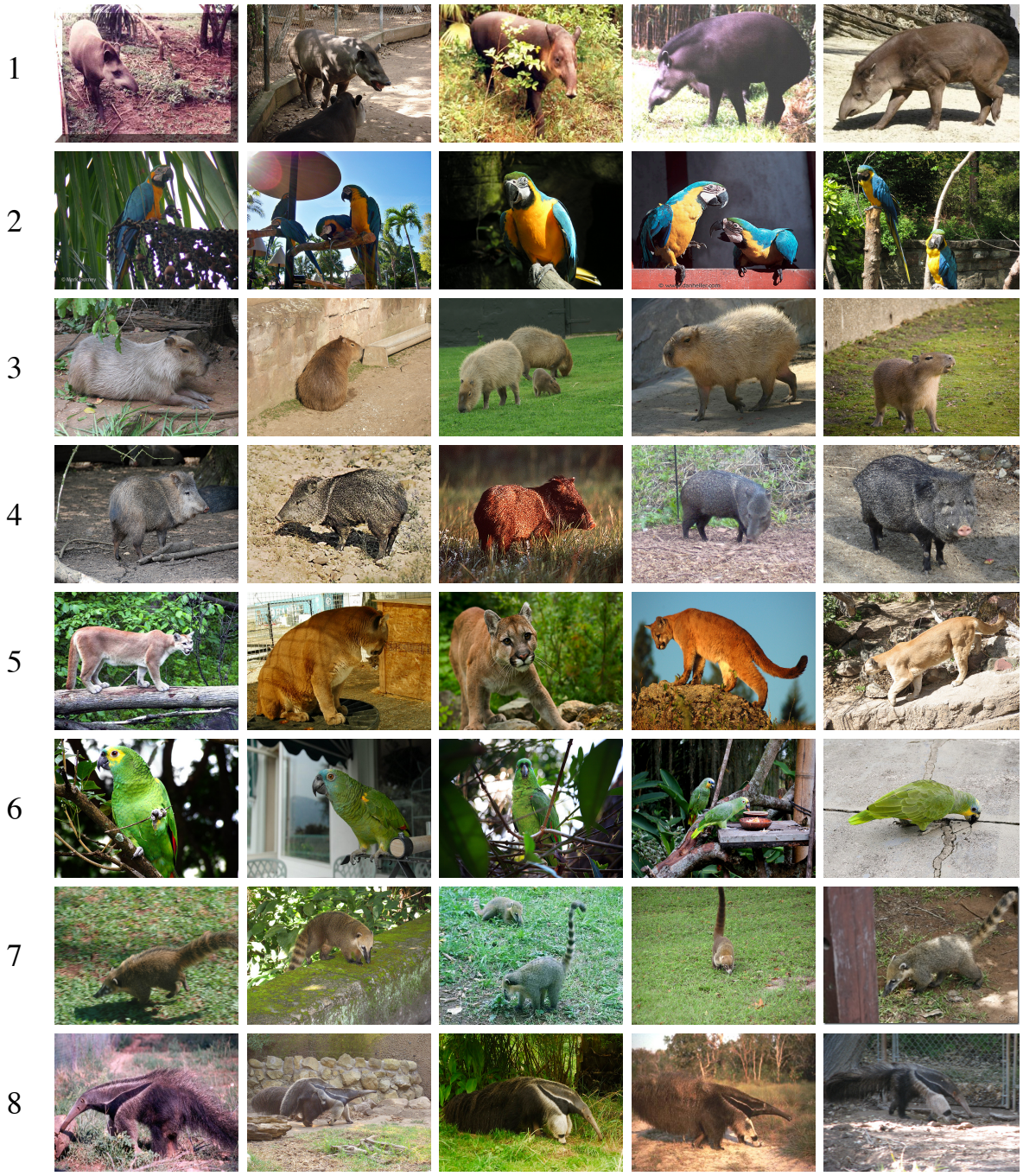


Figura 5.1: Exemplo do conjunto de imagens da IMAGEnet com as 8 espécies de animais do Pantanal. Cada linha da tabela representa uma classe desse conjunto. Linha 1 (Anta-brasileira), Linha 2 (Arara-canindé), Linha 3 (Capivara), Linha 4 (Cateto), Linha 5 (Onça-parda), Linha 6 (Papagaio-verdadeiro), Linha 7 (Quati-de-cauda-anelada) e Linha 8 (Tamanduá-bandeira)

5.1.2 Conjunto de Teste

O conjunto de imagens de teste é composto por imagens térmicas e RGB com cenas de dimensões 640×480 pixels. As imagens foram capturadas com a

câmera térmica FLIR SC640. Os parâmetros da câmera térmica foram ajustados para os valores-padrão de $0,98$ de emissividade, 60% de umidade relativa e $25°\mathrm{C}$ de temperatura atmosférica. As imagens foram capturadas de 1 a 5 metros de distância em ângulo reto.

Considerando que uma rede neural convolucional precisa de um número muito grande de imagens para o treinamento [Russakovsky et al., 2015], e que ainda não existe uma base consolidada de imagens térmicas para espécies animais do Pantanal, nós usamos as imagens térmicas apenas para a etapa de teste da rede neural convolucional. Vale ressaltar que esse procedimento não beneficia o desempenho de nenhuma das abordagens, uma vez que elas usam o mesmo conjunto de treinamento e teste. Além disso, a abordagem proposta utiliza as imagens térmicas apenas na etapa de teste, para a proposição das regiões de interesse.

As 1600 imagens térmicas e RGB estão divididas nas mesmas 8 classes de espécies de animais do Pantanal do conjunto de treinamento. As imagens foram capturadas no centro de reabilitação de animais silvestres (CRAS) na cidade de Campo Grande localizada no estado de Mato Grosso do Sul, mantido pelo Instituto de Meio Ambiente de Mato Grosso do Sul (Imasul). A Figura 5.2 mostra exemplos da base térmica e RGB para cada classe dos animais estudados.

5.2 Métricas de avaliação

Para avaliar a nossa abordagem, este trabalho empregou as métricas descritas em Girshick [2015] que são muito usadas para a avaliação de redes neurais convolucionais baseadas em regiões de interesse. Basicamente, as regiões obtidas pelas abordagens analisadas foram comparadas com regiões de referência, anotadas por especialistas, chamadas de *ground-truth*.

Para fazermos a comparação das regiões e aplicar as métricas, precisamos definir quando uma região de interesse detectada pode ser considerada verdadeira, ao ser comparada com a região anotada. Para isso, normalmente é usado o valor da intersecção sobre a união das áreas das regiões, onde dado uma região de interesse r_i e uma região de referência g_j, temos que:

$$IoU(r_i, g_j) = \frac{I(r_i, g_j)}{U(r_i, g_j)} \forall j \tag{5.1}$$

onde, $I(r_i, g_j)$ representa o valor da área de intersecção da região r_i com o *ground-truth* g_j e $U(r_i, g_j)$ representa o valor da união das áreas das duas regiões. A Figura 5.3 ilustra as áreas consideradas paras os cálculos da IoU.

Figura 5.2: Exemplo do conjunto de imagens térmicas e RGB das 8 espécies de animais do Pantanal analisadas. Cada linha da tabela mostra uma das espécies do banco, e nas três primeiras colunas temos as imagens térmicas de cada classe, e nas três colunas restantes temos as respectivas imagens RGB.

Assim, conseguimos verificar o quão próximo a região de interesse detectada está da região anotada através dessa métrica, como é mostrado na Figura 5.4. Podemos identificar através desses três exemplos, como o resultado do IoU apresenta a precisão da identificação de uma região r_i com um *ground-truth* g_j. No primeiro exemplo, temos uma IoU de 0.39, e considerando que o valor minimo usado normalmente é de 0.5, essa região não seria um região válida. No segundo e terceiro exemplo, temos uma IoU de 0.72 e 0.99 respectivamente, que mostra que essas regiões seriam consideradas válidas, mas o terceiro exemplo teria uma precisão maior.

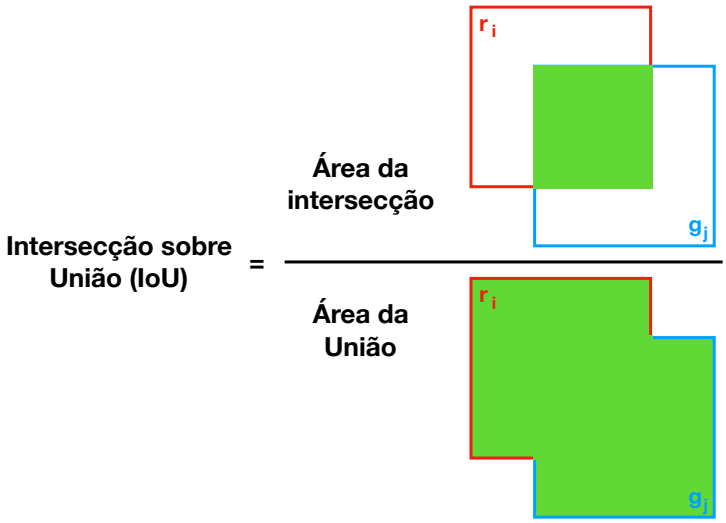


Figura 5.3: Ilustração das áreas da intersecção sobre união de duas regiões r_i e g_j.

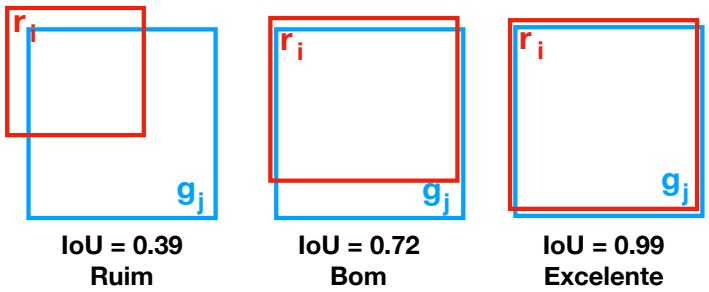


Figura 5.4: Diferentes resultados que podemos obter com a IoU, mostrando o quão duas regiões r_i e g_j estão próximas.

Porém, ainda precisamos avaliar se a classe predita na região r_i é igual a classe definida por g_j, pois podemos ter vários *ground-truth* de diferentes classes em uma imagem. Assim, se o valor de $IoU(r_i, g_j) \geq 0.5$ para algum j, e as classes de r_i e g_j forem iguais, então a região r_i é considerada um verdadeiro positivo (vp) para a classe de g_j, caso contrário, ela é considerada falso positivo (fp) para a classe de g_j.

A Figura 5.5 mostra um exemplo para uma região r_i e dois *ground-truth* g_j e g_{j-1} que são de diferentes classes, X e Y. Neste exemplo, a região r_i foi considerada um verdadeiro positivo (vp) para a classe X, pois teve um $IoU(r_i, g_{j-1}) = 0.71 \geq 0.5$ e a região r_i e g_{j-1} possuem a mesma classe. Podemos observar ainda que a região r_i teve uma maior IoU com a região g_j, porém como elas não são da mesma classes, o *ground-truth* g_j não foi considerado.

Uma vez definidos os rótulos de cada região de interesse para cada classe c, podemos calcular quatro métricas de avaliação para medir o desempenho da CNN na classe c: a precisão (PR_c), a revocação (RE_c), medida-F (MF_c), e a precisão média (PM_c). Tais medidas são formalizadas nas Equações 5.2 e 5.3.

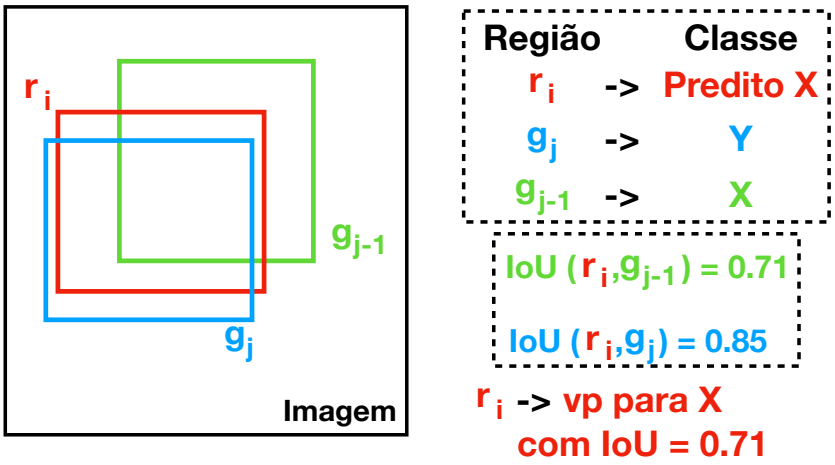


Figura 5.5: Exemplo de utilização do IoU para a verificação da precisão de uma região predita comparada com dois *ground-truth*.

$$PR_c = \frac{vp}{vp + fp} \quad \text{e} \quad RE_c = \frac{vp}{p} = \frac{vp}{vp + fn} \tag{5.2}$$

$$PM_c = \frac{PR_c}{RE_c} \quad \text{e} \quad MF_c = 2 \times \frac{PR_c \times RE_c}{PR_c + RE_c} \tag{5.3}$$

A Figura 5.6 mostra o gráfico gerado pela precisão média e suas principais características. No eixo-x temos os valores da revocação (RE), no eixo-y temos os valores da precisão (PR). A área sob a curva corresponde ao valor obtido pelo cálculo da precisão média. Assim, se tivermos uma precisão média de $0,70$, teremos 70% do gráfico coberto pela linha. Podemos observar que para obtermos uma precisão média de 100%, a linha deve atingir o canto superior direito, fazendo com que todo o gráfico seja coberto.

Por fim, uma vez que temos um total de C classes, e desejamos saber o desempenho geral da CNN sobre todas as classes, podemos calcular a micro-média da medida-F (MMF) e a média da precisão média[1] (MPM), formalizadas pela Equação 5.4.

$$MPM = \sum_{c \in C} \frac{PM_c}{C} \quad \text{e} \quad MMF = \sum_{c \in C} \frac{MF_c}{C} \tag{5.4}$$

5.3 Experimentos

Para a execução dos experimentos nós usamos o algoritmo *Fast R-CNN* seguindo a arquitetura adotada por Girshick [2015]. Além disso, nós usamos duas arquiteturas da rede neural convolucional VGGNet, descrita na Seção 3.4.2, uma com 16 camadas e outra com 19 camadas.

[1]Também conhecida com Mean Average Precision (mAP).

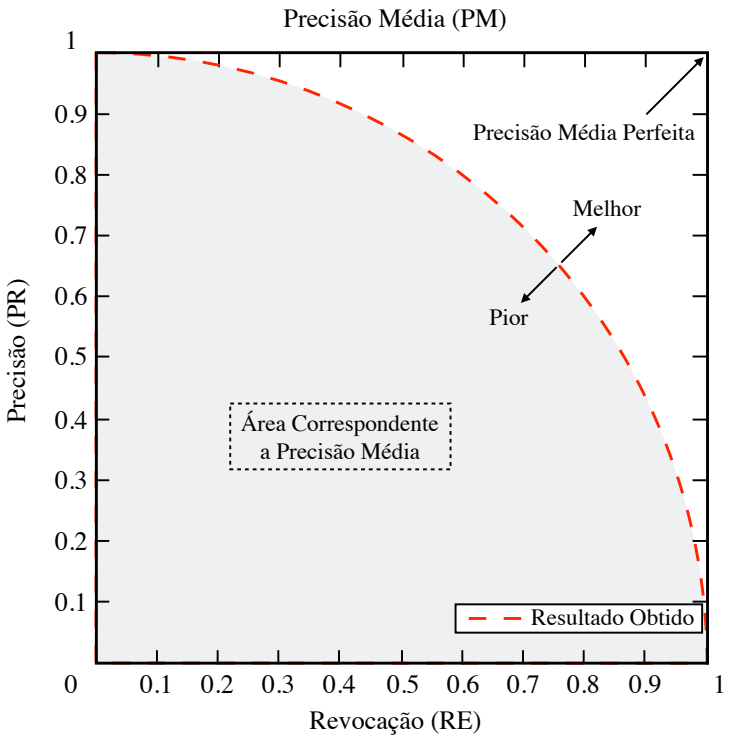


Figura 5.6: Exemplo do gráfico gerado pela precisão média, no eixo x temos a revocação e no eixo y temos a precisão. A área sob a curva corresponde ao valor da precisão média.

As redes neurais convolucionais foram treinadas com o conjunto de treinamento definido na Seção 5.1.1, e eles foram realizados sobre conjuntos de duas imagens por vez. Seguindo os padrões usados na matconvnet [Vedaldi and Lenc, 2015], nós usamos 75 épocas para o treinamento da CNN, onde uma época corresponde a uma passagem por todas as imagens da base. A taxa de aprendizado usada foi de $\frac{1e-3}{64}$ nas 50 primeiras épocas e de $0.1 * \frac{1e-3}{64}$ nas 25 épocas restantes.

Para a etapa de teste, nós executamos duas variações da abordagem proposta e duas implementações da abordagem *Fast R-CNN* [Girshick, 2015] sobre o conjunto de teste descrito na Seção 5.1.2. A abordagem proposta foi testada com um limiar fixo, onde usamos valor de incremento $Inc = 1$, e com um limiar variável, onde usamos a distância entre os filtros $dis_f = 0.2$. Nos dois casos, para a técnica SLIC nós usamos os valores de parâmetro de compactação $m = 5$ e número de superpixels $k = 1500$, seguindo os valores adotados por Arruda et al. [2015], e para a eliminação de regiões ruidosas usamos o valor de $P_{min} = 4$.

A primeira variação da abordagem *Fast R-CNN* é a implementação original adotada por Girshick [2015], que aplica o algoritmo de proposição de regiões *Selective Search* sobre a imagem RGB. Já na segunda implementação, nós aplicamos o *Selective Search* sobre a imagem RGB e Térmica para obter uma comparação mais justa, uma vez que nossa abordagem utiliza as imagens

térmicas além da RGB. Nos dois casos foram adotados os parâmetros de execução padrões definidos por Girshick [2015].

Por fim, foram utilizados o teste estatístico de análise de variança ANOVA e o pós-teste de Tukey para validar os resultados dos experimentos. Além disso, para de facilitar a analise dos resultados foi utilizado a estatística descritiva diagrama de caixas e bigode [Ross, 2014] que apresenta o valor central de desempenho obtido por uma abordagem e como os demais valores variam com relação a ele. Para isso, o diagrama possui três itens principais: a linha, a caixa e os bigodes. A linha é mostrada em destaque dentro da caixa e representa o valor mediano dos resultados, os limites da caixa representam a faixa de valores que contêm 50% do resultados obtidos. Por fim, os bigodes representam a faixa de valores onde a maioria dos resultados estão, e caso algum resultado ultrapasse a faixa de valores definidas pelos bigodes, eles são considerados como ruídos e são representados por pequenos círculos fora do bigode. A Figura 5.7 apresenta um exemplo da estrutura do diagrama de caixas e bigode.

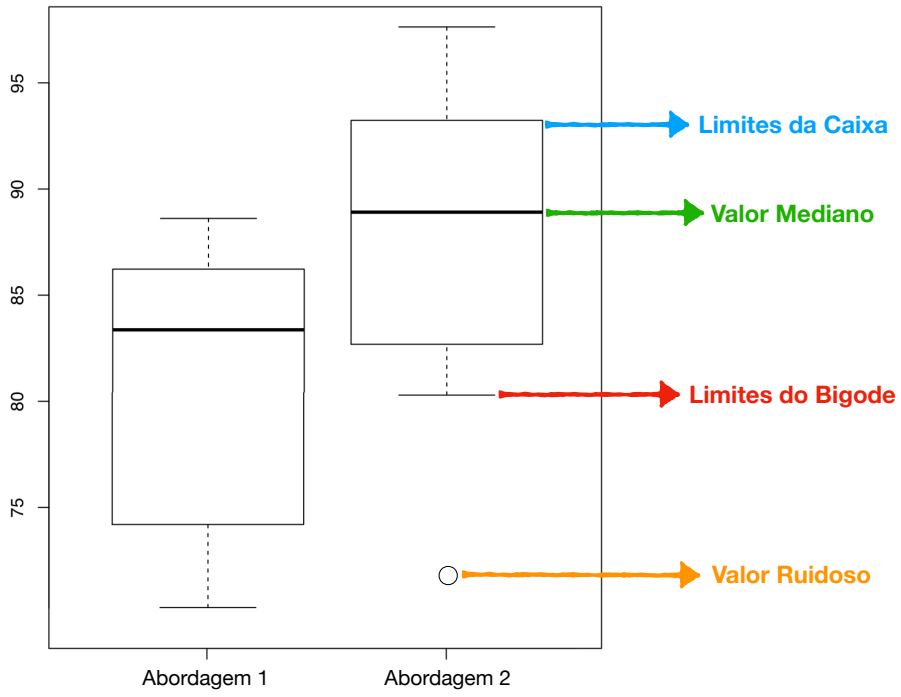


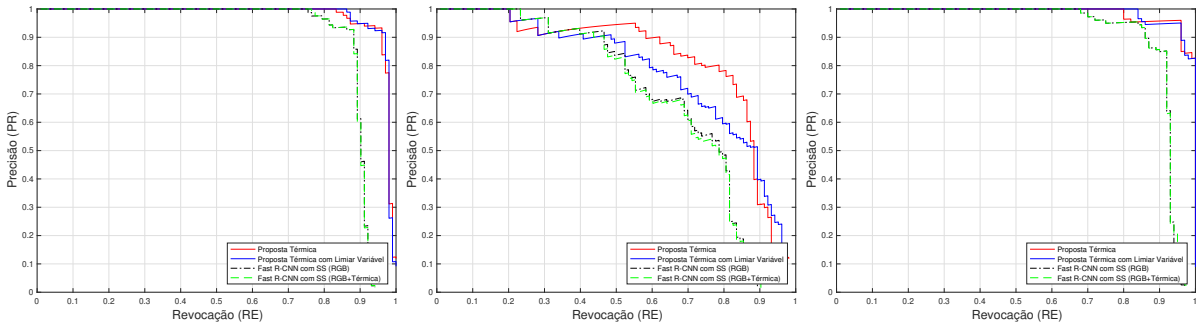
Figura 5.7: Exemplo da estrutura dos diagramas de caixas e bigode.

5.3.1 Resultados dos Experimentos

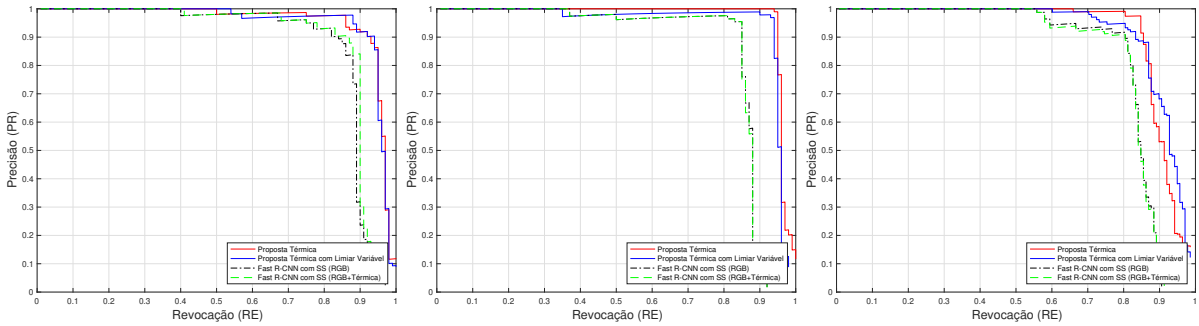
As variações da abordagem proposta por esse trabalho foram comparadas com as duas implementações da abordagem *Fast R-CNN* [Girshick, 2015]. Os experimentos foram realizados sobre o conjunto de 1600 imagens térmicas e

RGB de espécies de animais do Pantanal (descrito na Seção 5.1.2). As abordagens foram avaliadas quantitativamente pelas métricas precisão média (PM) e medida-F (MF) (descrita na Seção 5.2).

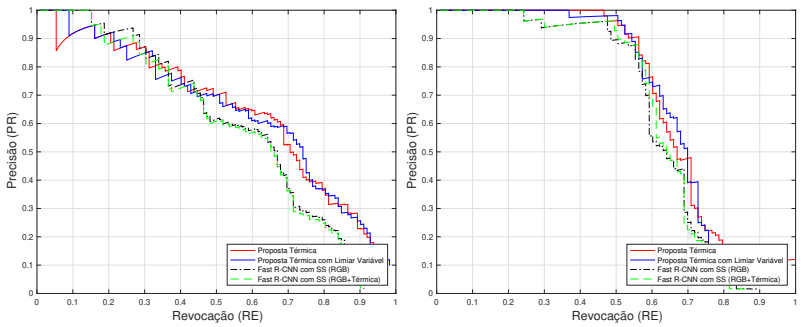
As Figuras 5.8 e 5.9 mostram os gráficos da precisão média das quatro abordagens, para as oito classes de animais, usando duas arquiteturas da rede neural convolucional VGGNet, uma com 16 e outra com 19 camadas. Para todos os gráficos, no eixo-x é mostrado os valores obtidos pela revocação (RE), e no eixo-y é mostrado os valores da precisão (PR), ambos variando de 0 até 1. Assim, o valor da precisão média é dada pela área sob a curva, e quanto maior é essa área, maior o valor da precisão média.



(a) Anta (b) Arara-canindé (c) Capivara

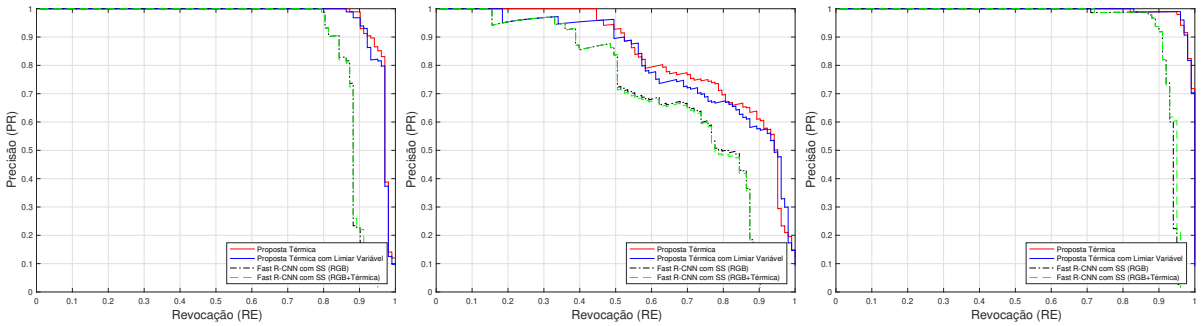


(d) Cateto (e) Onça-parda (f) Papagaio-verdadeiro

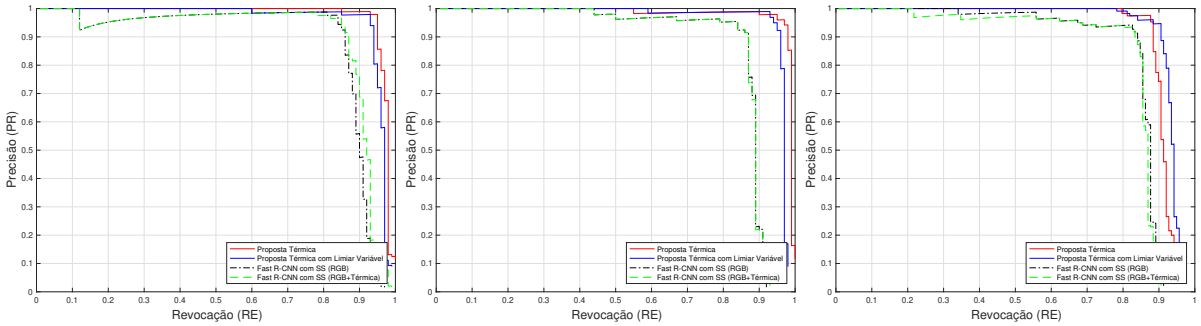


(g) Quati (h) Tamanduá-bandeira

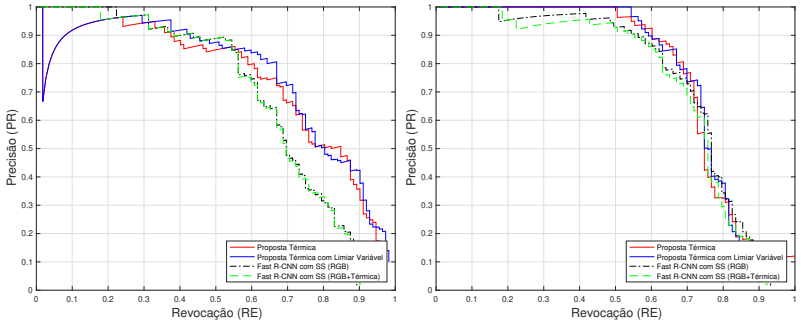
Figura 5.8: Resultados da precisão média para as oito classes analisadas com a arquitetura VGGNet de 16 camadas.



(a) Anta (b) Arara-canindé (c) Capivara



(d) Cateto (e) Onça-parda (f) Papagaio-verdadeiro



(g) Quati (h) Tamanduá-bandeira

Figura 5.9: Resultados da precisão média para as oito classes analisadas com a arquitetura VGGNet de 19 camadas.

As Tabelas 5.1 e 5.2 mostram os valores da medida-F e da precisão média para as 8 classes de animais e duas arquitetura testadas. Além disso, a última linha das tabelas apresentam a média geral das duas medidas sobre todas as classes, ou seja, são descritas a média da precisão média (MPM) e a micro-média da medida-F (MMF).

5.3.2 Testes Estatísticos

Realizamos uma avaliação estatística dos resultados usando o teste de análise de variança ANOVA [Shapiro and Wilk, 1965, Iversen et al., 1987], e o pós-

Espécies do Pantanal	**Precisão média (PM) (%)**				**Medida-F (MF) (%)**			
	Fast R-CNN		Proposta		Fast R-CNN		Proposta	
	SS (Default)	SS com Térmica	Limiar Fixo	Limiar Variável	SS (Default)	SS com Térmica	Limiar Fixo	Limiar Variável
Anta	87,08	86,97	**91,58**	91,43	87,96	87,96	**93,60**	93,14
Arara-canindé	68,38	68,38	**78,89**	74,66	48,57	48,57	53,79	**56,38**
Capivara	88,95	88,95	**98,06**	97,97	78,30	78,30	**80,65**	77,82
Cateto	83,48	88,24	**90,88**	90,42	85,08	85,08	90,53	**91,67**
Onça-parda	82,16	82,09	**92,27**	90,31	89,47	89,47	**96,94**	94,00
Papagaio-verdadeiro	80,97	80,66	86,49	**87,47**	78,15	77,82	**83,68**	83,27
Quati	59,45	58,03	**63,72**	63,66	54,08	53,81	59,90	**60,91**
Tamanduá-bandeira	61,16	62,00	**69,30**	66,66	62,89	64,60	64,52	**66,67**
Média geral	76,45	76,91	**83,89**	**82,82**	73,06	73,20	**77,95**	**77,98**

Tabela 5.1: Resultados experimentais para a abordagem proposta e para e abordagem *Fast R-CNN* na arquitetura VGGNet com 16 camadas.

Espécies do Pantanal	**Precisão média (PM) (%)**				**Medida-F (MF) (%)**			
	Fast R-CNN		Proposta		Fast R-CNN		Proposta	
	SS (Default)	SS com Térmica	Limiar Fixo	Limiar Variável	SS (Default)	SS com Térmica	Limiar Fixo	Limiar Variável
Anta	83,77	83,78	**91,91**	91,52	88,17	88,17	92,63	**93,26**
Arara-canindé	68,54	68,28	**81,45**	79,15	61,18	61,18	**66,67**	**66,67**
Capivara	90,24	90,24	**97,35**	97,22	**81,94**	81,58	81,30	80,00
Cateto	86,12	87,90	**91,95**	90,40	90,32	90,43	**95,92**	94,90
Onça-parda	82,73	82,64	**92,00**	90,52	89,12	89,12	**95,57**	93,20
Papagaio-verdadeiro	81,44	80,39	88,42	**90,35**	79,83	79,83	86,89	**87,80**
Quati	66,51	66,98	72,36	**74,48**	66,67	66,67	67,72	**71,07**
Tamanduá-bandeira	72,31	70,47	**75,42**	74,13	64,52	64,10	67,92	**71,17**
Média geral	78,95	78,83	**86,35**	**85,97**	77,72	77,63	**81,83**	**82,26**

Tabela 5.2: Resultados experimentais para a abordagem proposta e para e abordagem *Fast R-CNN* na arquitetura VGGNet com 19 camadas.

teste de Tukey [Tukey, 1949]. O teste ANOVA verifica se existem diferenças na distribuição de uma medida entre três ou mais grupos. As medidas consideradas foram a precisão média e medida-F obtidas para as arquiteturas VGGNet de 16 e 19 camadas, onde os grupos são as diferentes abordagens analisadas.

O teste ANOVA é um teste de hipóteses, portanto precisamos definir a hipótese de estudo. O ANOVA usa duas hipóteses, a hipótese nula H_0 e a alternativa H_1, onde a primeira define que não existe diferença significativa entre os desempenhos das abordagens analisadas, e a segunda define que existe pelo menos uma abordagem com desempenho significativamente diferente. O objetivo da análise é verificar se a hipótese nula pode ser refutada para considerar que os resultados obtidos são significativos, e representam o real desempenho das abordagens.

Basicamente, o teste ANOVA encontra um valor que indica a variança entre as abordagens, chamada de razão F, e a sua distribuição de probabilidade

[de Andrade Martins, 2000]. Em seguida, é usado um nível α para definir o limite de rejeição da hipótese nula[2]. Com isso, obtemos o valor-p que nos diz a probabilidade dos resultados terem sido obtidos, assumindo que as abordagens fossem iguais (hipótese nula). Logo, quanto menor é o valor-p, mais confiança temos para descartar a hipótese nula e garantir que existe pelo menos uma abordagem com resultados significativamente diferentes das demais.

Ao realizar o teste ANOVA para os resultados das quatro abordagens na arquitetura VGGNet de 16 camadas obtemos o valor-p $3.51e^{-09}$ para a precisão média e $1.32e^{-07}$ para a medida-F, logo, podemos descartar a hipótese nula e concluir que pelo menos duas abordagens possuem desempenho significativamente diferentes para as duas medidas. Com isso, precisamos verificar a relação entre cada par de abordagens para sabermos quais pares possuem desempenhos significativamente diferentes. Para isso, aplicamos o pós-teste de Tukey que faz uma verificação múltipla com todos os pares de abordagens. A Tabela 5.3 mostra os resultados da aplicação do teste de Tukey para as quatro abordagens analisadas na arquitetura VGGNet de 16 camadas.

Pares de Abordagens		**Valor-p**	
		Precisão média	**Medida-F**
Fast R-CNN (RGB+Térmica)	*Fast R-CNN* (Default)	0,8481158	0,9979449
Proposta (Variável)	*Fast R-CNN* (Default)	0,0000008	0,0000077
Proposta (Fixo)	*Fast R-CNN* (Default)	0,0000001	0,0000084
Proposta (Variável)	*Fast R-CNN* (RGB+Térmica)	0,0000042	0,0000113
Proposta (Fixo)	*Fast R-CNN* (RGB+Térmica)	0,0000003	0,0000124
Proposta (Fixo)	Proposta (Variável)	0,5674784	0,9999719

Tabela 5.3: Resultados do pós-teste de Tukey para as quatro abordagens avaliadas na arquitetura VGGNet de 16 camadas.

Observamos que o valor-p para a comparação de qualquer implementação da nossa abordagem com uma das implementações do *Fast R-CNN* alcançaram valores abaixo de $0,0000042$ para a precisão média e abaixo de $0,0000124$ para a medida-F. Portanto, todos esses pares de abordagens podem ser considerados com desempenho significativamente diferentes, validando os resultados alcançados para a arquitetura de 16 camadas.

Podemos observar que a comparação entre as implementações do *Fast R-CNN* (linha 1) obteve um valor-p de $0,84$ para a precisão média e $0,99$ para a medida-F. Por outro lado, a comparação das variações da abordagem proposta (linha 6) alcançou um valor-p de $0,56$ para a precisão média e $0,99$ para a medida-F. Isso indica que as abordagens não possuem desempenho significativamente diferentes entre si, porém, esse resultado era esperado pois elas

[2]Normalmente é usado um valor de α igual a $0,05$ (5%).

continuam sendo a mesma abordagem, mas com modificações na proposição de regiões para o *Fast R-CNN* e no tipo de limiar para a abordagem proposta.

Por fim, realizamos o teste ANOVA para os resultados da arquitetura VGG-Net de 19 camadas, e obtivemos o valor-p $1.80e^{-09}$ para a precisão média e $6.15e^{-06}$ para a medida-F, indicando que temos no mínimo duas abordagens significativamente diferentes para as duas medidas. A Tabela 5.4 mostra os resultados do pós-teste de Tukey para as 4 abordagens nessa arquitetura. Podemos observar novamente, que obtivemos diferença significativa através do valor-p para as comparações entre as variações da abordagem proposta e das implementações do *Fast R-CNN*, validando os resultados alcançados para a arquitetura de 19 camadas.

Pares de Abordagens		**Valor-p**	
		Precisão média	**Medida-F**
Fast R-CNN (RGB+Térmica)	*Fast R-CNN* (Default)	0,9989655	0,9996494
Proposta (Variável)	*Fast R-CNN* (Default)	0,0000004	0,0001538
Proposta (Fixo)	*Fast R-CNN* (Default)	0,0000002	0,0004979
Proposta (Variável)	*Fast R-CNN* (RGB+Térmica)	0,0000003	0,0001227
Proposta (Fixo)	*Fast R-CNN* (RGB+Térmica)	0,0000001	0,0003959
Proposta (Fixo)	Proposta (Variável)	0,9699192	0,9568403

Tabela 5.4: Resultados do pós-teste de Tukey para as quatro abordagens avaliadas na arquitetura VGGNet de 19 camadas.

5.3.3 Discussão dos Resultados

Com os resultados do teste ANOVA e pós-teste de Tukey confirmando a diferença significativa entre as abordagens, podemos avaliar os resultados apresentados nas Tabelas 5.1 e 5.2. Nas duas arquiteturas temos que a precisão média e medida-F para as variações da abordagem proposta são superiores aos resultados das implementações da abordagem *Fast R-CNN* para a maioria das classes de animais estudados.

Na arquitetura de 16 camadas, a nossa abordagem melhorou em média $5,9\%$ a precisão média e $4,89\%$ a medida-F quando comparado com os resultados das implementações do *Fast R-CNN*. Por outro lado, para a arquitetura de 19 camadas, a nossa abordagem obteve $7,14\%$ de desempenho superior ao *Fast R-CNN* para a precisão média e $4,2\%$ para a medida-F. Assim, podemos notar que o uso de imagens térmicas e o SLIC para a detecção de regiões de interesse na abordagem proposta consegue melhorar, em média, $5,5\%$ o desempenho da CNN.

Podemos avaliar esses resultados de forma visual através do diagrama de caixas e bigode apresentados nas Figuras 5.10 e 5.11. Os diagramas mostram

os desempenhos das quatro abordagens com a arquitetura VGGNet de 16 e 19 camadas. Notamos pela mediana e os limites superiores das caixas e dos bigodes das abordagens propostas que elas apresentaram desempenho superior as implementações do *Fast R-CNN* nas duas arquiteturas.

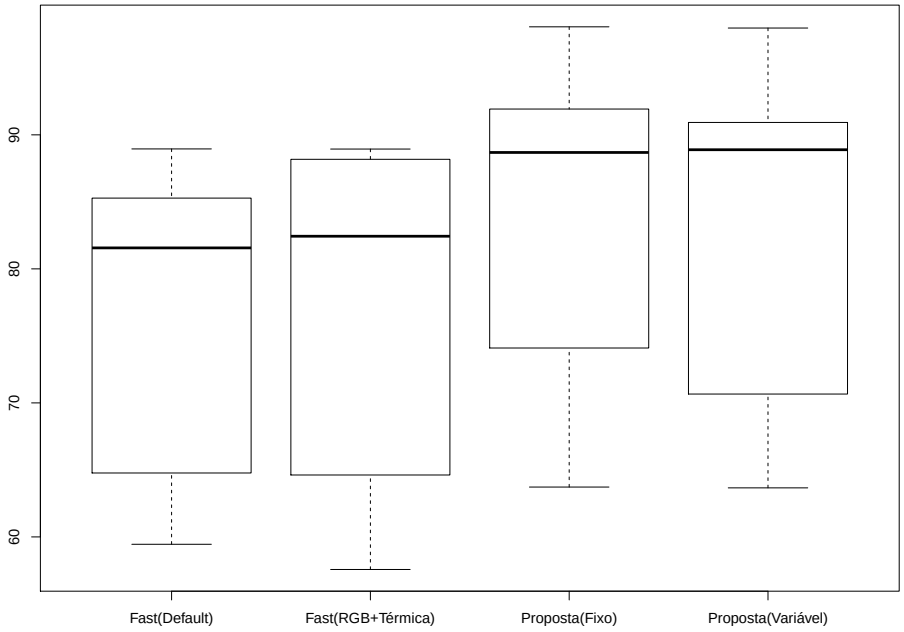


Figura 5.10: Diagrama de caixas e bigodes para as quatro abordagens avaliadas na arquitetura VGGNet de 16 camadas.

Por outro lado, notamos através dos resultados apresentados nas Tabelas 5.1 e 5.2 e nos diagramas das Figuras 5.10 e 5.11, que a implementação do *Fast R-CNN* com a aplicação do *Selective Search* nas imagens térmicas e RGB, em comparação com a sua implementação original não apresentou uma diferença significativa nos resultados, apesar de melhorá-los em alguns casos. Observamos que isso aconteceu pois a implementação original já cria aproximadamente duas mil regiões propostas com a imagem RGB, e o *Selective Search* sobre a imagem térmica não trouxe regiões significativamente diferentes.

Da mesma forma, as variações da nossa abordagem tiveram desempenhos semelhantes para as duas métricas analisadas. Notamos que essa semelhança ocorreu principalmente pelo uso de $Inc = 1$ que gerou um limiar fixo que conseguiu delimitar corretamente os superpixels que pertencem aos animais para toda a base de imagens, se aproximando dos resultados obtidos pelo limiar variável que se adapta as condições de temperatura dos superpixels de cada imagem da base.

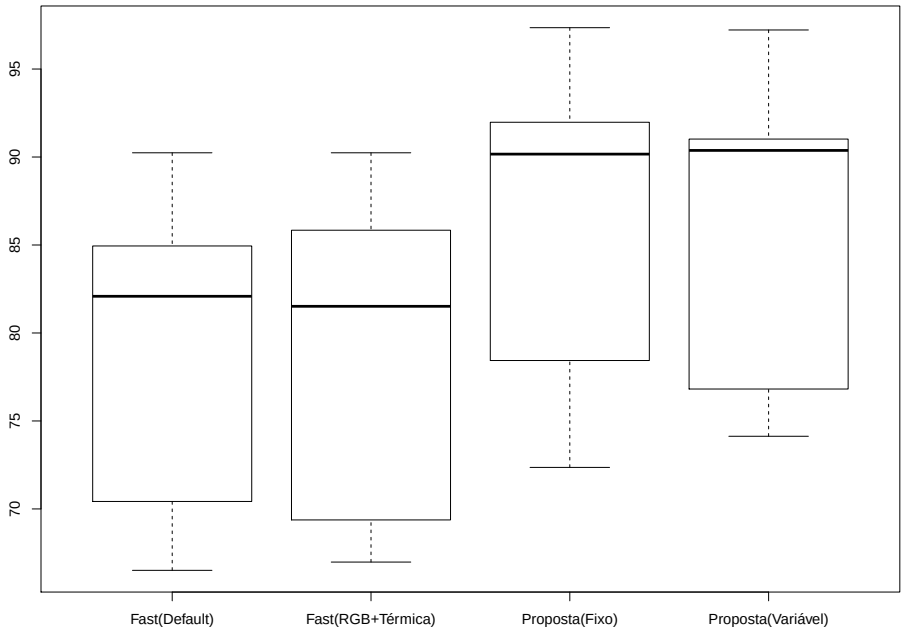


Figura 5.11: Diagrama de caixas e bigodes para as quatro abordagens avaliadas na arquitetura VGGNet de 19 camadas.

Além disso, analisando as Tabelas 5.1 e 5.2, podemos observar que os resultados de todas as abordagens na arquitetura com 19 camadas possuem desempenho melhor que na arquitetura com 16 camadas. A média dessa melhoria foi de $2,5\%$ para a precisão média e $4,31\%$ para a medida-F. Assim, observamos que a inserção de mais camadas na CNN melhoraram em média $3,5\%$ o desempenho das abordagens.

Por fim, podemos avaliar os desempenhos das abordagens em relação a cada classe, identificando as classes mais desafiadoras para cada abordagem. Para isso, geramos o diagrama de caixas e bigode sobre os resultados das 8 classes analisadas para as duas medidas. A Figura 5.12 mostra o diagrama de caixas para as 8 classes analisadas sobre os resultados das precisão média para as duas arquiteturas testadas.

Podemos observar que a classe com melhor desempenho foi a capivara com valores variando de $88,95\%$ a $98,06\%$. Para essa classe a nossa abordagem obteve melhor desempenho que as implementações do *Fast R-CNN*, alcançando uma melhora de 9% para a VGGNet com 16 camadas, e 7% para a VGGNet de 19 camadas. Por outro lado, temos que a classe com pior desempenho foi o quati com valores variando de $58,03\%$ a $74,48\%$. Contudo, mesmo para essa classe a nossa abordagem teve um desempenho melhor que as implementações do *Fast R-CNN*, alcançando uma melhora de 4% com a VGGNet de 16 camadas e $7,5\%$ com a VGGNet de 19 camadas.

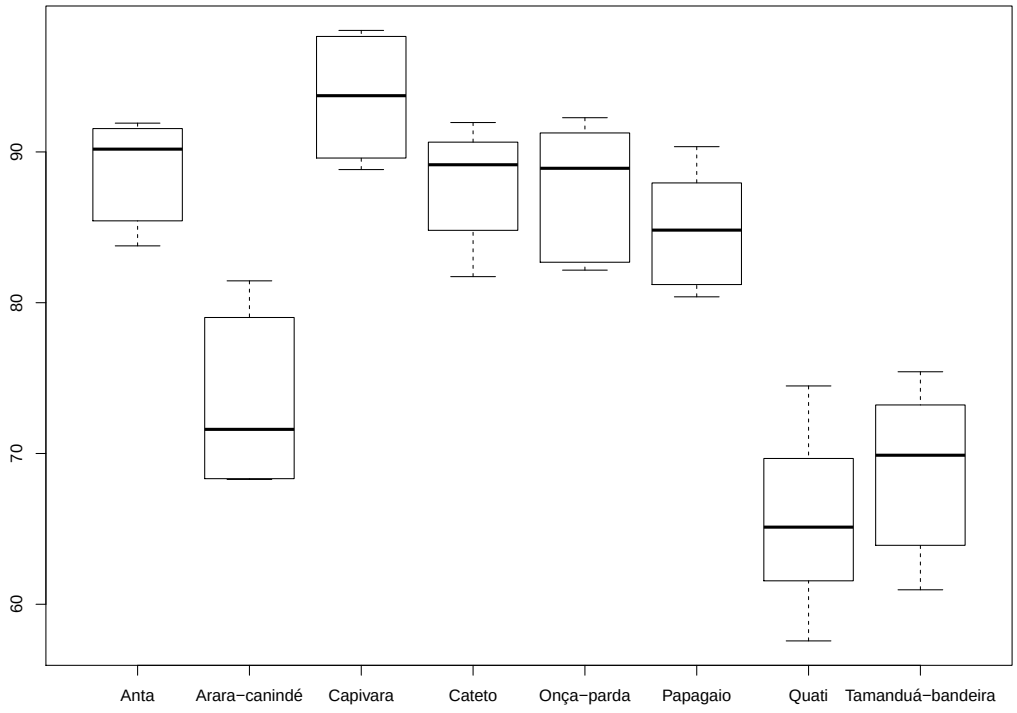


Figura 5.12: Diagrama de caixas dos resultados da precisão média para as oito classes e duas arquiteturas.

Da mesma forma podemos avaliar o diagrama de caixas das 8 classes analisadas sobre os resultados da medida-F nas duas arquiteturas através da Figura 5.13. Podemos observar que para essa medida, a classe com melhor resultado foi a onça-parda com valores de $89,12\%$ a $96,94\%$. Para essa classe, a nossa abordagem melhorou o desempenho em relação as implementações do *Fast R-CNN* em $7,4\%$ com a VGGNet de 16 camadas e $6,4\%$ com a VGGNet de 19 camadas. Por outro lado, a classe que obteve pior desempenho para essa métrica foi a arara-canindé com valores variando de $48,57\%$ a $66,67\%$. Para essa classe a nossa abordagem também obteve um melhor desempenho, superando o *Fast R-CNN* em $7,8\%$ para a VGGNet de 16 camadas e $5,4\%$ para a VGGNet de 19 camadas.

Notamos que para as duas métricas as classes que tiveram mais dificuldades para serem classificadas foram a arara-canindé, quati e tamanduá-bandeira. Observamos que isso aconteceu pois essas classes são mais desafiadoras para as duas abordagens. Para o *Fast R-CNN* que considera as características de cor e textura, temos que esses animais possuem um padrão de pelagem mais complexo ou mais semelhante ao ambiente, tornando mais difícil a proposição das regiões. Por outro lado, verificamos que eles são mais difíceis de serem delimitados pela nossa abordagem, pois possuem longas ex-

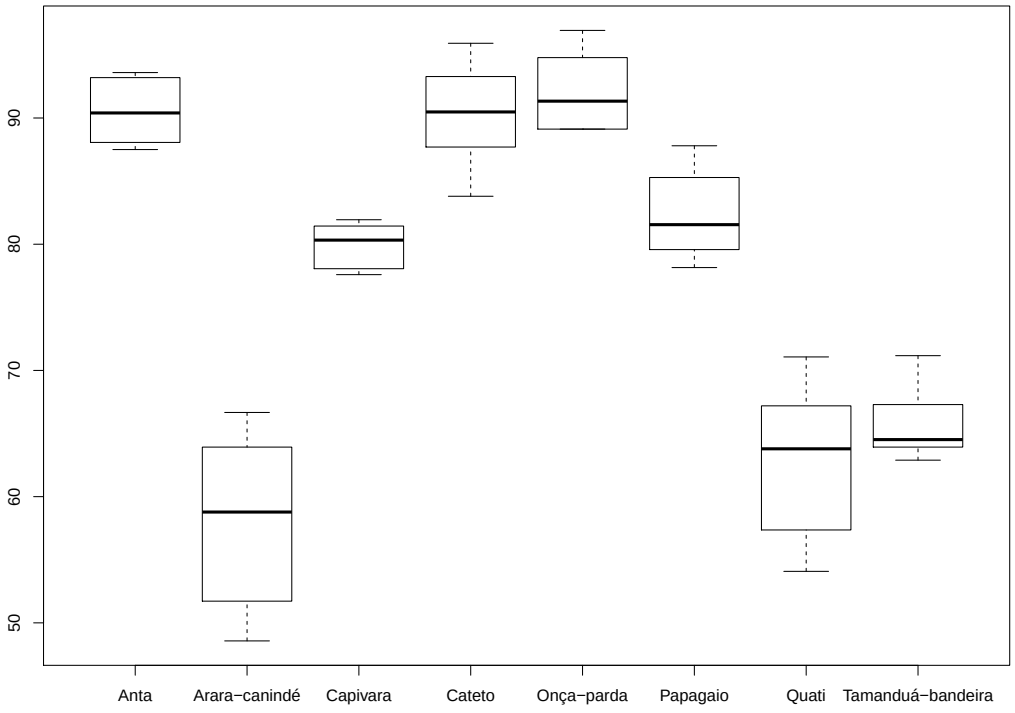
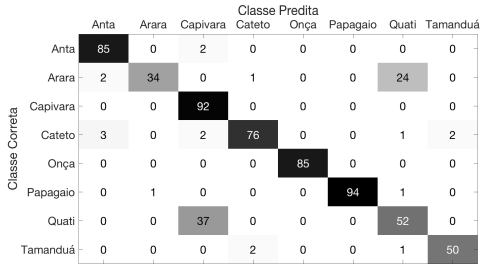


Figura 5.13: Diagrama de caixas dos resultados da medida-F para as oito classes e duas arquiteturas.

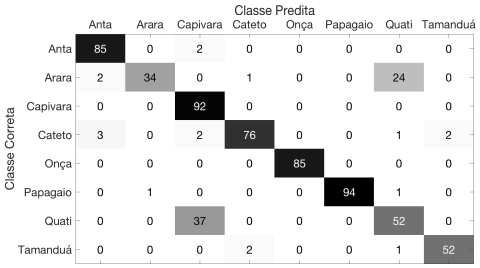
tremidades, como as caudas e asas, que são bem mais frias que o animal, chegando a temperatura ambiente no caso das penas da ave, o que dificulta a delimitação pela temperatura corporal do animal.

Porém, mesmo nesses casos a nossa abordagem obteve melhor resultado nas comparações com as implementações do *Fast R-CNN*, e observamos que o principal motivo é que ela usa as características de temperatura na localização do animal na imagem, não sofrendo influência das características da pelagem do animal, região de fundo e iluminação da imagem. Por isso podemos observar que os melhores resultados da abordagem proposta em comparação com o *Fast R-CNN*, ocorrem para classes de animais que possuem uma pelagem mais semelhante ao ambiente de fundo, como para a onça-parda. Por outro lado, como a abordagem *Fast R-CNN* só usa as características de cor e textura da imagem RGB, ela obtém um desempenho muito inferior em imagens desse tipo.

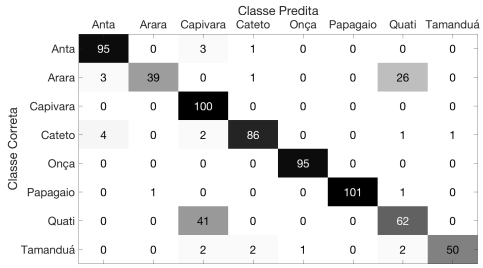
Por fim, podemos analisar o desempenho das abordagens em cada classe através da matriz de confusão que apesenta a forma que as regiões foram classificadas por cada abordagem. As Figuras 5.14 e 5.15 mostram a matriz de confusão com os resultados das classificações das regiões pelas quatro abordagens nas duas arquiteturas analisadas.



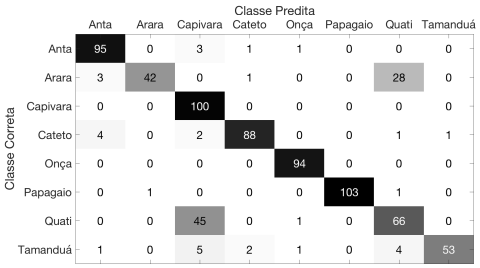
(a) Fast R-CNN - SS (Default)



(b) Fast R-CNN - SS com Térmica

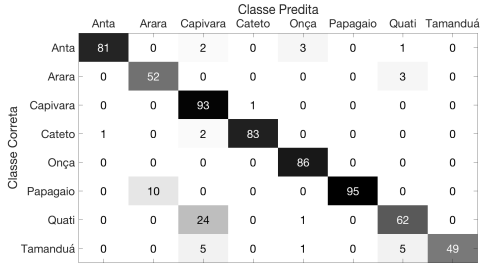


(c) Proposta - Limiar Fixo

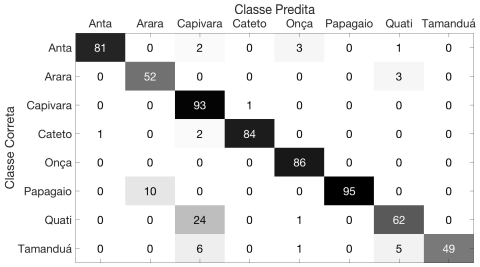


(d) Proposta - Limiar Variável

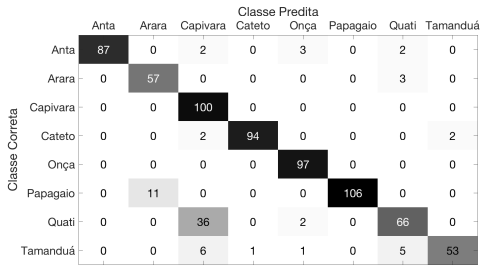
Figura 5.14: Matriz de confusão para as regiões propostas na arquitetura VGGNet de 16 camadas.



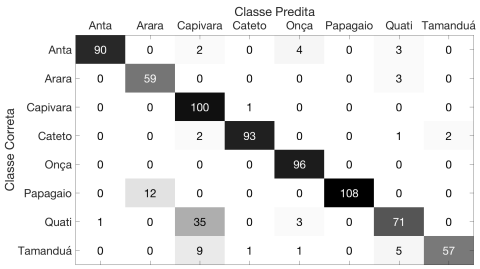
(a) Fast R-CNN - SS (Default)



(b) Fast R-CNN - SS com Térmica



(c) Proposta - Limiar Fixo



(d) Proposta - Limiar Variável

Figura 5.15: Matriz de confusão para as regiões propostas na arquitetura VGGNet de 19 camadas.

Na arquitetura VGGNet de 16 camadas, podemos observar que as classes arara-canindé e quati foram as mais desafiadoras para todas as abordagens. Na classe arara-canindé observamos que algumas regiões foram classificadas como da classe quati, e algumas regiões da classe quati foram classificadas como da classe capivara. Isso acontece principalmente pelo desafio na delimitação do animal pelas abordagens e pela similaridade de pelagem entre as classes.

Na arquitetura VGGNet de 19 camadas, podemos observar que ocorreu uma melhoria na classificação correta das regiões para todas as abordagens. Essa melhoria pode ser vista para várias classes, porém, para a classe arara-canindé podemos observar que os erros de classificação com a classe quati na arquitetura VGGNet de 16 camadas foram superados. Isso indica que ao inserir mais camadas na rede neural convolucional, ela consegue aprender mais características de cada classe e melhorar a classificação das regiões.

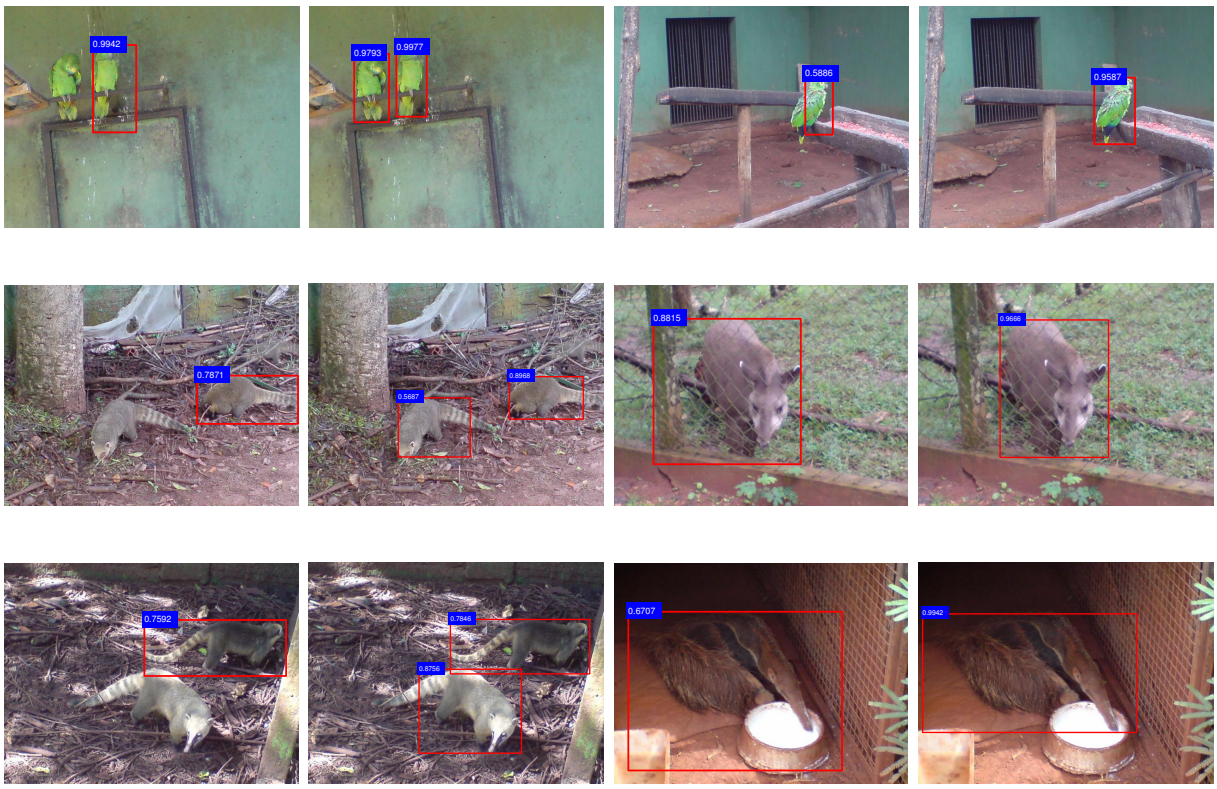
Por fim, podemos observar que para as duas arquiteturas a abordagem proposta obteve um número maior de regiões classificadas corretamente. Isso indica que a nossa abordagem conseguiu identificar corretamente mais animais nas cenas se comparado com as abordagens do *Fast R-CNN*. Porém, vale ressaltar que o número de regiões classificadas são diferentes pois cada abordagem propõem uma quantidade diferente de regiões e a classificação depende da classe predita pela rede e do IoU da região com o *ground-truth*.

5.3.4 Avaliação Qualitativa

Realizamos uma avaliação qualitativa das regiões de interesse que foram detectadas pelas abordagens. A Figura 5.16 mostra alguns exemplos da comparação visual das regiões de interesse propostas pelas abordagens sobre as mesmas imagens. Foram usados quatro classes para essa avaliação, o tamanduá-bandeira e quati que são duas das classes que tiveram piores desempenhos, e o papagaio e a anta que estão entre as com melhor desempenho.

Nas duas primeiras colunas temos três exemplos de identificação de múltiplos animais com as abordagens *Fast R-CNN* e proposta, e nas duas colunas restantes temos três exemplos para a identificação de apenas um animal na imagem. Podemos observar que nas imagens com múltiplos animais a abordagem proposta se saiu melhor, conseguindo identificar todos os animais na imagem. Já para as imagens com um animal, podemos observar que a região de interesse obtida pela nossa abordagem, consegue delimitar melhor o animal em comparação com a abordagem *Fast R-CNN*.

A semelhança das características da pelagem do animal com o seu ambiente é um fator importante no baixo desempenho da abordagem *Fast R-CNN*. Isso fica evidente nas imagens de exemplo com múltiplos animais, onde na



(a) *Fast R-CNN* (b) Proposta (c) *Fast R-CNN* (d) Proposta

Figura 5.16: Comparação visual da detecção de regiões de interesse propostas pelas abordagens analisadas. Nas primeiras duas colunas temos três exemplos de identificação de múltiplos animais na imagem pelas duas abordagens, e na duas últimas temos três exemplos de identificação de apenas um animal pelas abordagens. Os valores associados as regiões das imagens representam o grau de confiança dado pela CNN.

primeira linha temos a imagem de um papagaio que possui a coloração verde, e está em um ambiente com fundo verde. Assim como nas duas últimas linhas, que mostram dois exemplos de identificação de quati que possuem uma pelagem mais cinza e marrom próxima a coloração do chão.

Esse tipo de semelhança não é um problema para a abordagem proposta, pois ela usa as imagens térmicas para a detecção dos animais na imagem RGB. Portanto, não sofre influência da coloração da pelagem do animal e, do ambiente em que ele está, obtendo um melhor desempenho nas situações em que o animal está camuflado no ambiente.

As imagens informam também para cada região à probabilidade predita pela CNN, ou seja, a probabilidade que a CNN determinou para uma região de interesse ser da classe identificada. Essa probabilidade é obtida na última camada da rede neural convolucional, pelas pontuações da região avaliada para todas as classes. Podemos observar, que em todos os casos a abordagem proposta também obteve uma maior probabilidade predita nas suas regiões.

Isso é importante, pois se a probabilidade for muito baixa, quer dizer que a CNN não está conseguindo classificar corretamente a região de interesse detectada pela abordagem. Logo, podemos usar essa probabilidade para verificar o desempenho das abordagens em propor boas regiões para a classificação.

A diferença nas probabilidades preditas obtidas pelas duas abordagens podem ser observadas principalmente nas imagens que possuem apenas um animal. Na primeira linha temos a imagem de um papagaio onde a abordagem proposta teve uma probabilidade predita de $95,87\%$ contra $58,86\%$ para a abordagem *Fast R-CNN*. Na segunda linha temos a imagem de uma anta, onde a abordagem proposta conseguiu $96,66\%$ de probabilidade contra $88,15\%$ do *Fast R-CNN*, e na última linha temos uma imagem de um tamanduá-bandeira, onde a abordagem proposta obteve uma probabilidade predita de $99,42\%$ contra $67,07\%$ da abordagem *Fast R-CNN*.

A diferença na probabilidade predita que a abordagem proposta conseguiu sobre a abordagem *Fast R-CNN*, ocorreu principalmente por ela conseguir delimitar melhor o animal na imagem através das imagens térmicas. Assim, durante a classificação somente a região que contêm o animal é passado pela CNN, sem outras características da região de fundo que atribuem erro ao processo de classificação.

CAPÍTULO

6

Conclusão

Neste trabalho nós propomos uma abordagem de visão computacional para a identificação de animais silvestres do Pantanal. Nós desenvolvemos uma nova metodologia que usa imagens térmicas para localizar o animal na cena e propor regiões para serem classificadas pela rede neural convolucional. O método utiliza o algoritmo de segmentação SLIC e a diferença de temperatura entre o corpo do animal e o seu ambiente para detectar as regiões de interesse. Além disso, nós analisamos duas arquiteturas de redes neurais convolucionais com o intuito de avaliar se a profundidade melhora o desempenho da abordagem.

A abordagem proposta superou desafios clássicos de identificação em imagens visuais, como os problemas de iluminação, e a semelhanças entre a pelagem do animal e o seu ambiente. Além disso, neste trabalho nós construímos uma base de imagens composta por 1.600 imagens térmicas e visuais de espécies do Pantanal ameaçadas de extinção.

A abordagem proposta apresentou um desempenho superior a técnica *Fast R-CNN* sobre a base de imagens analisada, baseado nas métricas de validação precisão média, medida-F e em testes estatísticos como a análise de variança ANOVA e o pós-teste de Tukey. Os resultados experimentais indicaram que a nossa abordagem é capaz de detectar e classificar animais silvestres do Pantanal com uma precisão média de $86,35\%$, contra $78,95\%$ obtida pela técnica *Fast R-CNN* para a melhor arquitetura analisada. No geral, a nossa abordagem alcançou em média uma precisão $5,5\%$ maior que as implementações do *Fast R-CNN*, chegando a alcançar 10% e 12% para algumas classes de animais. Além disso, verificamos que a inserção de mais camadas melhoraram em média $3,5\%$ o desempenho das abordagens.

Nas etapas futuras, nós pretendemos expandir a base de imagens térmicas e visuais de espécies animais do Pantanal, e comparar a abordagem proposta com outras técnicas como o *Faster R-CNN* e Yolo2. Além disso, iremos avaliar diferentes arquiteturas de redes neurais convolucionais como a ResNet.

Referências Bibliográficas

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2274–2282, November 2012. ISSN 0162-8828. Citado nas páginas 11, 12, 37, e 39.

J. Aguzzi, C. Costa, Y. Fujiwara, R. Iwase, E. Ramirez-Llorda, and P. Menesatti. A novel morphometry-based protocol of automated video-image analysis for species recognition and activity rhythms monitoring in deep-sea fauna. *Sensors*, 9(11):8438–8455, 2009. Citado na página 6.

B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, Nov 2012. ISSN 0162-8828. Citado na página 30.

C. Alho and J. Sabino. A conservation agenda for the Pantanal's biodiversity. *Brazilian Journal of Biology*, 71:327 – 335, 04 2011. ISSN 1519-6984. Citado nas páginas 1, 2, e 5.

P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 328–335, June 2014. Citado na página 30.

M. S. Arruda, B. B. Machado, W. N. Gonçalves, L. Cullen, J. H. P. Dias, C. C. Garcia, and J. F. R. Jr. Thermal image segmentation in studies of wildlife animals. In *XI Workshop de Visão Computacional*, pages 204–209, 2015. ISBN 978-85-8023-032-1. Citado nas páginas 6 e 53.

B. M. Bolstad, R. A. Irizarry, M. Åstrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, 2003. Citado na página 22.

Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML)*, pages 111–118, 11 2010. Citado na página 22.

G. Chen, T. X. Han, Z. He, R. Kays, and T. Forrester. Deep convolutional neural network based species recognition for wild animal monitoring. In *IEEE International Conference on Image Processing (ICIP)*, pages 858–862, Oct 2014. Citado nas páginas 2, 6, e 7.

B. A. Collier, S. S. DITCHKOFF, J. B. RAGLIN, and J. M. SMITH. Detection probability and sources of variation in white-tailed deer spotlight surveys. *The Journal of Wildlife Management*, 71(1):277–281, 2007. ISSN 1937-2817. Citado na página 9.

J. d. S. V. da Silva and M. de Moura Abdon. Delimitação do pantanal brasileiro e suas sub-regiões. *Pesquisa Agropecuária Brasileira*, 33(13):1703–1711, 1998. Citado na página 1.

G. de Andrade Martins. *Estatística geral e aplicada .* Editora Atlas SA, 2000. Citado na página 58.

E. H. Fegraus, K. Lin, J. A. Ahumada, C. Baru, S. Chandra, and C. Youn. Data acquisition and management software for camera trap data: A case study from the {TEAM} network. *Ecological Informatics*, 6(6):345 – 353, 2011. ISSN 1574-9541. Citado nas páginas 2 e 5.

P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004. ISSN 1573-1405. Citado na página 31.

D. L. Garner, H. B. Underwood, and W. F. Porter. Use of modern infrared thermography for wildlife population surveys. *Environmental Management*, 19(2):233–238, 1995. ISSN 1432-1009. Citado na página 9.

R. Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, December 2015. Citado nas páginas 43, 49, 52, 53, e 54.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, June 2014. Citado na página 33.

A. Gomez, A. Salazar, and F. Vargas. Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very

deep convolutional neural networks. *ArXiv e-prints*, mar 2016. Citado nas páginas 2, 6, e 7.

T. R. Goodall, A. C. Bovik, and N. G. Paulter. Tasking on natural statistics of infrared images. *IEEE Transactions on Image Processing*, 25(1):65–79, Jan 2016. ISSN 1057-7149. Citado na página 8.

A. Graves, A. r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013. Citado na página 8.

W. M. K. H. B. Graves, E. D. Bellis. Censusing white-tailed deer by airborne thermal infrared imagery. *The Journal of Wildlife Management*, 36(3):875–884, 1972. ISSN 0022541X, 19372817. Citado na página 9.

R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1992. Citado nas páginas 14 e 41.

J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, 28(1):100–108, 1979. Citado nas páginas 11 e 37.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *ArXiv e-prints*, December 2015. Citado nas páginas 6, 7, e 22.

J.-H. Hwang, S. Jun, S.-H. Kim, D. Cha, K. Jeon, and J. Lee. Novel fire detection device for robotic fire fighting. In *International Conference on Control Automation and Systems (ICCAS)*, pages 96–100, October 2010. Citado na página 8.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv e-prints*, 2015. Citado na página 22.

G. R. Iversen, A. R. Wildt, H. Norpoth, and H. P. Norpoth. *Analysis of variance*. Number 1. Sage, 1987. Citado na página 56.

W. J. Junk, C. N. da Cunha, K. M. Wantzen, P. Petermann, C. Strüssmann, M. I. Marques, and J. Adis. Biodiversity and its conservation in the pantanal of mato grosso, brazil. *Aquatic Sciences*, 68(3):278–309, 2006. ISSN 1420-9055. Citado na página 1.

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *The*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1725–1732, June 2014. Citado na página 8.

M. Kastek, R. Dulski, P. Trzaskawka, T. Sosnowski, and H. Madura. Concept of infrared sensor module for sniper detection system. In *35th International Conference on Infrared Millimeter and Terahertz Waves*, pages 1–2, September 2010. Citado na página 8.

R. Kays, S. Tilak, B. Kranstauber, P. A. Jansen, C. Carbone, M. J. Rowcliffe, T. Fountain, J. Eggert, and Z. He. Monitoring wild animal communities with arrays of motion sensitive camera traps. *ArXiv e-prints*, September 2010. Citado na página 6.

D. M. Kline and V. L. Berardi. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications*, 14(4):310–318, 2005. ISSN 1433-3058. Citado na página 25.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. Citado nas páginas 3, 6, 7, 21, 22, 26, e 42.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 2016/05/28 1989. Citado nas páginas 3, 7, 22, e 26.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. Citado na página 22.

A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, December 2009. ISSN 0162-8828. Citado na página 11.

P. Luo, X. Wang, and X. Tang. Hierarchical face parsing via deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2480–2487, June 2012. Citado na página 8.

A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008. Citado na página 11.

Y.-M. H. Ng, M. Yu, Y. Huang, and R. Du. Diagnosis of sheet metal stamping processes based on 3-d thermal energy distribution. *IEEE Transactions on Automation Science and Engineering*, 4(1):22–30, January 2007. ISSN 1545-5955. Citado na página 8.

D. S. J. R. Vadivambal. Applications of thermal imaging in agriculture and food industry - a review. *Food and Bioprocess Technology*, 4(2):186–199, 2011. ISSN 1935-5130. Citado na página 8.

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015. Citado na página 35.

A. Rosenfeld. Connectivity in digital pictures. *J. ACM*, 17(1):146–160, Jan. 1970. ISSN 0004-5411. Citado nas páginas 14 e 41.

A. Rosenfeld. *Digital picture processing*. Academic press, 1976. Citado na página 14.

S. M. Ross. *Introduction to probability and statistics for engineers and scientists*. Academic Press, 2014. Citado na página 54.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. Citado nas páginas 7, 47, e 49.

D. Scherer, A. Müller, and S. Behnke. *Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition*, pages 92–101. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. Citado na página 22.

S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965. Citado na página 56.

J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 731–737, June 1997. Citado na página 11.

H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227 – 244, 2000. ISSN 0378-3758. Citado na página 21.

R. Siegel. Land mine detection. *IEEE Instrumentation Measurement Magazine*, 5(4):22–28, December 2002. ISSN 1094-6969. Citado na página 8.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014. Citado nas páginas 6, 7, 26, 28, 42, e 43.

Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3476–3483, June 2013. Citado na página 8.

F. A. Swarts. *The Pantanal of Brazil, Paraguay and Bolivia: Selected Discourses on the Worlds Largest Remaining Wetland System*. Hudson MacArthur Publishers, January 2000. Citado na página 1.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015. Citado nas páginas 6, 7, e 43.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *ArXiv e-prints*, 2017. Citado na página 22.

G. Tattersall and V. Cadena. Insights into animal temperature adaptations revealed through thermal imaging. *The Imaging Science Journal*, 58(5):261–268, 2010. Citado na página 8.

A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1653–1660, June 2014. Citado na página 8.

J. W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, pages 99–114, 1949. Citado na página 57.

J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104 (2):154–171, 2013. ISSN 1573-1405. Citado nas páginas 30, 31, e 32.

A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015. Citado na página 53.

A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In D. Forsyth, P. Torr, and A. Zisserman, editors, *European Conference of Computer Vision*, volume 5305 of *Lecture Notes in Computer Science*, pages 705–718. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88692-1. Citado na página 11.

O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *European Conference of Computer Vision*, volume 6315 of *Lecture Notes in Computer Science*, pages 211–224. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15554-3. Citado na página 11.

L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, June 1991. ISSN 0162-8828. Citado na página 11.

X. Yu, J. Wang, R. Kays, P. A. Jansen, T. Wang, and T. Huang. Automated identification of animal species in camera trap images. *EURASIP Journal on Image and Video Processing*, (1):52, 2013. ISSN 1687-5281. Citado nas páginas 2, 5, e 6.

C. L. Zitnick and P. Dollár. *Edge boxes: Locating object proposals from edges*, pages 391–405. Springer International Publishing, 2014. ISBN 978-3-319-10602-1. Citado na página 30.