Combinação de Superpixels e Redes Convolucionais para Segmentação de Bovinos

Diogo Nunes Gonçalves

SERVIÇO DE PÓS-GRADUAÇÃO DA FACOM-UFMS

Data de Depósito:

Assinatura:_____

Combinação de Superpixels e Redes Convolucionais para Segmentação de Bovinos

Diogo Nunes Gonçalves

Orientador: *Prof Dr Hemerson Pistori*

Dissertação apresentada à Faculdade de Computação Facom-UFMS como parte dos requisitos necessários à obtenção do título de Mestre em Ciência da Computação.

UFMS - Campo Grande
Novembro/2018

Aos meus pais,
Osmar e Marlene,

Aos meus irmãos e cunhados,
Wesley, Jociane, Pâmela e Anderson,

À namorada,
Isabella,

Ao meu orientador,
Hemerson Pistori.

Agradecimentos

Primeiramente agradeço à Deus pela oportunidade e força que Ele me concedeu nos momentos de dificuldades. Agradeço à minha família, especialmente aos meus pais, avós, irmãos e cunhados pelo apoio incondicional nos momentos difíceis. Agradeço ao meu orientador, professor Dr. Hemerson Pistori pelo apoio, paciência e conselhos durante a sua orientação. Agradeço a minha namorada, Isabella pelo apoio nos momentos que mais precisei. Agradeço aos professores da Faculdade de Computação pelos ensinamentos que recebi durante o mestrado. Agradeço aos colegas do INOVISÃO, pelos ensinamentos obtidos. Agradeço também a Fundação Universidade Federal de Mato Grosso do Sul pela oportunidade de concluir meu mestrado. Por fim, agradeço a NVIDIA pela doação da placa de vídeo utilizada nos experimentos e a CAPES pelo apoio financeiro.

Resumo

A pecuária é uma das áreas mais importantes da economia brasileira, embora ainda possua índices produtivos considerados baixos devido à baixa produtividade média. Para mudar este cenário, tecnologias da pecuária de precisão podem ser adotadas, permitindo um incremento na produtividade e na rentabilidade do produtor. A pecuária de precisão é a medição de diferentes parâmetros dos animais no campo, a modelagem desses parâmetros e o auxílio na tomada de decisão do produtor. Atualmente tem crescido o uso da visão computacional nessas tecnologias para torná-las automáticas. Basicamente essas tecnologias utilizam a visão computacional para interpretar imagens, extraindo suas características. Uma das partes mais importantes de um sistema de visão computacional é a segmentação, que tem como objetivo dividir uma imagem em regiões com propriedades similares, podendo então dividir partes que interessam ao sistema e partes que não interessam (e.g., bovino e fundo). O objetivo deste trabalho é investigar algoritmos de segmentação para solucionar problemas da pecuária de precisão. Para isso, são avaliados dois algoritmos de segmentação: Superpixel+CNN e SegNet. No primeiro algoritmo a imagem é dividida em superpixels, depois uma CNN é utilizada para classificar os superpixels em objeto de interesse (bovino, carcaça, etc) e fundo. No segundo algoritmo, uma CNN é utilizada na abordagem SegNet para classificar cada pixel da imagem nos objetos de interesse. Através destas avaliações foi proposta um novo método que combina estes dois algoritmos, onde a ideia é obter as suas vantagens. Os algoritmos foram avaliados em dois problemas da pecuária de precisão, tipificação de carcaças cujo o objetivo é atribuir um valor correspondente a sua qualidade e segmentação de bovinos, que pode ser um primeiro passo para a pesagem de um animal através de uma imagem, por exemplo. Para isso, foram construídas duas bases de imagens, uma de carcaça com 226 imagens obtidas em um frigorífico na região de Mato Grosso do Sul e outra com 154 imagens de bovinos obtidos em uma fazenda na mesma região. O melhor resultado de segmentação das carcaças usando a métrica

acurácia pixel-a-pixel foi de 98.9% obtido pela combinação OR, uma das variações do método proposto neste trabalho. Já para a métrica IoU, o melhor resultado foi de 93.6% obtido pelas combinações MAX, MULT e MEAN, também propostas neste trabalho. Para o problema de segmentação de bovinos, o melhor resultado para a métrica acurácia pixel-a-pixel foi de 94.1% obtido pela combinação OR. Para o IoU, a SegNet obteve um resultado de 83.5%, sendo superior aos outros algoritmos.

Abstract

Cattle raising is one of the most important areas of the Brazilian economy, although it still has low production rates due to low average productivity. To change this scenario, precision livestock technologies can be adopted, allowing an increase in productivity and profitability of the producer. The precision livestock is the measurement of different parameters of the animals in the field, the modeling of these parameters and the aid in the decision making of the producer. Currently, the use of computer vision in these technologies has grown to make them automatic. Basically these technologies use the computer vision to interpret images by extracting their features. One of the most important parts of a computer vision system is the segmentation, which aims to divide an image into regions with similar properties (e.g., cattle and background). The objective of this work is to investigate segmentation algorithms to solve problems of precision livestock. For this, two segmentation algorithms are evaluated: Superpixel+CNN and SegNet. In the first algorithm, the image is divided into superpixels, then a CNN is used to classify the superpixels into object of interest (cattle, carcass, etc.) and background. In the second algorithm, a CNN is used in the SegNet approach to classify each pixel of the image in the object of interest. Through these evaluations a new method was proposed combining these two algorithms, where the main idea is to obtain their advantages. The algorithms were evaluated in two precision livestock problems, carcass typification whose objective is to assign a value corresponding to their quality and segmentation of cattle, which can be a first step for weighing an animal through an image, for example. For this purpose, two image databases were constructed: a carcass database with 226 images obtained in a slaughterhouse in the region of Mato Grosso do Sul and the other database with 154 images of cattle obtained on a farm in the same region. The best segmentation result of the carcasses database using the pixel accuracy metric was 98.9% obtained by the OR combination, one of the variations of the method proposed in this work. For the IoU metric, the best result was 93.6%

obtained by the MAX, MULT and MEAN combinations, also proposed in this work. For the cattle segmentation problem, the best result for the pixel accuracy metric was 94.1% obtained by the OR combination. For the IoU, SegNet obtained a result of 83.5%, being superior to the other algorithms.

Sumário

Sumário xiv
Lista de Figuras xvii
Lista de Tabelas xix
Lista de Abreviaturas xxi
Lista de Algoritmos xxiii

1 Introdução **1**
1.1 Motivação 1
1.2 Objetivos 2
1.3 Trabalhos Correlatos 3
1.3.1 Visão Computacional na Pecuária de Precisão 3
1.3.2 Segmentação com Superpixels 3
1.3.3 Redes Neurais Convolucionais 4
1.4 Principais Contribuições 5
1.5 Organização 6

2 Referencial Teórico **7**
2.1 Superpixels 7
2.2 Redes Neurais Convolucionais 9
2.2.1 Camada Convolucional 9
2.2.2 Camada ReLU 11
2.2.3 Camada *Pooling* 11
2.2.4 Camada Totalmente Conectada 12
2.2.5 Softmax 13
2.3 Arquiteturas 13
2.3.1 Visual Geometry Group Network (VGGNet) 14
2.3.2 ResNet 15

3 Algoritmos de Segmentação **19**
3.1 Superpixel + CNN 19

3.1.1 Segmentação das Imagens em Superpixels 19
3.1.2 Treinamento da CNN 20
3.1.3 Classificação de Superpixels 21
3.2 SegNet 22
3.2.1 Codificador 22
3.2.2 Decodificador 23
3.2.3 Arquiteturas da SegNet 24

4 Método Proposto **27**
4.1 MEAN 27
4.2 MULT 28
4.3 MAX 28
4.4 OR 29
4.5 AND 30

5 Materiais e Métodos **31**
5.1 Base de Imagens 31
5.1.1 Carcaça 31
5.1.2 Boi em pé 34
5.2 Delineamento Experimental 34
5.3 Métricas de Avaliação 36

6 Resultados e Discussões **39**
6.1 Treinamento das CNNs 39
6.1.1 Superpixel + CNN 39
6.1.2 SegNet 40
6.2 Comparação entre Algoritmos de Segmentação 42
6.3 Resultados do Método Proposto 47

7 Conclusão **51**

Referências **55**

Lista de Figuras

2.1 Exemplo da aplicação do SLIC com $k = 100$ e $k = 500$. 9

2.2 Neste exemplo, um volume de entrada $V_e^{w,h,d}$ é convoluído com um conjunto de filtros F. Nota-se que a saída desta convolução é um volume $V_s \in \Re^{w_s,h_s,n}$, com n sendo a quantidade de filtros. 10

2.3 Neste exemplo, um filtro representado em laranja é deslocado por um determinado *stride*. 11

2.4 Exemplo operação de pooling em dois retângulos representado em laranja e verde, respectivamente. Ambos os retângulos possuem tamanho 2×2 e o *stride* utilizado no deslocamento é 2. . . . 12

2.5 Exemplo da arquitetura de uma camada totalmente conectada. Note que w_e é a quantidade de neurônios da entrada, $f(i, j)$ é o peso da conexão entre o neurônio i e o neurônio j e w_s é a quantidade de neurônios da saída. 13

2.6 Desenho da arquitetura da VGGNet de 16 camadas. Algo interessantes sobre sua arquitetura é que, após a AlexNet ter sido projetada com camadas de convoluções seguidas, a VGGNet também foi projetada desta forma. 15

2.7 Desenho da arquitetura da VGGNet de 19 camadas. Note que, com a inclusão de 3 novas camadas convolucionais, o número total de camadas de convoluções é 16. 15

2.8 Ilustração do aprendizado residual da ResNet. Note que a soma das matrizes ocorre após duas camadas de convolução. 16

2.9 Ilustração da arquitetura da ResNet50. Note que as camadas foram divididas em 4 blocos em cinza que possui um padrão. Em cima de cada bloco está a quantidade de vezes que se repete o padrão que o bloco em questão é repetido. 17

3.1 Exemplo da criação e rotulação do banco de superpixels. Na primeira coluna o algoritmo SLIC é aplicado em cada uma das imagens de treinamento de forma que o banco de superpixels seja formado, conforme a segunda coluna. A terceira coluna ilustra a rotulação dos superpixels com as classes do problema (fundo e bovino). 20

3.2 Exemplo de transferência de aprendizagem da arquitetura VGG16. As camadas convolucionais são inicializadas com um CNN pré-treinada, enquanto que as camadas totalmente conectadas são inicializadas com pesos aleatórios. 21

3.3 Exemplo da SegNet com duas partes: codificador e decodificador. O codificador é composto por camadas convolucionais e pooling. Por outro lado, o decodificador é composto por camadas convolucionais, upsampling e softmax. 22

3.4 Exemplo da camada de *pooling* e *upsampling*. A camada de *pooling* calcula o mapa de características e os índices (inteiro pré-definido para cada posição). Por outro lado, a camada de *upsampling* recebe o mapa de características de baixa resolução e os índices para construir um mapa de ativação de alta resolução. . . 24

3.5 Exemplo da arquitetura da SegNet (VGG16). O retângulo superior corresponde a parte do codificador e o inferior a parte do decodificador. 25

4.1 Exemplo da combinação MEAN onde $M_{SCNN}(x,y,0)$ corresponde a probabilidade do pixel (x,y) pertencer ao objeto de interesse, $M_{SCNN}(x,y,1)$ a probabilidade de pertencer ao fundo e o resultado da segmentação do Superpixel+CNN (**F**: fundo e **O**: objeto de interesse). Da mesma forma $M_{SegNet}(x,y,0)$ e $M_{SegNet}(x,y,1)$ correspondem a SegNet. Já o $p(x,y,0)$ corresponde a média de $M_{SCNN}(x,y,0)$ e $M_{SegNet}(x,y,0)$ e $p(x,y,1)$ a média de $M_{SCNN}(x,y,1)$ e $M_{SegNet}(x,y,1)$. O resultado da segmentação da combinação difere tanto do Superpixel+CNN como da SegNet, considerando as probabilidades médias. 29

5.1 Exemplos de quatro imagens presentes na base de imagens utilizadas nos experimentos. Partes da imagem estão borrada devido a questões legais. 32

5.2 Exemplos de três imagens anotadas manualmente para realizar os experimentos. A primeira coluna apresenta as imagens originais e a segunda apresenta as imagens anotadas. Os pixels em verde pertencem a classe carcaça enquanto que os pixels em preto pertencem ao fundo. O rosto e algumas partes da imagem estão borradas por questões legais. 33
5.3 Exemplos de quatro imagens presentes na base de imagens utilizadas nos experimentos. 34
5.4 Exemplos de duas imagens anotadas manualmente para realizar os experimentos. A primeira coluna apresenta as imagens originais e a segunda apresenta as imagens anotadas. Os pixels em verde pertencem a classe boi enquanto que os pixels em preto pertencem ao fundo. 35
5.5 Exemplo da Matriz de Confusão para os problemas deste trabalho. 37
5.6 Interpretação da medida Intersecção sobre a União. 38

6.1 Função de perda para o treinamento das CNNs por 50 épocas usando diferentes valores de k para as duas bases de imagens. . 40
6.2 Função de perda para o treinamento da SegNet com VGG16, VGG19 e ResNet50 para as duas bases de imagens e por 150 épocas. 41
6.3 Resultados da classificação de superpixels obtidos com $k = 100, 500$ e 1000 para o algoritmo Superpixel+CNN (VGG16). Os rostos estão borrados por questões legais. 43
6.4 Resultados da segmentação para SegNet com VGG16, VGG19 e ResNet50. Os rostos estão borrados por questões legais. 43
6.5 Resultados comparativos da SegNet e Superpixel+CNN com as melhores arquiteturas. Os resultados apresentados da Superpixel+CNN são para $k = 1000$. Os rostos e partes da imagem foram borrados por questões legais. 45
6.6 Resultados comparativos da SegNet e Superpixel+CNN em 3 imagens da base boi em pé. Os resultados apresentados são para $k = 500$ e melhores arquiteturas. 46
6.7 Resultados comparativos da SegNet (VGG16), Superpixel+CNN (VGG16, $k = 1000$) e as combinações MEAN e OR em três imagens da base carcaça. Os rostos e partes da imagem foram borrados por questões legais. 49
6.8 Resultados comparativos da SegNet (VGG19), Superpixel+CNN (Resnet50, $k = 1000$) e as combinações MEAN e OR em três imagens da base boi em pé. 49

Lista de Tabelas

4.1 Exemplo da combinação OR que foi implementada neste trabalho, onde O representa um pixel classificado como objeto de interesse e F é quando um pixel é classificado como fundo. 30
4.2 Exemplo do operador lógico AND implementado, onde O representa representa um pixel classificado como objeto de interesse e F é quando um pixel é classificado como fundo. 30

6.1 Acurácia no conjunto de treinamento e validação do algoritmo Superpixel+CNN com diferentes configurações (arquiteturas, número de superpixels) para as duas bases de imagens. A acurácia corresponde ao número de superpixels corretamente classificados pela CNN. 41
6.2 Acurácia no conjunto de treinamento e validação da SegNet com a VGG16, VGG19 e ResNet50 para as duas bases de imagens. A acurácia corresponde ao número de pixels corretamente classificados pela SegNet. 42
6.3 Comparação entre os algoritmos de segmentação: Superpixel+CNN e SegNet para a base carcaça. 44
6.4 Comparação entre os algoritmos de segmentação: Superpixel+CNN e SegNet para a base boi em pé. 46
6.5 Resultados das combinações das abordagens Superpixel+CNN e SegNet para a base de imagens carcaça. Foram utilizados a Superpixel+CNN (VGG16, $k = 1000$) e a SegNet (VGG16), pois foram as que obtiveram os melhores resultados nesta base. 48
6.6 Resultados das combinações das abordagens Superpixel+CNN e SegNet para a base de imagens boi em pé. Foi utilizado para os experimentos a Superpixel+CNN (Resnet50, k=1000) e a SegNet (VGG19), pois foram as que obtiveram os melhores resultados nesta base. 50

Lista de Abreviaturas

CNN Rede Neural Convolucional - *Convolutional Neural Network*

FCN Rede Totalmente Convolucional - *Fully Convolutional Network*

SLIC *Simple Linear Iterative Clustering*

VGG *Visual Geometry Group Network*

ResNet *Residual Network*

FN Falso Negativo

FP Falso Positivo

VN Verdadeiro Negativo

VP Verdadeiro Positivo

AP Acurácia pixel-a-pixel

IoU Intersecção sobre União - *Intersection over Union*

ILSVRC ImageNet Large Scale Visual Recognition Competition

Lista de Algoritmos

1 SLIC é implementado como um *k-means* local. 8

CAPÍTULO

1

Introdução

1.1 Motivação

A pecuária é um dos segmentos que mais se destaca na economia brasileira. O Brasil detém o maior rebanho bovino comercial do mundo, com mais de 226 milhões de cabeças, e o abate supera 35 milhões de animais por ano. Em 2016, o PIB do agronegócio correspondeu a 24% do PIB do Brasil, sendo que a pecuária representou 31% do agronegócio. O Brasil produziu aproximadamente 10 milhões de toneladas de carne bovina e exportou mais de 1,630 milhão de toneladas para diversos países em 2017 (Kist, 2017a). No mesmo período, a pecuária de leite produziu 34.823 milhões de litros de leite, sendo o quarto maior produtor de leite mundial (Kist, 2017b). No cenário nacional, o Mato Grosso do Sul é destaque em relação ao abate de carnes bovinas ficando atrás somente de Mato Grosso e São Paulo, totalizando 24.308 cabeças por dia (Associação Brasileira Indústrias Exportadoras Carnes, 2017).

Apesar dos números elevados, o sistema nacional ainda apresenta índices produtivos pouco representativos devido à baixa produtividade média por área comparada a outros países. Em 10 anos, o objetivo do Brasil é dobrar a lotação animal por área, passando de 1,3 para 2,6 cabeças por hectare, e saltar de 1,4 mil litros de leite por vaca ao ano para 2 mil litros (Beling, 2014).

Para alcançar esses objetivos, é necessário adotar tecnologias por meio da pecuária de precisão que permitam incrementos em produtividade e maior rentabilidade ao produtor. A pecuária de precisão consiste na medição de diferentes parâmetros dos animais e do ambiente, a modelagem desses dados e o seu uso para tomada de decisões. Para tornar as medições automáticas,

métodos computacionais com base em imagens estão sendo cada vez mais estudados e propostos na literatura (Frost et al., 1997). Estes métodos utilizam a visão computacional cujo objetivo é construir sistemas artificiais para interpretar imagens. Por exemplo, um sistema de visão computacional poderia, a partir de uma imagem, estimar o peso de um bovino ou atribuir um valor correspondente a qualidade da carcaça. Com tais sistemas, torna-se possível a análise rápida de grandes rebanhos além de possibilitar o controle individual de cada animal.

Em geral, um sistema de visão computacional é dividido em cinco etapas: aquisição de imagens, pré-processamento, segmentação, extração de características e reconhecimento de padrões. Em particular, a segmentação de imagens é o processo de dividir uma imagem em regiões com propriedades similares para facilitar a sua análise. Para os problemas deste trabalho, a segmentação de imagens consiste em separar os pixels da imagem em duas classes: regiões que contém o objeto de interessse (bovino, carcaça, etc) e as regiões sem interesse ou fundo.

Geralmente, a etapa de segmentação afeta diretamente os resultados das etapas posteriores. Por exemplo, dada a região da imagem que contém o bovino obtida a partir da segmentação, o sistema pode estimar o peso ou outra medida desejada. Dessa forma, erros na segmentação poderiam afetar a estimativa relacionada ao peso. Portanto, a segmentação é considerada uma das etapas mais importantes e complexas de um sistema de visão computacional, tornando muito importante a escolha das técnicas ou algoritmos de segmentação.

1.2 Objetivos

A partir desses direcionamentos, o objetivo geral deste trabalho é o desenvolvimento de um método para segmentar imagens em problemas da pecuária de precisão. Para isso, propomos uma combinação entre superpixels e redes neurais convolucionais, pois ambos os algoritmos possuem resultados promissores em segmentação de imagens.

Para alcançar o objetivo geral, os seguintes objetivos específicos foram definidos:

- Construir duas bases de imagens: carcaça e boi em pé;
- Propor e implementar formas de combinações entre rede neural convolucional e superpixel;
- Validar as combinações propostas comparando com métodos da literatura que usam superpixels e/ou redes neurais convolucionais;

1.3 Trabalhos Correlatos

Esta seção apresenta os trabalhos de visão computacional na pecuária de precisão e trabalhos sobre as duas técnicas abordadas neste projeto, redes neurais convolucionais e superpixel.

1.3.1 Visão Computacional na Pecuária de Precisão

Com os avanços recentes da computação, sistemas assistidos por computador tem tornado possível a automatização de processos em várias atividades, tais como na pecuária de precisão. Vários trabalhos de visão computacional tem sido propostos, como o trabalho proposto por Hertem et al. (2013). Os autores adaptaram cinco algoritmos de segmentação baseados em subtração de fundo, e testaram em fundos dinâmicos (campo aberto) e estáticos (foi introduzida uma parede atrás dos bovinos). Eles perceberam que o melhor algoritmo foi baseado em detecção de bordas por subtração de fundo com erro médio absoluto de $6.7(\pm 5.7)$ pixels, e que nenhum algoritmo produziu resultados satisfatórios em fundos dinâmicos.

Viazzi et al. (2014) compararam sistemas de câmeras 2-D e 3-D para medir a postura traseira em vacas leiteiras. Eles perceberam que sistemas com câmeras 2-D, com vista lateral do bovino, não possuem bons resultados principalmente pela diferença de iluminação, podendo ocorrer sombras que dificultam uma boa segmentação. Para contornar este problema, a câmera foi colocada acima do bovino, conseguindo melhorar a precisão na segmentação e obtendo uma acurácia de 91% utilizando câmeras 2-D e 90% utilizando câmeras 3-D.

Alguns trabalhos também utilizam métodos de segmentação para estimar o peso e o escore do bovino. Vindis et al. (2010) propuseram uma segmentação de bovino através da imagens térmicas para estimar o peso do bovino. Uma dificuldade deste método é que o bovino precisa estar com uma temperatura mais elevada do que o ambiente a sua volta, então os autores fizeram a captura das imagens com o bovino rodeado por paredes resfriadas.

1.3.2 Segmentação com Superpixels

Superpixel é uma abordagem que divide uma imagem em regiões com propriedades similares, como cor, textura e brilho. Estas regiões similares são chamadas de superpixels. Os algoritmos de superpixels podem ser divididos em baseados em grafos e baseados em gradiente.

Em algoritmos baseados em grafos, cada pixel é mapeado em um nó do grafo, e o peso da conexão entre dois nós é dado pela similaridade entre eles.

O algoritmo *Normalized Cuts* (Shi and Malik, 2000) divide a imagem em grafos recursivamente através de similaridades de contornos e texturas. A principal desvantagem desse algoritmo é o seu elevado custo computacional (Levinshtein et al., 2009). Felzenszwalb and Huttenlocher (2004) também propuseram um algoritmo para obter superpixels baseados em grafos. Este algoritmo agrupa os nós de forma aglomerativa de tal forma que cada superpixel é uma árvore geradora miníma dos pixels.

Por outro lado, os algoritmos baseados em gradiente realizam um agrupamento inicial, e a cada iteração um refinamento baseado em gradiente ocorre para obter uma melhor segmentação. O algoritmo Mean-shift (Comaniciu and Meer, 2002) é um algoritmo baseado em busca onde o centroide de cada agrupamento se move de acordo com o gradiente local. O algoritmo não impõe restrição com relação ao número, tamanho ou compactação dos superpixels. Outro algoritmo baseado em gradiente é o Quick-shift (Vedaldi and Soatto, 2008), que move cada ponto no espaço de característica para o vizinho mais próximo que aumente a densidade de Parzen. Este algoritmo também não restringe o tamanho e a quantidade de superpixels.

O principal algoritmo dessa categoria e comumente usado como base é o SLIC (*Simple Linear Iterative Clustering*) proposto por Achanta et al. (2012). O SLIC é baseado na aplicação do k-means localmente para gerar k superpixels usando como similaridade a cor e a posição espacial dos pixels (x,y). Este algoritmo está detalhado na Seção 2.1.

1.3.3 Redes Neurais Convolucionais

As redes neurais convolucionais (*Convolutional Neural Network* - CNN) consistem em camadas convolucionais, que utilizam o algoritmo de *backpropagation* para aprender os parâmetros/filtros de cada camada. Desde a sua criação, a CNN tem sido caracterizada por três propriedades básicas, as conexões locais, o compartilhamento de peso e o *pooling* local. As duas primeiras propriedades permitem que o modelo aprenda os padrões visuais locais importantes com menos parâmetros ajustáveis que um modelo totalmente conectado, e a terceira propriedade prepara a rede para possuir invariância à translação (LeCun et al., 1989).

Uma das primeiras propostas de redes neurais convolucionais é a rede LeNet-5 descrita por LeCun et al. (1989) para o reconhecimento óptico de caracteres. Comparada às redes convolucionais profundas atuais, a rede foi relativamente modesta devido aos recursos computacionais limitados da época e aos desafios do treinamento de algoritmos para redes com mais camadas. Embora houvesse muito potencial em redes convolucionais mais profundas, só recentemente elas se tornaram predominantes, devido ao aumento do poder

computacional atual, da quantidade de dados para treinamento disponíveis na internet e do desenvolvimento de métodos mais eficazes para a formação de tais modelos.

Um exemplo recente e notável do uso de redes convolucionais profundas para classificação de imagens é o desafio Imagenet (Krizhevsky et al., 2012) em que uma CNN obteve um erro consideravelmente menor comparado com o erro das abordagens tradicionais de visão computacional (usando SIFT e Máquinas de Vetores de Suporte). Redes neurais convolucionais também obtiveram recentemente sucesso para diferentes aplicações, incluindo estimação de pose humana (Toshev and Szegedy, 2014), análise de faces (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).

Com os resultados promissores em classificação, as redes neurais convolucionais foram recentemente estendidas para o problema de segmentação (Shelhamer et al., 2017). Nessas extensões, as redes neurais convolucionais possuem apenas camadas de convolução, sendo portanto chamadas de Redes Totalmente Convolucionais (*Fully Convolutional Networks* - FCN). A saída da FCN é uma imagem do mesmo tamanho da entrada sendo que cada pixel da saída corresponde à probabilidade do pixel pertencer a uma determinada classe. Dessa forma, a imagem de saída é utilizada para segmentação de imagens.

Após os resultados promissores das FCNs, Badrinarayanan et al. (2017) propuseram a SegNet, uma rede neural convolucional para segmentação. A arquitetura de segmentação é composta por camadas de codificação, camadas de decodificação e uma camada de classificação pixel-a-pixel. As camadas de codificação atuam da mesma maneira como as camadas das CNNs anteriores, codificando a imagem de entrada em um mapa de características com baixa resolução. Por outro lado, as camadas de decodificação recebem a saída com baixa resolução e aumentam a resolução até que a saída esteja do mesmo tamanho da imagem de entrada. Os resultados da SegNet são promissores e podem ser aplicados nos problemas desse trabalho.

1.4 Principais Contribuições

Neste trabalho foram avaliados dois algoritmos, Superpixel+CNN e SegNet, que tem recebido uma crescente atenção na literatura. Ambos algoritmos foram aplicados na segmentação de carcaças e bovinos usando imagens coloridas. Para isso, dois conjuntos com 380 imagens foram construídos e anotados manualmente.

Como contribuição principal, foi proposto um novo método de combinação

do Superpixel+CNN e a SegNet. Basicamente, a metodologia de combinação recebe a saída de ambos algoritmos e combinam as probabilidades por meio de operações lógicas e matemáticas, tais como OR, AND, MAX, MULT e MEAN.

De acordo com os resultados experimentais, o Superpixel+CNN obtem resultados precisos nas bordas dos objetos de interesse, embora perca contexto quando o superpixel é pequeno. Os resultados da SegNet mostraram que há um maior contexto devido a utilização de toda a imagem para a classificação de um único pixel, porém em detrimento de uma precisão nas bordas.

Apesar dos resultados promissores dos algoritmos propostos na literatura, o método proposto neste trabalho forneceu incrementos na acurácia, alcançando 98,9% e 94,1% nos problemas de carcaça e bovinos, respectivamente. Portanto, estes resultados mostram que o método proposto considera as vantagens de cada algoritmo, obtendo assim incrementos nos resultados.

1.5 Organização

O restante desse trabalho é descrito como segue. O Capítulo 2 apresenta uma descrição detalhada dos dois algoritmos abordados neste trabalho: superpixels e redes neurais convolucionais. Os dois algoritmos de segmentação (Superpixels+CNN e SegNet) são descritos no Capítulo 3. O método proposto que combina o resultado dos dois algoritmos é detalhado no Capítulo 4. No Capítulo 5 são apresentadas as bases de imagens, a configuração experimental e as métricas de avaliação. Os resultados e discussões dos algoritmos de segmentação são apresentados no Capítulo 6. Por fim, o Capítulo 7 apresenta as conclusões e os trabalhos futuros.

CAPÍTULO

2

Referencial Teórico

2.1 Superpixels

Dada uma imagem I com dimensões $w \times h$, os algoritmos de segmentação por superpixel tem como objetivo dividir I em um conjunto de k superpixels $S = \{S_1, S_2, \ldots, S_k\}$. Cada superpixel S_i é composto por pixels que representam um componente conectado. Dessa forma, pode-se definir uma função $w \times h \rightarrow k$ que atribui um único superpixel para cada pixel $p_j \in w \times h$. Cada algoritmo constrói a função de forma diferente, sendo um destes algoritmos o SLIC proposto por Achanta et al. (2012). O SLIC gera agrupamentos com base na similaridade de cor e proximidade no plano da imagem.

O processo de agrupamento inicia com a formação de um conjunto de centroides $C = \{c_1, c_2, \ldots, c_k\} \mid c_i = [l_i, a_i, b_i, x_i, y_i]$, onde l_i, a_i, b_i são componentes do espaço de cor CIELAB e x_i, y_i representam uma posição espacial na imagem. Os centroides são inicializados de forma que cada centroide esteja a uma distância $g = \sqrt{\frac{w \times h}{k}}$ entre si, formando assim grades de tamanho g. Para evitar que um centroide seja inicializado em um pixel de borda ou ruidoso, cada centroide é deslocado para o menor gradiente da região de tamanho 3×3 ao seu redor.

O próximo passo é atribuir para cada pixel $p_j = [l_j, a_j, b_j, x_j, y_j]$ da imagem o seu centroide mais próximo usando a Equação 2.3. Essa equação combina de forma ponderada a distância de cor (Equação 2.1) e espacial (Equação 2.2). Para reduzir a quantidade de cálculos, somente pixels em uma região $2g \times 2g$ ao redor de cada centroide são de seu interesse.

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \tag{2.1}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - x_j)^2} \tag{2.2}$$

$$d = \sqrt{d_c^2 + \left(\frac{d_s}{g}\right)^2 m^2} \tag{2.3}$$

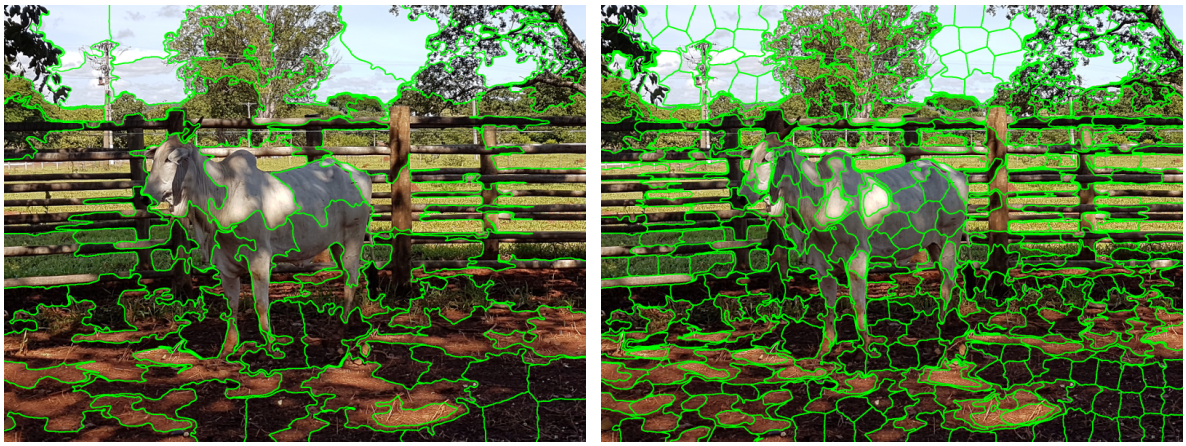
onde d_c representa a diferença de cor no espaço CIELAB, d_s é a distância espacial na imagem, $g = \sqrt{\frac{w \times h}{k}}$ e m é uma constante que permite definir a importância que d_c e d_s tem na equação. Quanto maior for o valor de m, d_s terá uma maior importância, resultando em superpixels mais compactos. Por outro lado, se m for pequeno, os superpixels se ajustam melhor as bordas, porém suas formas terão tamanhos diferentes e formatos não regulares.

Após as atribuições, cada centroide é atualizado com a média dos pixels atribuídos a ele. Os passos de atribuição e atualização dos centroides são repetidos de forma iterativa até os centroides convergirem ou um limite de iteração seja alcançado. O SLIC é descrito no Algoritmo 1.

Algoritmo 1: SLIC é implementado como um *k-means* local.

```
Inicializar os centroides c_i por meio de amostragem de pixels em uma
grade regular com distância g;
Mover os centroides para o menor gradiente de uma vizinhança 3 × 3;
Definir rótulo r_j = −1 para cada pixel p_j;
Definir distância d_j = ∞ para cada pixel p_j;
repeat
    for c_i em C do
        for cada pixel p_j na região 2g × 2g ao redor de c_i do
            Calcular a distância d entre c_i e p_j (Equação 2.3);
            if d < d_j then
                d_j = d;
                r_j = i;
            end
        end
    end
    Atualizar os centroides;
until limite de iterações seja alcançado ou os centroides convergirem ;
```

A Figura 2.1 apresenta o resultado da aplicação do SLIC com $k = 100$ e 500. Para valores altos de k, os superpixels são mais ajustados as bordas, entretanto, a região que ele representa é menor e com menos informação.



(a) k = 100 (b) k = 500

Figura 2.1: Exemplo da aplicação do SLIC com $k = 100$ e $k = 500$.

2.2 Redes Neurais Convolucionais

A arquitetura de uma CNN é formada por diversas camadas que transformam o volume de entrada (e.g., imagem colorida) em um volume de saída (e.g., probabilidade das classes). Os principais tipos de camadas, descritas nas seções seguintes, são camada convolucional, camada ReLU, camada de pooling, camada totalmente conectada e softmax.

2.2.1 Camada Convolucional

A camada convolucional é responsável pelo maior processamento de uma rede neural convolucional. Esta camada recebe um volume de entrada, convolui com um conjunto de filtros para produzir um volume de saída. Os parâmetros da camada convolucional consistem em um conjunto de n filtros $F = \{f_1, \ldots, f_n\} \mid f_i \in \Re^{w_f, h_f, d_f}$, onde w_f é a largura, h_f é a altura e d_f é a profundidade do filtro e um conjunto de n bias. Geralmente estes filtros são pequenos espacialmente, por exemplo, volumes com tamanho $5 \times 5 \times 3$. Após o aprendizado destes filtros por *backpropagation*, o volume de entrada $V_e \in \Re^{w,h,d}$ é convoluído deslizando cada filtro através de todo o volume de entrada, computando o produto ponto a ponto entre o filtro e a região correspondente. Considere um conjunto F de filtros e um volume de entrada V_e, a convolução é calculada da seguinte forma:

$$V_s(x,y,i) = \sum_{l=0}^{w_f-1} \sum_{m=0}^{h_f-1} \sum_{o=0}^{d_f-1} V_e(x+l, y+m, o) f_i(l,m,o) + b_i \tag{2.4}$$

para todos os filtros i, onde $1 \leq x \leq w$, $1 \leq y \leq h$, $1 \leq i \leq n$ e bias b_i.

A Figura 2.2 ilustra um exemplo do funcionamento da camada convolucional ao receber um volume de entrada V_e e um conjunto de filtros F. Nota-se

que a saída desta camada é um volume $V_s \in \Re^{w_s,h_s,n}$, com n sendo a quantidade de filtros.

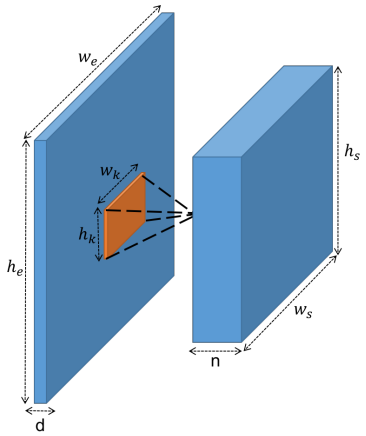


Figura 2.2: Neste exemplo, um volume de entrada $V_e^{w,h,d}$ é convoluído com um conjunto de filtros F. Nota-se que a saída desta convolução é um volume $V_s \in \Re^{w_s,h_s,n}$, com n sendo a quantidade de filtros.

A camada convolucional possui três parâmetros principais que controlam o tamanho do volume da saída: profundidade (quantidade de filtros), *stride* e *zero-padding*:

- Profundidade: corresponde ao número de filtros n usados na convolução. Cada filtro é ativado para diferentes características do volume de entrada. Por exemplo, a primeira camada convolucional recebe uma imagem colorida como entrada e os diferentes filtros são ativados na presença de bordas em diferentes orientações.

- *Stride*: ao realizar a convolução, o filtro deve ser deslizado por toda a imagem. O *stride* indica o deslocamento a cada iteração, o passo em que o x e o y da Equação 2.4 são incrementados. Por exemplo, *stride* igual a 1 indica que o filtro é deslocado pixel a pixel na convolução. Para o *stride* igual a 2, o filtro é deslocado dois pixels por iteração. Neste caso, o volume de saída é menor espacialmente. A Figura 2.3 apresenta um filtro representado em laranja e o mesmo filtro deslocado por um determinado *stride*.

- *Zero-padding*: tem como objetivo controlar o tamanho espacial do volume de saída, ou seja, determina se o volume de entrada deve ser preenchido usando um esquema de preenchimento padrão ao redor da borda. É comum usarmos o preenchimento das bordas de forma que o V_e e V_s tenham o mesmo tamanho espacial.

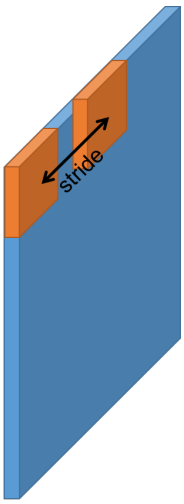


Figura 2.3: Neste exemplo, um filtro representado em laranja é deslocado por um determinado *stride*.

2.2.2 Camada ReLU

A Camada ReLU (*Rectified Linear unity*) geralmente ocorre após a camada convolucional e tem como principal objetivo permitir uma convergência mais rápida no treinamento da rede neural convolucional mantendo o poder de generalização (Krizhevsky et al., 2012). Para isso, aplica-se uma função não linear ao volume de entrada conforme

$$V_s = \max(0, V_e). \tag{2.5}$$

A Camada ReLU inclui propriedades não lineares sem afetar a resposta da camada anterior. Existem outras funções não lineares que podem ser utilizadas, como a tangente hiperbólica $V_s = \tanh(V_e)$ e sigmoide $V_s = (1 + \exp^{-V_e})^{-1}$, entretanto a rapidez com que estas funções convergem é menor do que a ReLU (Krizhevsky et al., 2012).

2.2.3 Camada Pooling

Uma função de *pooling* substitui as respostas de uma determinada região do volume de entrada V_e por uma estatística no volume de saída V_s. Esta região geralmente é um retângulo de tamanho (p,q) que é deslocado por toda a imagem, executando então uma operação estatística a cada deslocamento. Existem várias operações estatísticas que são utilizadas nesta camada, como máximo, média e média ponderada. A operação mais utilizada é o máximo, que substitui todos os valores contidos no retângulo, pelo maior valor conforme a

Equação 2.6.

$$V_s(x,y,o) = \max_{l=0}^{p-1} \max_{m=0}^{q-1} V_e(x+p, y+q, o) \tag{2.6}$$

A camada de *pooling* tem como principais vantagens reduzir a quantidade de parâmetros, controlar o sobreajuste (*overfitting*) e incluir invariância à translação. Esta camada requer basicamente dois parâmetros: o tamanho do retângulo e o *stride*. A Figura 2.4 apresenta um exemplo da operação de máximo com um retângulo de tamanho 2×2 e *stride* 2. Neste exemplo, os valores do volume de entrada são {18, 32, 46, 16} e {98, 54, 67, 12} para dois retângulos respectivamente. Na primeira iteração, indicada pelo retângulo em laranja, a maior resposta é 46. Na segunda iteração, representada em verde, a maior resposta é 98. Este procedimento se repete ao longo de todo o volume de entrada. Considerando este exemplo, nota-se que 75% das ativações do volume de entrada são descartadas.

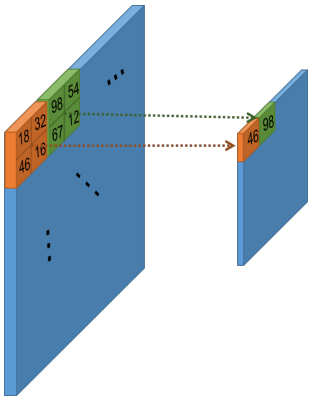


Figura 2.4: Exemplo operação de pooling em dois retângulos representado em laranja e verde, respectivamente. Ambos os retângulos possuem tamanho 2×2 e o *stride* utilizado no deslocamento é 2.

2.2.4 Camada Totalmente Conectada

A camada totalmente conectada é um conjunto de neurônios onde cada neurônio da entrada está conectado com todos os neurônios da saída. Esta camada é considerada como parte da etapa de classificação, pois é responsável por traçar um caminho de decisão a partir das respostas dos filtros das camadas anteriores. A resposta dos neurônios de saída é calculada de acordo com a Equação 2.7.

$$V_s(i) = \sum_{j=1}^{w_e} f(i,j) V_e(j) + b_i \tag{2.7}$$

onde w_e é a quantidade de neurônios da entrada, $1 \leq i \leq w_s$ com w_s sendo a quantidade de neurônios da saída e $f(i,j)$ é o peso da conexão entre o neurônio

i e o neurônio j. Essa camada é muito utilizada nas redes neurais tradicionais.

A Figura 2.5 ilustra o exemplo de uma camada totalmente conectada.

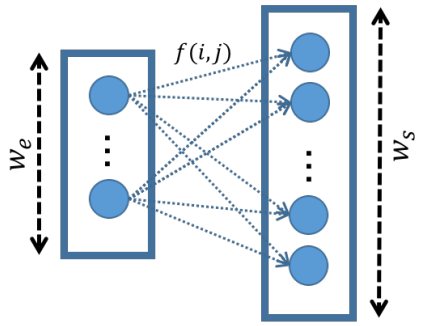


Figura 2.5: Exemplo da arquitetura de uma camada totalmente conectada. Note que w_e é a quantidade de neurônios da entrada, $f(i,j)$ é o peso da conexão entre o neurônio i e o neurônio j e w_s é a quantidade de neurônios da saída.

2.2.5 Softmax

A camada softmax é, em geral, a última etapa de classificação de uma rede neural convolucional. O número de neurônios da saída é igual a quantidade de classes que o problema possuí. O softmax é uma equação de ativação que produz estimativas positivas e tem como objetivo definir a probabilidade do volume de entrada ser de cada classe do problema. A Equação 2.8 apresenta o cálculo das probabilidades. Nota-se que esta equação é uma normalização, ou seja, a soma de todas as saídas é igual a 1.

$$p(i) = \frac{\exp^{V_e(i)}}{\sum_j \exp^{V_e(j)}}, \tag{2.8}$$

onde V_e é o volume de entrada e $p(i)$ é a probabilidade de V_e ser da classe i.

Cada neurônio $p(1), p(2), \ldots, p(n)$ da camada softmax possui a probabilidade do volume de entrada ser da classe $1, 2, \ldots, n$, respectivamente, onde n é a quantidade de classes do problema. Como exposto anteriormente, a soma total destes neurônios é igual a 1.

2.3 Arquiteturas

Nesta seção são descritas as principais arquiteturas propostas nos últimos anos. Essas arquiteturas são construídas através da composição das camadas descritas anteriormente. O número de camadas de cada arquitetura, em geral, corresponde ao número de camadas cujos parâmetros precisam ser aprendidos (e.g., camada convolucional e camada totalmente conectada). As camadas ReLU, pooling e softmax somente aplicam funções sobre os volumes sem a necessidade de aprendizado.

2.3.1 Visual Geometry Group Network (VGGNet)

VGGNet é uma arquitetura proposta por Simonyan and Zisserman (2014). Esta rede participou da ILSVRC do ano de 2014, sendo a vice-campeã daquele ano. Uma das suas maiores contribuições foi mostrar que a profundidade da CNN pode ser um parâmetro crítico em termos de desempenho, sendo que quanto maior a profundidade, melhor é a sua acurácia. Um dos problemas desta CNN é o seu uso de memória e parâmetros (VGGNet com 16 camadas e 19 camadas possuem 138 e 144 milhões de parâmetros que devem ser apreendidos, respectivamente), o que a deixa extremamente custosa. Os autores apresentaram duas arquiteturas, uma com 16 camadas e outra com 19 camadas. Na VGGNet, após cada camada de convolução e as camadas totalmente conectadas, aplica-se ReLU.

VGGNet16

A Figura 2.6 ilustra a arquitetura da VGGNet16. Note que após a AlexNet ter sido projetada com camadas de convoluções seguidas, a VGGNet também foi projetada desta forma, duas vezes com duas camadas de convoluções seguidas {(Conv 1 e Conv 2), (Conv 3 e Conv 4)} e três vezes com três camadas de convoluções seguidas {(Conv 5, Conv 6 e Conv 7), (Conv 8, Conv 9 e Conv 10), (Conv 11, Conv 12 e Conv 13)}. Note também que, após cada camada de convolução aplica-se o ReLU.

As duas primeiras camadas de convolução Conv 1 e Conv 2 possuem 64 filtros de tamanho 3×3. Uma informação importante é que em todas as camadas convolucionais desta rede, o *stride* e o *padding* são sempre 1. As camadas de convolução Conv 3 e Conv 4 possuem 128 filtros de tamanho 3×3. As camadas Conv 5, Conv 6 e Conv 7 utilizam 256 filtros de tamanho 3×3 para a convolução. Por fim, as camadas Conv 8, Conv 9, Conv 10, Conv 11, Conv 12 e Conv 13 utilizam 512 filtros de tamanho 3x3. A VGGNet16 utiliza 5 camadas de *pooling*, todas com retângulo 2×2 e o *stride* 2. As três camadas totalmente conectadas desta rede tem a mesma arquitetura da AlexNet, sendo duas com 4096 neurônios e uma com 1000 neurônios. Da mesma forma é a camada softmax, que contem 1000 neurônios, pelo mesmo motivo da rede ter sido projetada para a ILSVRC.

VGGNet19

A VGGNet de 19 camadas segue a mesma arquitetura da VGGNet de 16 camadas, sendo a única diferença a inclusão de 3 camadas convolucionais. Estas inclusões ocorreram após a camada de convolução Conv 7, Conv 10 e Conv 13 da VGGNet16, respectivamente. A Figura 2.7 ilustra as inclusões

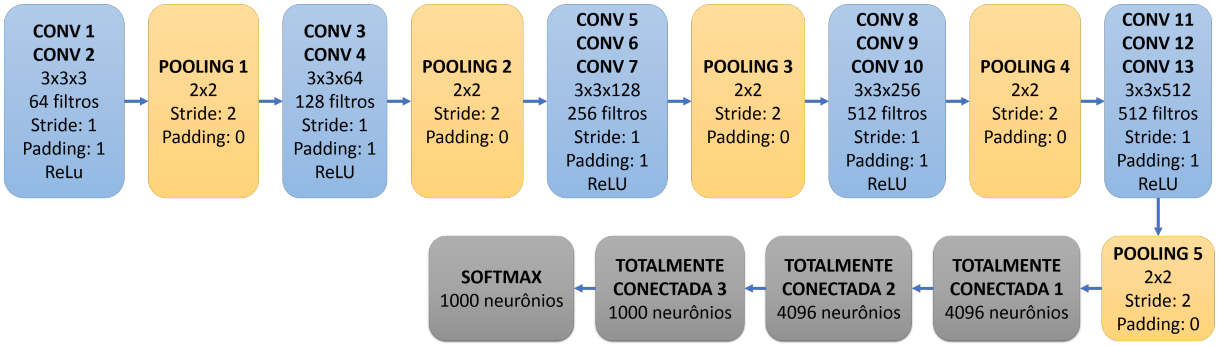


Figura 2.6: Desenho da arquitetura da VGGNet de 16 camadas. Algo interessantes sobre sua arquitetura é que, após a AlexNet ter sido projetada com camadas de convoluções seguidas, a VGGNet também foi projetada desta forma.

das 3 novas camadas, aumentando para 16 o número total de camadas de convolução. Assim como a VGGNet16, aplica-se ReLU após cada camada de convolução.

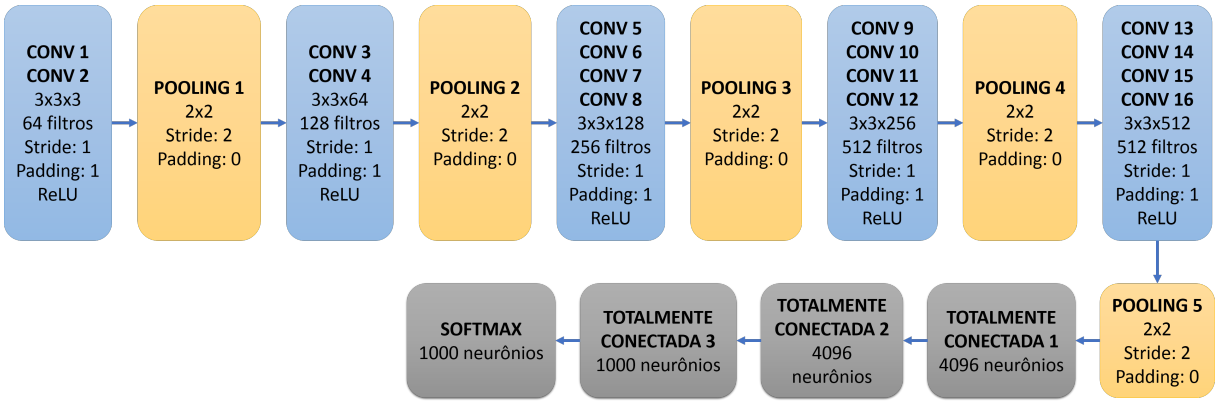


Figura 2.7: Desenho da arquitetura da VGGNet de 19 camadas. Note que, com a inclusão de 3 novas camadas convolucionais, o número total de camadas de convoluções é 16.

2.3.2 ResNet

A Resnet é uma rede neural convolucional proposta por Ren et al. (2015) e foi a vencedora da ILSVRC de 2015. Esta arquitetura propôs o aprendizado residual para treinar CNNs cada vez mais profundas. Os autores forneceram experimentos usando 50, 101 e 152 camadas, que são substancialmente mais profundas do que as arquiteturas descritas acima. Há somente 2 camadas de *pooling*, uma de *pooling* máximo após a primeira camada de convolução e uma de *pooling* médio na antepenúltima camada da rede.

Como descrito anteriormente, o aprendizado residual foi proposto nesta rede. Esta reformulação na arquitetura é motivada pelo fenômeno conhecido como degradação, onde o aprendizado das camadas mais profundas acaba

ficando saturado. Por isso, os autores tiveram a ideia de somar ponto a ponto a entrada de uma camada convolucional à saída de outra usando a Equação 2.9. Em geral, a cada duas camadas de convolução utiliza-se o aprendizado residual. A forma que este aprendizado ocorre está ilustrado na Figura 2.9. Existem alguns problemas que podem ocorrer, como diferenças de tamanho nas matrizes da soma, então os autores propõem duas soluções: i) usar um zero *padding* até as duas matrizes estiverem da mesma dimensão (esta solução não acrescenta parâmetros na rede); ii) aprender uma matriz com pesos que projete as matrizes para a mesma dimensão.

$$V_s = F(V_e) + V_e, \tag{2.9}$$

onde V_e e V_s são a entrada e a saída da camada residual considerada, F(V_e) é o volume de saída após a aplicação das duas camadas de convolução.

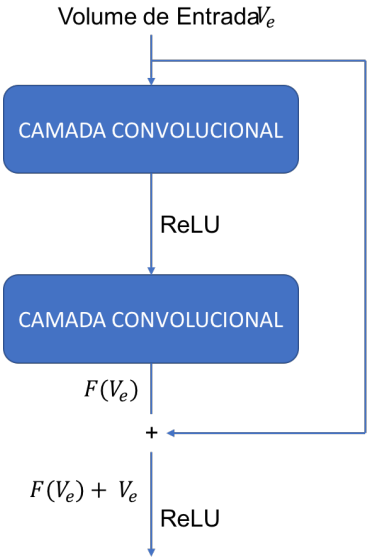


Figura 2.8: Ilustração do aprendizado residual da ResNet. Note que a soma das matrizes ocorre após duas camadas de convolução.

ResNet50

A Figura 2.9 ilustra a arquitetura da ResNet50, mostrando o padrão das camadas da rede. Note que, como ficaria inviável a exposição de todas as camadas na Figura, dividiu-se as camadas convolucionais em 4 blocos em cinza que possui um padrão de repetição. A quantidade de vezes que cada bloco se repete está na parte de cima do bloco em questão. Note pela Figura que aplica-se ReLU após cada camada de convolução e a camada totalmente conectada. A ResNet50 possui uma dimensão do volume de entrada fixo no tamanho de $224 \times 224 \times 3$. Após receber o volume de entrada, aplica-se uma convolução com 64 filtros de tamanho 7×7 e *stride* 2. Depois da primeira

camada de convolução, aplicam-se um *pooling* máximo com retângulos 2×2 e stride 2. Então aplica-se 48 camadas de convoluções que foram divididas em 4 blocos de diferentes quantidades de filtros que possuem tamanhos variados, mas que seguem um padrão. As últimas camadas são uma camada de *pooling* médio, uma camada totalmente conectada com 1000 neurônios seguidas pela camada softmax, também com 1000 neurônios.

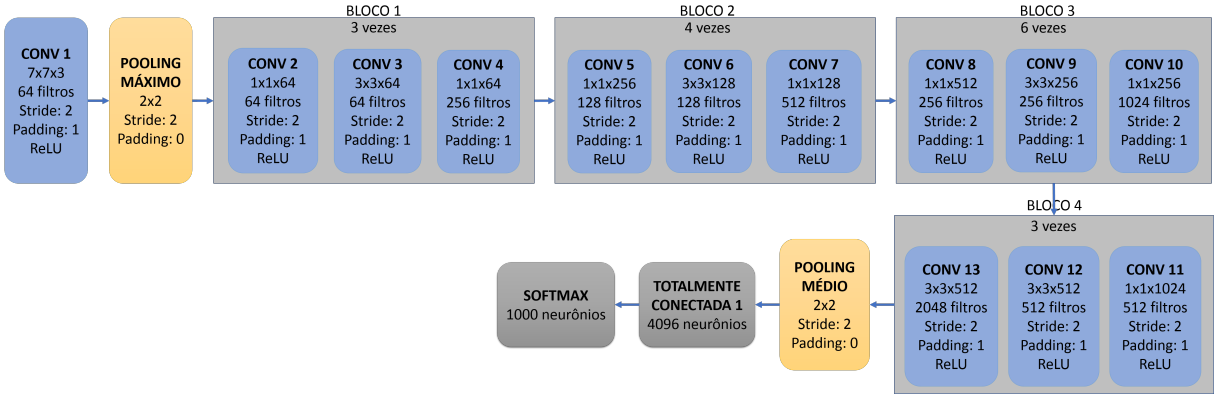


Figura 2.9: Ilustração da arquitetura da ResNet50. Note que as camadas foram divididas em 4 blocos em cinza que possui um padrão. Em cima de cada bloco está a quantidade de vezes que se repete o padrão que o bloco em questão é repetido.

CAPÍTULO

3

Algoritmos de Segmentação

3.1 Superpixel + CNN

O Superpixel+CNN proposto por Ferreira et al. (2017) utiliza os superpixels gerados pelo SLIC no treinamento de uma CNN. Inicialmente as imagens de treinamento são divididas em superpixels usando um algoritmo, por exemplo o SLIC. Em seguida uma CNN é treinada usando como entrada os superpixels, diferentemente dos trabalhos da literatura, em que as CNNs são treinadas com uma imagem inteira. Dessa forma, a CNN é capaz de classificar partes da imagem (superpixels) com o intuito de segmentação. Portanto, a última etapa do algoritmo é classificar os superpixels de uma nova imagem para segmentá-la.

3.1.1 Segmentação das Imagens em Superpixels

Dado um conjunto de n imagens de treinamento $(I_1, \ldots, I_n)$, cada imagem é dividida em aproximadamente k superpixels usando o algoritmo SLIC. Dessa forma, um conjunto de superpixels $S^{I_i} = \{S_1^{I_i}, \ldots, S_k^{I_i}\}$ é obtido para cada imagem I_i. Os superpixels de todas as imagens são concatenados em um único conjunto $S = [S^{I_1}, \ldots, S^{I_n}]$ para criação do banco de imagens que será utilizado no treinamento da CNN. Esta etapa pode ser visualizada na Figura 3.1, em que na primeira coluna é executado o algoritmo SLIC enquanto que na segunda coluna os superpixels de todas as imagens são concatenados.

Em seguida, cada superpixel é rotulado manualmente com as classes do problema. Um superpixel pode conter pixels pertencentes a duas ou mais classes do problema. Nesse caso, o superpixel é rotulado com a classe visu-

almente dominante, isto é, a classe que possui a maior quantidade de pixels. Portanto, cada superpixel $S_j^{I_i}$ possui um rótulo $r_j^{I_i}$ ao final dessa etapa, conforme ilustrado na última coluna da Figura 3.1.

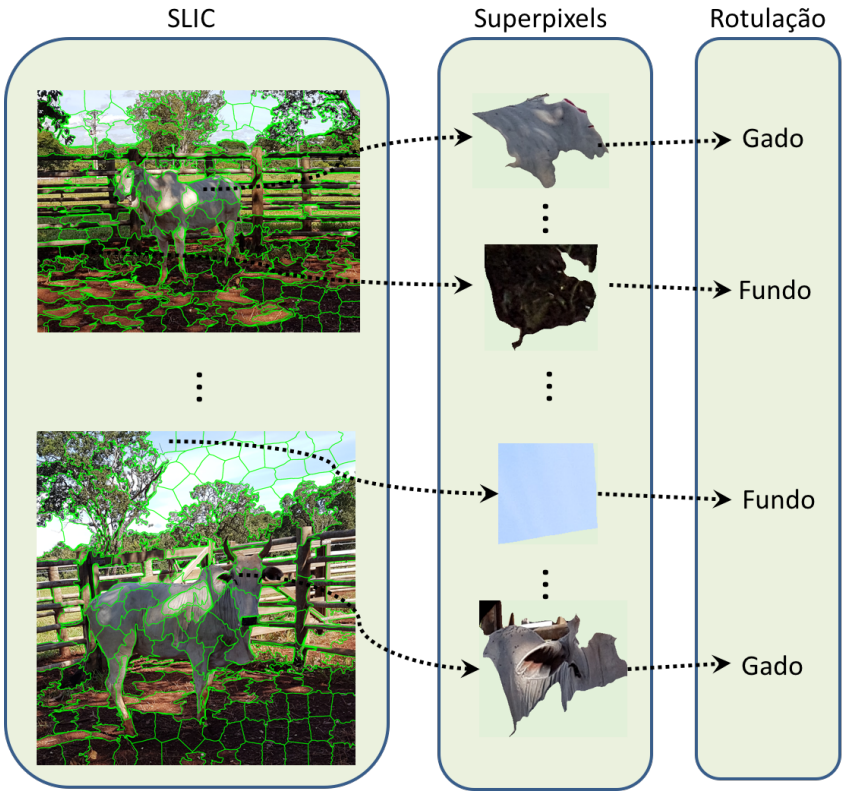


Figura 3.1: Exemplo da criação e rotulação do banco de superpixels. Na primeira coluna o algoritmo SLIC é aplicado em cada uma das imagens de treinamento de forma que o banco de superpixels seja formado, conforme a segunda coluna. A terceira coluna ilustra a rotulação dos superpixels com as classes do problema (fundo e bovino).

3.1.2 Treinamento da CNN

Nesta etapa, uma CNN é treinada com o banco de superpixels gerado na etapa anterior. Como os superpixels estão em tamanhos diferentes, cada superpixel é redimensionado para um tamanho fixo que corresponde aos requerimentos de entrada de cada arquitetura das redes neurais convolucionais (e.g., 256×256 neste trabalho).

A arquitetura das redes neurais convolucionais contém milhões de parâmetros, tornando inviável e problemático o treinamento em uma base com poucas imagens ou com pesos inicializados aleatoriamente. Para contornar esse problema, Oquab et al. (2014) propuseram a transferência de aprendizagem através do uso das camadas internas de CNN treinadas em grandes bases de imagens. Transferência de aprendizagem é o processo de utilizar uma CNN pré-treinada em um problema genérico, retirar as últimas camadas

que consistem em camadas totalmente conectadas e o classificador *softmax*, e treinar estas últimas camadas para resolver um problema específico. A utilização de camadas convolucionais de uma CNN pré-treinada permite que a CNN extraia características mais gerais das imagens, como bordas, enquanto que as camadas totalmente conectadas que serão treinadas possam utilizar estas características para classificar imagens para o problema específico.

Para usar a transferência de aprendizagem neste trabalho, os pesos das camadas convolucionais são inicializados com os pesos de CNNs treinadas na *Imagenet* ILSVRC 2012 com 1.28 milhões de imagens divididas em 1000 classes Russakovsky et al. (2015). Por outro lado, três camadas totalmente conectadas e o *softmax* são inicializados aleatoriamente. As camadas totalmente conectadas possuem 1024 neurônios enquanto que o softmax possui neurônios correspondentes ao número de classes do problema específico. A Figura 3.2 ilustra a transferência de aprendizagem da arquitetura VGG16. Durante o processo de treinamento, os pesos das camadas convolucionais inicializadas foram atualizados, ou seja, foram aprimorados para as bases de imagens abordadas neste trabalho.

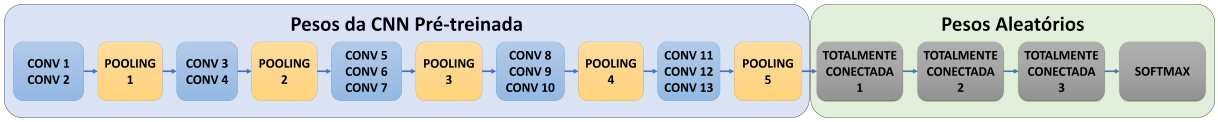


Figura 3.2: Exemplo de transferência de aprendizagem da arquitetura VGG16. As camadas convolucionais são inicializadas com um CNN pré-treinada, enquanto que as camadas totalmente conectadas são inicializadas com pesos aleatórios.

Após a construção da CNN como descrito acima, é usado o *backpropagation* para atualizar os pesos das camadas totalmente conectadas e das convolucionais. Os principais parâmetros do treinamento são: taxa de aprendizado, tamanho do lote de imagens e quantidade de épocas. Os parâmetros são descritos abaixo:

- **Taxa de aprendizado**: é uma constante entre zero e um $[0,1]$ que indica a velocidade de aprendizado;
- **Tamanho do lote de exemplos**: indica a quantidade de exemplos (imagens) utilizados no treinamento a cada iteração;
- **Época**: corresponde a uma passagem completa através de todo o banco de exemplos;

3.1.3 Classificação de Superpixels

Nesta etapa, ocorre a segmentação de uma imagem I usando a CNN treinada na etapa anterior. Para isso, são obtidos os superpixels $S^I = \{S^I_1, \ldots, S^I_k\}$

da imagem I usando o SLIC. Em seguida, cada superpixel S_j^I é classificado usando a CNN. Por fim para segmentar a imagem, cada pixel é classificado com a mesma classe do seu superpixel correspondente.

3.2 SegNet

A SegNet é uma rede neural convolucional para segmentação proposta por Badrinarayanan et al. (2017). A arquitetura desta CNN é dividida em duas partes conforme a Figura 3.3: 1) codificador que é usado para extrair as características da imagem de entrada, e 2) decodificador que reconstrói a imagem segmentada a partir das características extraídas. No trabalho original, o codificador (extrator de característica) da SegNet é idêntico topologicamente as camadas convolucionais da VGG16.

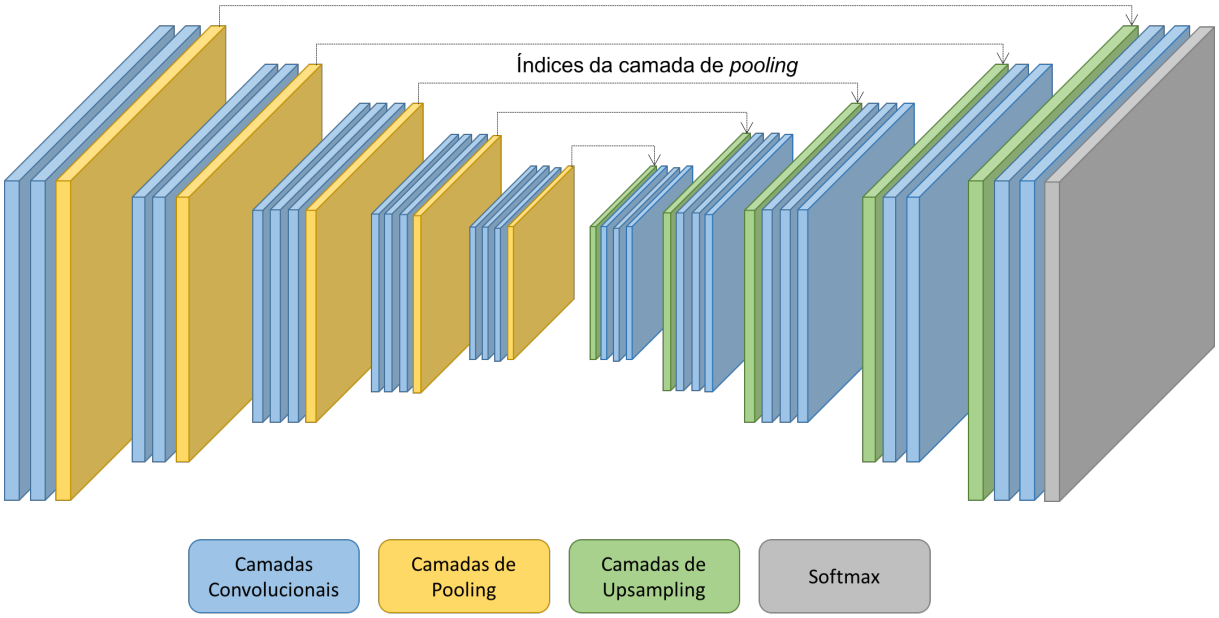


Figura 3.3: Exemplo da SegNet com duas partes: codificador e decodificador. O codificador é composto por camadas convolucionais e pooling. Por outro lado, o decodificador é composto por camadas convolucionais, upsampling e softmax.

3.2.1 Codificador

O codificador é composto por camadas convolucionais e *pooling* assim como as CNNs propostas para classificação, portanto ele pode ser implementado com base em qualquer arquitetura já proposta, como a VGG16 (utilizada no trabalho original), VGG19 e ResNet50. A camada convolucional executa uma convolução com filtros para produzir um conjunto de mapas de características. Em seguida, aplica-se uma normalização por lote e uma função ReLU nos mapas de características. Depois a camada de *pooling* máximo é aplicada com

uma janela 2×2 e *stride* 2, portanto sem sobreposição das janelas, reduzindo o mapa de características pela metade. Este processo pode ser visualizado na primeira parte da Figura 3.3. Ao final do codificador, o mapa de características é de baixa resolução, embora contenha informação que descreve o contexto de toda a imagem de entrada.

A diferença para as outras CNNs está no *pooling*, isto porque a SegNet precisa armazenar algumas informações para utilizar no decodificador. Os autores propuseram o armazenamento dos índices do *pooling* máximo, ou seja, a posição dos maiores valores de pixels. Note que isso pode ser armazenado usando apenas dois bits, isto porque o *pooling* máximo é 2×2 na maioria das arquiteturas.

3.2.2 Decodificador

O decodificador tem como objetivo reconstruir a imagem segmentada utilizando o mapa de característica de baixa resolução obtido do codificador (segunda parte da Figura 3.3). Para isso o decodificador possui camadas de *upsampling* que executam o inverso da função de *pooling*, isto é, aumentam a resolução do mapa de características. Para aumentar a resolução, a camada *upsampling* utiliza os índices da camada de *pooling* correspondente, por exemplo, a última camada de *pooling* do codificador é correspondente a primeira camada de *upsampling* do decodificador.

A Figura 3.4 ilustra este processo da camada *upsampling*. Na esquerda é ilustrada a camada de *pooling* do codificador e na direita a sua camada *upsampling* correspondente no decodificador. A camada de *pooling* ocorre normalmente, entretanto armazenam-se os valores máximos e os índices do *pooling*. Na camada *upsampling* estão o mapa de característica e os índices armazenados da camada de *pooling*, que são utilizados para reconstruir o mapa de característica de maior resolução. Note que, devido ao fato de somente os índices e os valores máximos do *pooling* serem armazenados, as outras posições do mapa de características são preenchidas com zeros.

Após a camada de *upsampling*, são executadas camadas de convolução com filtros no mapa de características resultante do *upsampling*, para produzir um mapa de características denso. Estes filtros tem como objetivo aprender os valores dos pixels zeros. Após diversas camadas do decodificador, o último mapa de características possui resolução igual a imagem de entrada. Por fim, a camada *softmax* é aplicada para classificar cada pixel da imagem no conjunto de classes do problema.

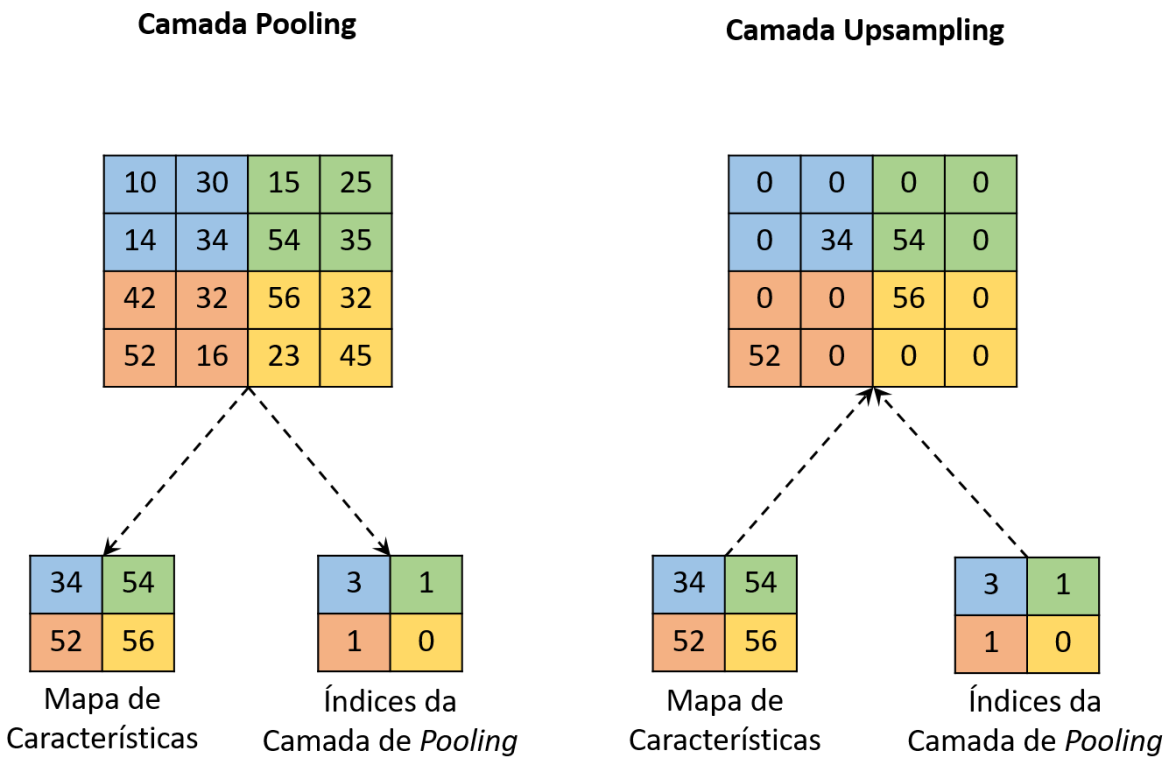


Figura 3.4: Exemplo da camada de *pooling* e *upsampling*. A camada de *pooling* calcula o mapa de características e os índices (inteiro pré-definido para cada posição). Por outro lado, a camada de *upsampling* recebe o mapa de características de baixa resolução e os índices para construir um mapa de ativação de alta resolução.

3.2.3 Arquiteturas da SegNet

Neste trabalho, as SegNets implementadas tiveram como base a VGG16, VGG19 e ResNet50 para o codificador. No decodificador, as camadas de *upsampling* e de convolução foram correspondentes às camadas de *pooling* e de convolução do codificador. A Figura 3.5 ilustra a arquiteturas da SegNet (VGG16). Na imagem podemos notar dois retângulos, no retângulo superior está a parte do codificador e no inferior está a parte do decodificador. No codificador, a arquitetura da SegNet (VGG16) é idêntica a CNN (VGG16), mudando somente no decodificador, que possui camadas de *upsampling* e de convolução de forma inversa as camadas do codificador. As outras arquiteturas, SegNet (VGG19) e SegNet (ResNet50), podem ser obtidas da mesma forma.

A aprendizagem da SegNet é supervisionada por meio da função de perda entropia cruzada. Como a saída deste algoritmo é a imagem segmentada, a função de perda é aplicada em cada pixel. Portanto, o treinamento da SegNet requer a imagem anotada por um especialista.

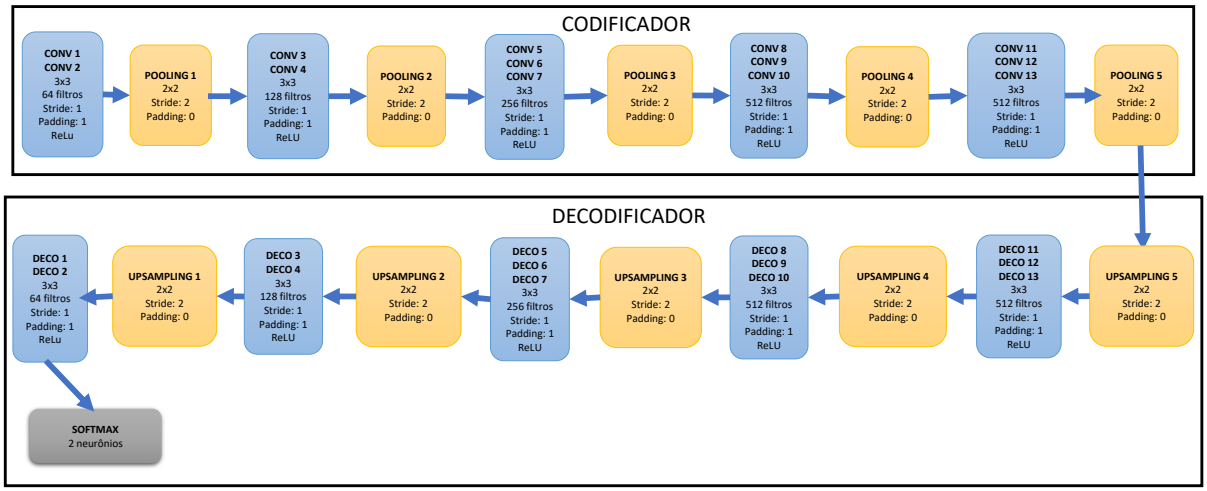


Figura 3.5: Exemplo da arquitetura da SegNet (VGG16). O retângulo superior corresponde a parte do codificador e o inferior a parte do decodificador.

CAPÍTULO

4

Método Proposto

Os resultados mostraram que ambos algoritmos, Superpixel+CNN e SegNet, possuem vantagens e desvantagens que se complementam. Portanto, este trabalho propõe a combinação dos dois algoritmos por meio de operadores lógicos e matemáticos. Com isso, pretende-se incrementar os resultados através da combinação das vantagens dos dois algoritmos.

Neste capítulo são descritas as combinações entre Superpixel+CNN e SegNet propostas neste trabalho. Para realizar essas combinações, aplica-se uma operação na saída dos dois algoritmos (Superpixel+CNN e SegNet) para gerar uma nova saída. Para isso, considere $M_{SCNN}(x,y,0)$ como a probabilidade do pixel (x,y) pertencer ao objeto de interesse e $M_{SCNN}(x,y,1)$ a probabilidade de pertencer ao fundo obtido como saída do Superpixel+CNN. Similarmente para a SegNet temos que $M_{SegNet}(x,y,0)$ é a probabilidade do pixel (x,y) pertencer ao objeto de interesse e $M_{SegNet}(x,y,1)$ a probabilidade de pertencer ao fundo.

Dadas as probabilidades, as operações de cada combinação são descritas nas seções abaixo.

4.1 MEAN

Esse operador calcula uma nova probabilidade para o objeto de interesse e fundo com base na média das probabilidades retornadas pelo Superpixel+CNN e SegNet. A probabilidade do objeto de interesse $p(x,y,0)$ e fundo $p(x,y,1)$ podem ser visualizadas nas Equações 4.1 e 4.2, respectivamente. O intuito dessa combinação é levar em consideração as certezas e incertezas de cada algoritmo de forma ponderada. Por fim, um pixel (x,y) é classificado com objeto de interesse se $p(x,y,0) > p(x,y,1)$ ou como fundo caso contrário. A Figura 4.1

apresenta um exemplo da combinação MEAN. A primeira linha mostra as probabilidades do objeto de interesse e fundo para o Superpixel+CNN seguida pelo resultado da segmentação de cada pixel (símbolos O e F representam objeto de interesse e fundo, respectivamente). A segunda linha da figura apresenta as probabilidades e resultado da segmentação para a SegNet. Por fim, a terceira linha apresenta o resultado da combinação das probabilidades por meio da média. Podemos perceber que o resultado da segmentação da combinação difere tanto do Superpixel+CNN como da SegNet. Por exemplo, o primeiro pixel foi classificado como fundo pelo Superpixel+CNN e como objeto de interesse pela SegNet. A combinação também classificou esse pixel como fundo, pois a probabilidade de ser fundo em $M_{SCNN}(x,y,1)$ é alta.

$$p(x,y,0) = \frac{M_{SCNN}(x,y,0) + M_{SegNet}(x,y,0)}{2} \tag{4.1}$$

$$p(x,y,1) = \frac{M_{SCNN}(x,y,1) + M_{SegNet}(x,y,1)}{2} \tag{4.2}$$

4.2 MULT

Para esta combinação, as probabilidades dos dois algoritmos são multiplicadas conforme as Equações 4.3 e 4.4. Esta combinação foi inspirada na probabilidade conjunta considerando os dois algoritmos independentes. Esta probabilidade é utilizada para observar resultados simultâneos de múltiplas variáveis, que neste caso são as probabilidades dos dois algoritmos. Dessa forma, essa combinação estima a probabilidade de ocorrência simultânea.

$$p(x,y,0) = M_{SCNN}(x,y,0) * M_{SegNet}(x,y,0) \tag{4.3}$$

$$p(x,y,1) = M_{SCNN}(x,y,1) * M_{SegNet}(x,y,1) \tag{4.4}$$

4.3 MAX

A combinação MAX utiliza o valor máximo entre as probabilidades dadas pelos algoritmos conforme as Equações 4.5 e 4.6. A ideia principal dessa combinação é considerar um algoritmo que forneceu um probabilidade alta (grande chance de ocorrência), mesmo que o outro algoritmo tenha probabilidade baixa. Por exemplo, considere que os dois algoritmos forneçam probabilidades de 0,95 e 0,2 para a classe carcaça (0,05 e 0,8 para a classe fundo), respectivamente. Nesse caso, a probabilidade da classe carcaça usando a combinação MAX é 0,95, mesmo que o segundo algoritmo tenha uma proba-

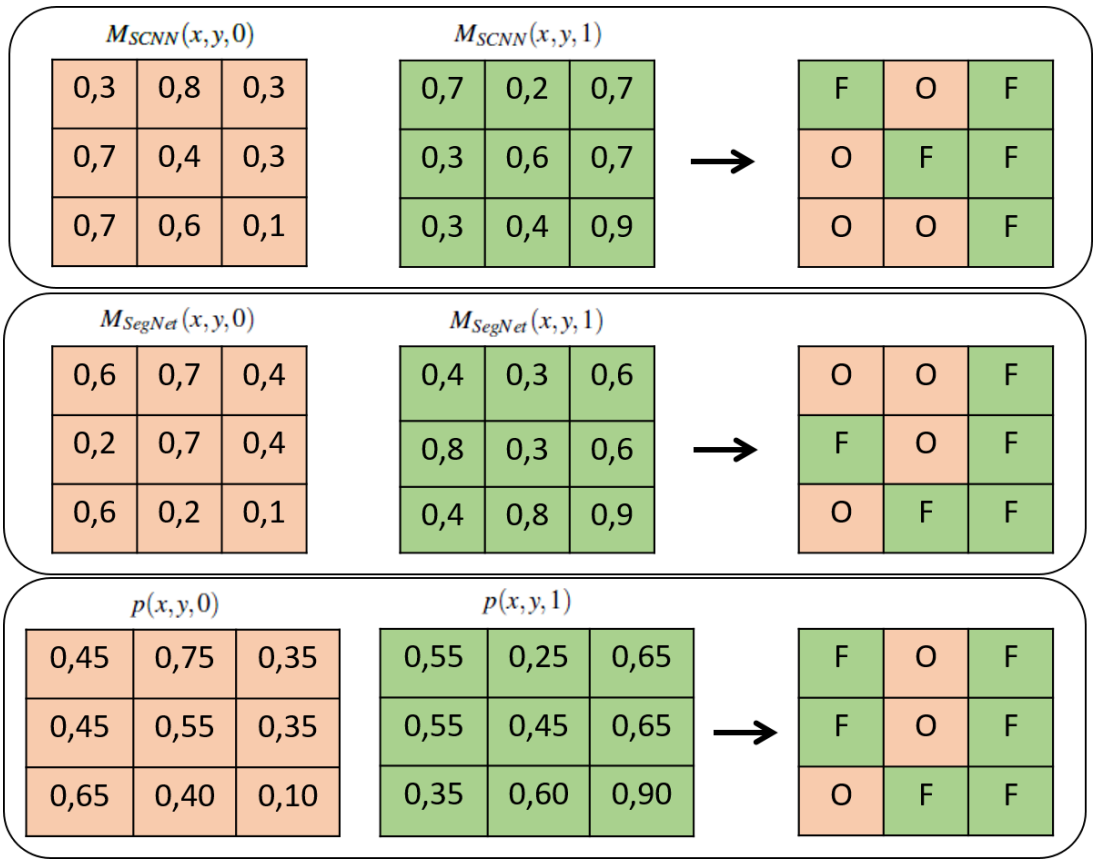


Figura 4.1: Exemplo da combinação MEAN onde $M_{SCNN}(x,y,0)$ corresponde a probabilidade do pixel (x,y) pertencer ao objeto de interesse, $M_{SCNN}(x,y,1)$ a probabilidade de pertencer ao fundo e o resultado da segmentação do Superpixel+CNN (**F**: fundo e **O**: objeto de interesse). Da mesma forma $M_{SegNet}(x,y,0)$ e $M_{SegNet}(x,y,1)$ correspondem a SegNet. Já o $p(x,y,0)$ corresponde a média de $M_{SCNN}(x,y,0)$ e $M_{SegNet}(x,y,0)$ e $p(x,y,1)$ a média de $M_{SCNN}(x,y,1)$ e $M_{SegNet}(x,y,1)$. O resultado da segmentação da combinação difere tanto do Superpixel+CNN como da SegNet, considerando as probabilidades médias.

bilidade baixa. Portanto, essa combinação desconsidera a probabilidade do segundo algoritmo, o que não ocorre com as combinações anteriores, como MEAN e MULT.

$$p(x,y,0) = \max(M_{SCNN}(x,y,0), M_{SegNet}(x,y,0)) \tag{4.5}$$

$$p(x,y,1) = \max(M_{SCNN}(x,y,1), M_{SegNet}(x,y,1)) \tag{4.6}$$

4.4 OR

Esta combinação utiliza o operador lógico OR no resultado da segmentação do Superpixel+CNN e SegNet. Considere a Tabela 4.1 como a operação OR, onde O é quando um pixel foi classificado como objeto de interesse e F quando é classificado como fundo. Portanto, essa combinação classifica um pixel como

objeto de interesse se pelo menos um dos algoritmos também o classificar. Um pixel é classificado como fundo se e somente se os dois algoritmos também o classificarem como fundo.

Tabela 4.1: Exemplo da combinação OR que foi implementada neste trabalho, onde *O* representa um pixel classificado como objeto de interesse e *F* é quando um pixel é classificado como fundo.

Superpixel+CNN	**SegNet**	**OR**
O	O	O
O	F	O
F	O	O
F	F	F

4.5 AND

Esta combinação é similar a combinação OR, sendo que neste caso é utilizado o operador lógico AND. Um pixel é classificado como objeto de interesse se e somente se os dois algoritmos o classificam assim. A Tabela 4.2 mostra o operador lógico AND que é aplicado em cada pixel para obter a segmentação.

Tabela 4.2: Exemplo do operador lógico AND implementado, onde *O* representa representa um pixel classificado como objeto de interesse e *F* é quando um pixel é classificado como fundo.

Superpixel+CNN	**SegNet**	**AND**
O	O	O
O	F	F
F	O	F
F	F	F

CAPÍTULO

5

Materiais e Métodos

Este capítulo apresenta os materiais e métodos que incluem a base de imagens de carcaças, delineamento experimental e as métricas de avaliação.

5.1 Base de Imagens

Nesta seção são descritas as bases de imagens que foram utilizadas nos experimentos para avaliar algoritmos de segmentação em problemas de agropecuária de precisão.

5.1.1 Carcaça

Esta base tem como objetivo avaliar algoritmos de segmentação de carcaças de bovinos. Para a captura de imagens, as carcaças foram obtidas após o abate, sangria, esfolação e retirada da cabeça, patas, rabo, glândulas mamárias (fêmea), testículos (macho) e verga, exceto suas raízes. A segmentação de carcaça em imagens é um dos primeiros passos para a construção de sistemas automáticos para a tipificação de carcaças. O processo de tipificação consiste em dividir a carcaça em classes de qualidade, sendo os principais critérios para esta divisão sua cobertura de gordura e conformação. Esta tipificação é importante porque permite uma definição da classe da carcaça, afetando o seu valor para o mercado interno e para o mercado externo, que só aceita carcaças de classes nobres.

A base de carcaças possui 226 imagens com dimensão 3264×2448 que foram capturadas em um frigorífico da região de Mato Grosso do Sul[1] utilizando um

[1] O nome do frigorífico foi omitido por questões legais.



Figura 5.1: Exemplos de quatro imagens presentes na base de imagens utilizadas nos experimentos. Partes da imagem estão borrada devido a questões legais.

celular Samsung Galaxy S7. As imagens possuem diferentes tipos de escala, iluminação e ângulos. A Figura 5.1 apresenta 4 exemplos em que podemos observar que a captura foi em um ambiente interno mas não controlado. As imagens possuem diferentes escalas, iluminação e interferências como a presença de funcionários, o que torna o desafio de segmentação ainda maior.

Para avaliar os resultados de segmentação, todas as imagens do banco foram anotadas manualmente, dividindo as imagens em duas classes, fundo e carcaça. A Figura 5.2 ilustra exemplos das imagens anotadas, sendo que a primeira coluna contem as imagens originais e a segunda contem as imagens anotadas. A anotação das imagens foi realizada utilizando o software *Labelme*[2].

[2]https://github.com/wkentaro/labelme

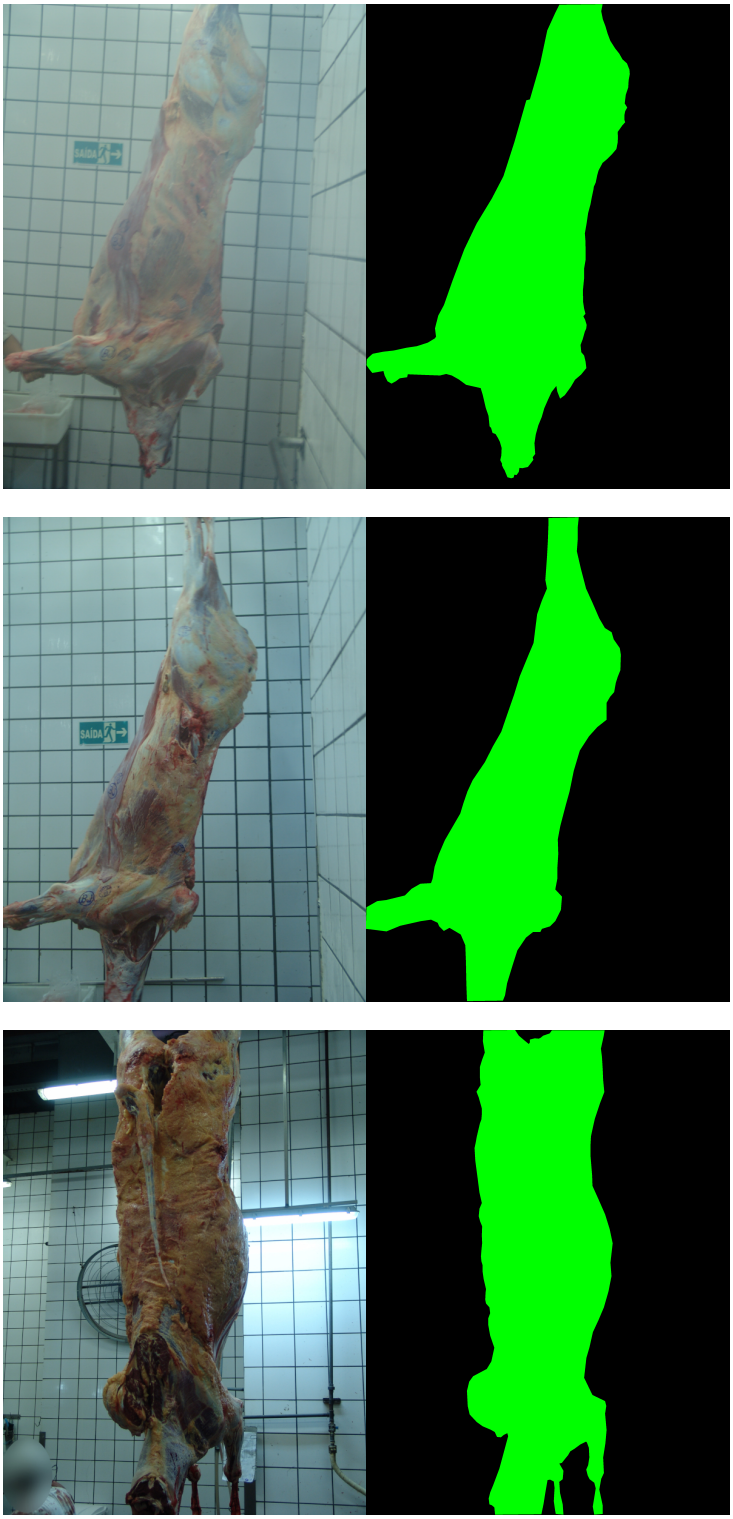


Figura 5.2: Exemplos de três imagens anotadas manualmente para realizar os experimentos. A primeira coluna apresenta as imagens originais e a segunda apresenta as imagens anotadas. Os pixels em verde pertencem a classe carcaça enquanto que os pixels em preto pertencem ao fundo. O rosto e algumas partes da imagem estão borradas por questões legais.



Figura 5.3: Exemplos de quatro imagens presentes na base de imagens utilizadas nos experimentos.

5.1.2 Boi em pé

Esta base de imagens tem como o objetivo avaliar algoritmos de segmentação de bovinos. A segmentação desses animais é o primeiro passo para a criação de sistemas automáticos que, por exemplo, estimar o peso do animal. A Base é composta por 154 imagens com dimensão de 4032 por 3024 pixels com diferentes escalas, iluminações e ângulos, o que a torna muito complexa. A Figura 5.3 apresenta 4 exemplos que demonstram a complexidade desta base.

Assim como na base carcaça, todas as imagens foram anotadas manualmente com o software *Labelme* para a avaliação dos resultados. A Figura 5.4 mostra exemplos de 2 imagens anotadas com o software, onde a primeira coluna corresponde as imagens originais e a segunda as anotadas.

5.2 Delineamento Experimental

Para os experimentos, as imagens foram divididas aleatoriamente em conjuntos de treinamento, validação e teste. Para isso, 60% das imagens foram utilizadas no treinamento, 20% na validação e o restante no teste. É impor-



Figura 5.4: Exemplos de duas imagens anotadas manualmente para realizar os experimentos. A primeira coluna apresenta as imagens originais e a segunda apresenta as imagens anotadas. Os pixels em verde pertencem a classe boi enquanto que os pixels em preto pertencem ao fundo.

tante mencionar que nenhum pré-processamento além do redimensionamento foi aplicado nas imagens. Os conjuntos de treinamento e validação foram utilizados para treinar e obter os melhores parâmetros de cada algoritmo de segmentação. Por outro lado, o conjunto de teste foi utilizado para avaliar a melhor configuração dos algoritmos.

Os parâmetros avaliados de cada algoritmo de segmentação são descritos a seguir:

- **Superpixel + CNN:** Este algoritmo combina segmentação por superpixel seguida pela classificação usando CNN. Na segmentação por superpixel, a quantidade de superpixels foi avaliada entre 100, 500 e 1000. Na classificação com CNN, avaliou-se as arquiteturas VGG16, VGG19 e ResNet50 devido aos seus resultados promissores reportados na literatura. Os pesos das camadas foram inicializados com CNNs treinadas na base de imagens ImageNet, que contém 1.2 milhões de imagens com 1000 classes. No treinamento, após teste empíricos no conjunto de treinamento e validação, usou-se taxa de aprendizado 0.001 para a ResNet e 0.0001 para as VGGs, 50 épocas e tamanho do *batch* igual a 32.

- **SegNet:** O codificador geralmente é baseado em uma arquitetura já proposta na literatura e, no caso deste trabalho, foi baseado em três arquiteturas: VGG16, VGG19 e ResNet50. Estas arquiteturas foram treinadas na base de imagens ImageNet, que contém aproximadamente 1.2 milhões de imagens com 1000 classes. Por outro lado, o decodificador foi inicializado com pesos aleatórios, sendo atualizados posteriormente pelo processo de treinamento. Os parâmetros usados no treinamento foram: 0.01 de taxa de aprendizado; 150 épocas; tamanho do *batch* igual a 16, após testes empíricos no conjunto de treinamento e validação.

5.3 Métricas de Avaliação

Para monitorar o processo de treinamento utilizou-se a função de perda *cross-entropy* e acurácia. Para avaliar o resultado da segmentação, foram utilizadas as métricas bem conhecidas na literatura (Shelhamer et al., 2017; Badrinarayanan et al., 2017): acurácia pixel-a-pixel e intersecção sobre a união por classe. Essas métricas podem ser calculadas através da matriz de confusão, que é uma matriz que permite a visualização do desempenho de um algoritmo. Para os problemas deste trabalho (duas classes), esta matriz é binária onde as linhas e colunas representam as duas classes do problema, objeto de interesse e fundo. As colunas representam as classes preditas pelo algoritmo e as linhas representam as classes verdadeiras. A Figura 5.5 representa a

matriz de confusão para os problemas deste trabalho. A posição Verdadeiro Positivo (VP) representa o número de pixels do objeto de interesse que foram corretamente preditos. O Verdadeiro Negativo (VN) representa o número de pixels que foram corretamente classificados como fundo. Portanto, VP e VN representam os acertos do algoritmo. Por outro lado, a posição Falso Positivo (FP) representa o número de pixels que foram incorretamente preditos como objeto de interesse, enquanto que o Falso Negativo (FN) representa o número de pixels incorretamente classificados como fundo.

		Classe Predita	
		Objeto de Interesse	Fundo
Classe Verdadeira	Objeto de Interesse	VP	FP
	Fundo	FN	VN

Figura 5.5: Exemplo da Matriz de Confusão para os problemas deste trabalho.

As métricas acurácia pixel-a-pixel e intersecção sobre a união são descritas a seguir:

- **Acurácia pixel-a-pixel**: essa métrica calcula a porcentagem de acertos de pixels para cada classe. Usando a matriz de confusão, as acurácias do objeto de interesse (AP_O) e do fundo (AP_F) podem ser calculadas por:

$$AP_O = \frac{VP}{VP+FN} \tag{5.1}$$

$$AP_F = \frac{VN}{VN+FP} \tag{5.2}$$

- **Intersecção sobre a União (IoU)**: para a classe de objeto de interesse O, considere $G_O = \{(x_0,y_0),\ldots,(x_n,y_n)\}$ como o conjunto de pixels da imagem anotada e $P_O = \{(x_0,y_0),\ldots,(x_m,y_m)\}$ como o conjunto de pixels da imagem predita que pertencem ao objeto de interesse. Essa medida calcula a interseção dos conjuntos $G_O \cap P_O$ indicando o número de acertos sobre a união $G_O \cup P_O$:

$$IoU = \frac{|\, G_O \cap P_O \,|}{|\, G_O \cup P_O \,|} \tag{5.3}$$

  A Figura 5.6 apresenta a interpretação dessa medida. Quanto maior a área da intersecção entre G_O e P_O, maior é o número de acertos e, consequentemente, o valor da medida.

$$IoU = \frac{G_O \cap P_O}{G_O \cup P_O} = \frac{}{}$$

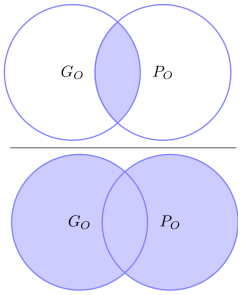


Figura 5.6: Interpretação da medida Intersecção sobre a União.

CAPÍTULO

6

Resultados e Discussões

Este capítulo apresenta os resultados, suas discussões e a comparação dos algoritmos de segmentação em problemas de agropecuária de precisão.

6.1 Treinamento das CNNs

Esta seção descreve os resultados obtidos durante o treinamento das CNNs para os algoritmos Superpixel+CNN e SegNet. Para todos os algoritmos, são analisadas a função de perda e as acurácias no conjunto de treinamento e validação (o conjunto de teste é utilizado para compararar a melhor configuração de cada algoritmo). Para Superpixel+CNN, a acurácia corresponde a porcentagem de superpixels corretamente classificados pela CNN. Por outro lado para a SegNet, a acurácia corresponde a porcentagem de pixels corretamente classificados. Essa análise é importante para validar o treinamento das CNNs e verificar questões como sobreajuste (*overfitting*).

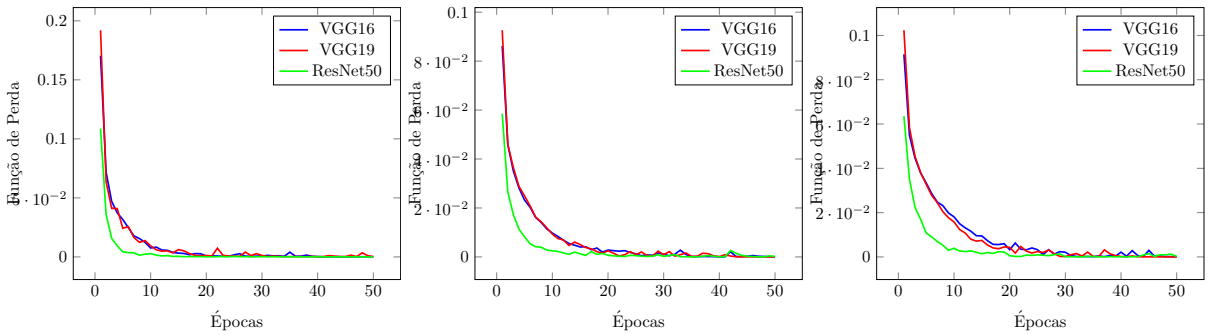
6.1.1 Superpixel + CNN

A função de perda do treinamento do Superpixel+CNN para 50 épocas e com k igual a 100, 500 e 1000 para as duas base de imagens, carcaça e boi em pé, pode ser visualizada na Figura 6.1. As Figuras 6.1a, 6.1b e 6.1c apresentam os resultados na base carcaça e as Figuras 6.1d, 6.1e e 6.1f apresentam para a base boi em pé.

Após as 50 épocas, todas as arquiteturas obtiveram um valor de perda baixo, mostrando que as CNNs foram treinadas adequadamente para as duas bases de imagens. Além disso, podemos perceber que as curvas caem rapida-

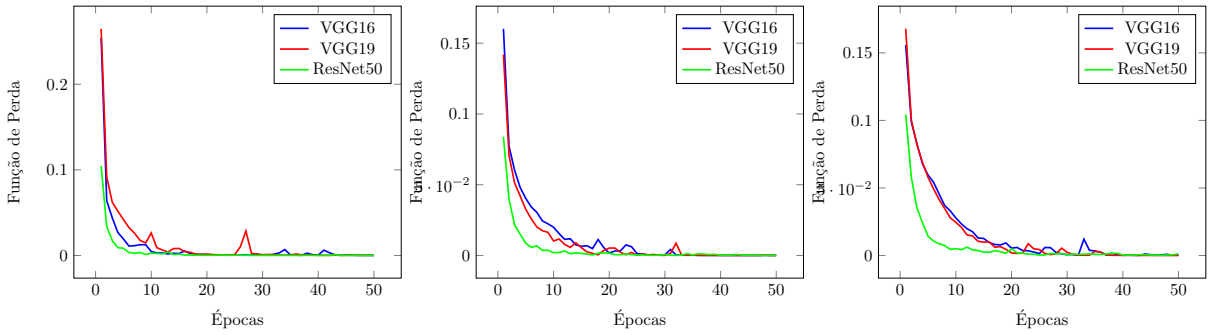
mente e, após poucas épocas, as CNNs já estabilizam a sua função de perda, com a ResNet50 obtendo uma queda mais acentuada. Com relação as arquiteturas, as três obtiveram resultados satisfatórios que demonstraram um grande potencial.

Carcaça



(a) k = 100 (b) k = 500 (c) k = 1000

Boi em pé



(d) k = 100 (e) k = 500 (f) k = 1000

Figura 6.1: Função de perda para o treinamento das CNNs por 50 épocas usando diferentes valores de k para as duas bases de imagens.

A Tabela 6.1 apresenta as acurácias no conjunto de treinamento e validação usando diferentes valores de k e arquiteturas para as duas bases de imagens. Todas as arquiteturas obtiveram acurácia de 100% no conjunto de treinamento e no conjunto de validação resultados maiores que 98%. Portanto, podemos observar que a redução da acurácia entre os dois conjuntos foi baixa, o que indica que não houve sobreajuste (*overfitting*).

Podemos concluir que os melhores resultados são obtidos com o $k = 500$, sendo a ResNet50 e a VGG19 as melhores arquiteturas.

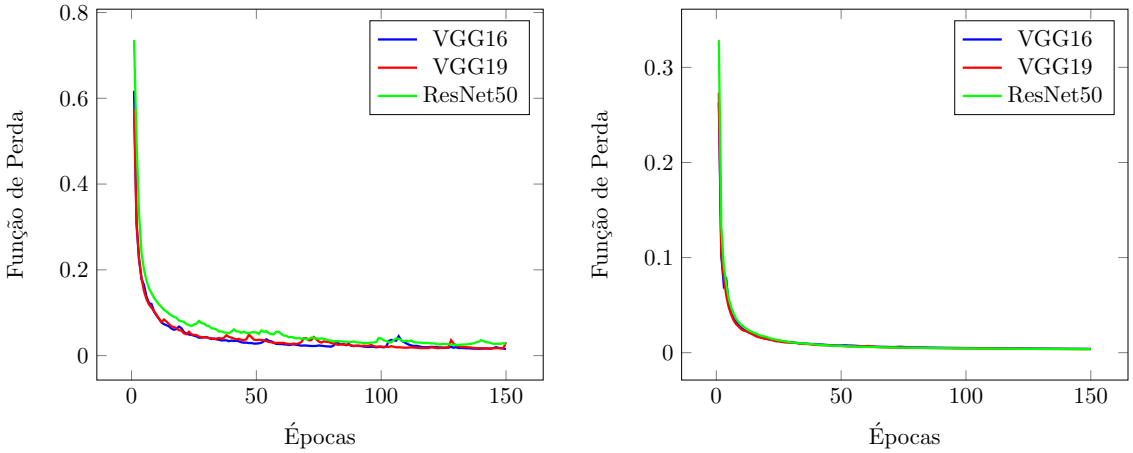
6.1.2 SegNet

A Figura 6.2 mostra a função de perda da SegNet com diferentes arquiteturas para as duas bases de imagens. Todas as arquiteturas obtiveram um valor de perda baixo após as 150 épocas, mostrando que foram treinadas ade-

Tabela 6.1: Acurácia no conjunto de treinamento e validação do algoritmo Superpixel+CNN com diferentes configurações (arquiteturas, número de superpixels) para as duas bases de imagens. A acurácia corresponde ao número de superpixels corretamente classificados pela CNN.

k	**Arquitetura**	**Carcaça**		**Boi em pé**	
		Treinamento	**Validação**	**Treinamento**	**Validação**
100	VGG16	**1,0**	0,985	**1,0**	**0,990**
	VGG19	**1,0**	0,984	**1,0**	0,988
	ResNet50	**1,0**	**0,987**	**1,0**	**0,990**
500	VGG16	**1,0**	0,992	**1,0**	**0,989**
	VGG19	**1,0**	**0,993**	**1,0**	0,987
	ResNet50	**1,0**	**0,993**	**1,0**	**0,989**
1000	VGG16	**1,0**	0,990	**1,0**	0,980
	VGG19	**1,0**	**0,991**	**1,0**	0,980
	ResNet50	**1,0**	0,990	**1,0**	**0,984**

quadamente. Pode-se observar também que as curvas caem rapidamente e, após poucas épocas, suas funções de perda já estão estabilizadas.



(a) Carcaça (b) Boi em pé

Figura 6.2: Função de perda para o treinamento da SegNet com VGG16, VGG19 e ResNet50 para as duas bases de imagens e por 150 épocas.

As acurácias da SegNet no conjunto de treinamento e validação para as duas bases de imagens são mostradas na Tabela 6.2. É importante lembrar que a acurácia desse algoritmo corresponde à porcentagem de pixels corretamente classificados. A SegNet com diferentes arquiteturas obtiveram resultados similares, com pequena vantagem para a VGG16 na base de imagens da carcaça e um resultado igual entre a VGG16 e VGG19 na base de imagens boi em pé. Podemos notar também que não há uma grande diferença entre as acurácias do conjunto de treinamento e validação, demonstrando assim que o algoritmo de segmentação alcançou uma boa generalização.

Tabela 6.2: Acurácia no conjunto de treinamento e validação da SegNet com a VGG16, VGG19 e ResNet50 para as duas bases de imagens. A acurácia corresponde ao número de pixels corretamente classificados pela SegNet.

Arquitetura	**Carcaça**		**Boi em pé**	
	Treinamento	**Validação**	**Treinamento**	**Validação**
SegNet (VGG16)	**0.996**	**0,983**	**1,0**	**0,990**
SegNet (VGG19)	0,989	0,975	**1,0**	**0,990**
SegNet (ResNet)	0,991	0,975	**1,0**	0,987

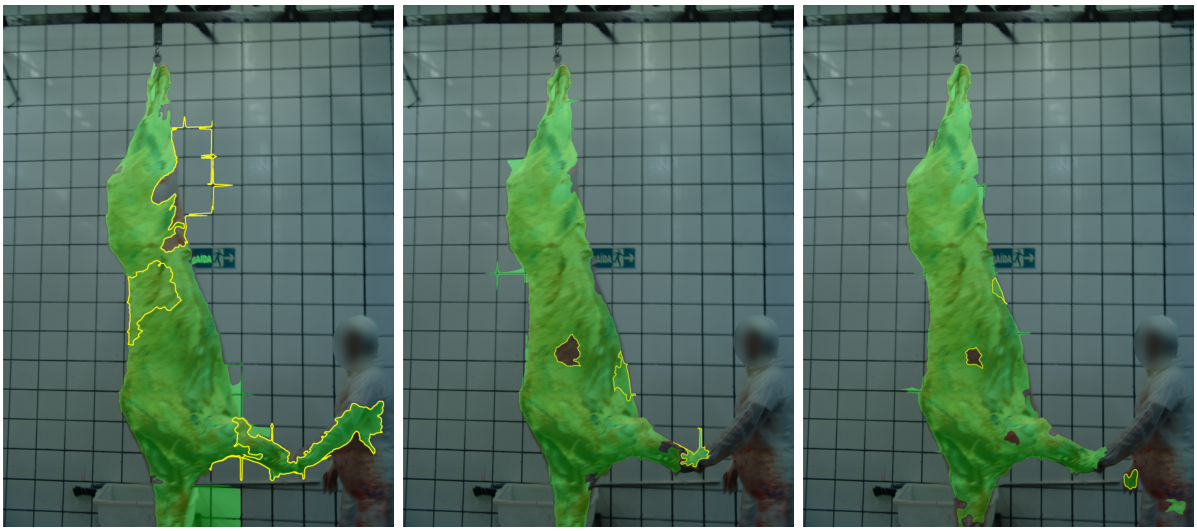
6.2 Comparação entre Algoritmos de Segmentação

Essa seção apresenta uma comparação entre os algoritmos usando métricas para avaliar algoritmos de segmentação para as duas bases de imagens. A Tabela 6.3 apresenta os resultados entre os algoritmos de segmentação utilizando a acurácia pixel-a-pixel e a Intersecção sobre a União (IoU) no conjunto de teste composto por 45 imagens da base de imagem carcaça. Os resultados mostrados na tabela são da classe do objeto de interesse (carcaça), desconsiderando a classe de fundo.

Considerando o Superpixel+CNN, as três arquiteturas obtiveram valores similares para ambas as métricas. Por outro lado, o número de superpixel k apresentou uma grande influência nos resultados das métricas, entre um k pequeno ($k = 100$) e um k grande ($k = 500$ e $k = 1000$). Comparando a acurácia pixel-a-pixel, o melhor resultado para $k = 100$ e 1000 foi de 0,843 e 0,961, respectivamente. Comparando a Intersecção sobre União, nota-se que a diferença entre os resultados é ainda maior, sendo os melhores resultados de 0,773 e 0,922 para $k = 100$ e 1000, respectivamente. Esse resultado se deve ao fato de que quanto maior o número de superpixels k, melhor delineadas são as bordas da imagem e, consequentemente, o resultado da segmentação (veja Figura 2.1 do Capítulo 2). Entretanto, comparando $k = 500$ e 1000, nota-se que não há uma grande diferença de acurácia e IoU. Isto porque quanto maior o k, o superpixel tende a ser pequeno e portanto não possui informação suficiente para descreve-lo.

A Figura 6.3 ilustra as considerações acima e destaca três superpixels extraídos com $k = 100, 500$ e 1000 (os demais superpixels foram omitidos para melhor visualização). Para $k = 100$ (Figura 6.3a), os superpixels são maiores e com mais contexto, o que facilitaria a sua classificação. Por outro lado, para $k = 500$ e 1000 (Figuras 6.3b e 6.3c), os superpixels são menores, fazendo com que as bordas sejam bem delineadas, embora com menos contexto, fazendo com que alguns superpixels sejam incorretamente classificados.

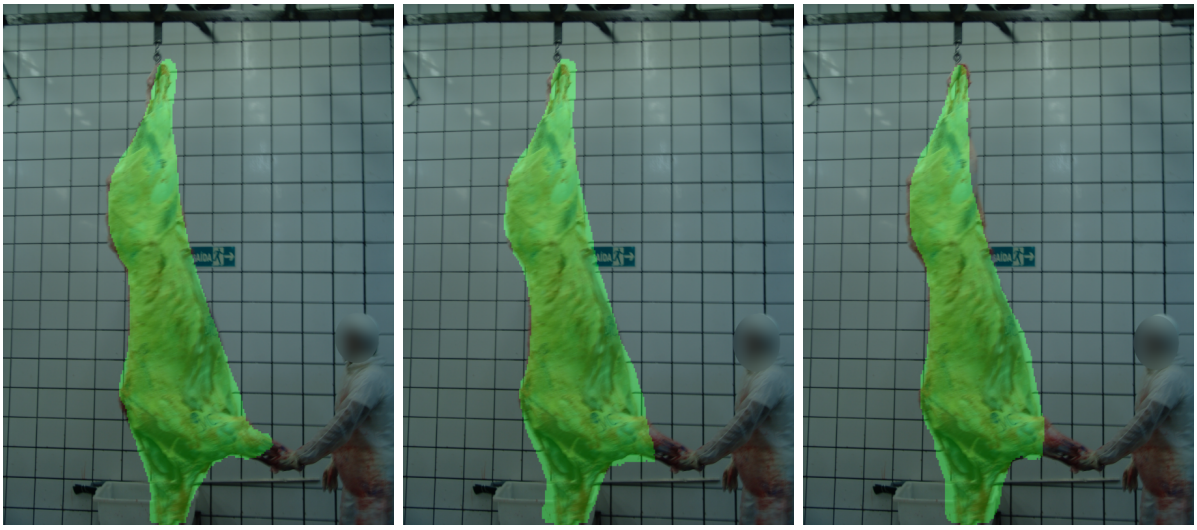
Para os resultados da SegNet, a arquitetura VGG19 obteve melhor resultado na acurácia pixel-a-pixel se comparado com as outras arquiteturas, com



(a) k = 100 (b) k = 500 (c) k = 1000

Figura 6.3: Resultados da classificação de superpixels obtidos com $k = 100, 500$ e 1000 para o algoritmo Superpixel+CNN (VGG16). Os rostos estão borrados por questões legais.

0,972. Por outro lado, na Intersecção sobre a União, o resultado obtido pela VGG16 foi superior com 0,920 contra 0,886 da VGG19 e 0,897 da ResNet50. O resultado da segmentação usando VGG19 e a ResNet50 em geral é superestimado como ilustrado nas Figuras 6.4b e 6.4c, fazendo com que a acurácia pixel-a-pixel da carcaça seja alta, embora a IoU seja menor quando comparada com a VGG16 (Figura 6.4a).



(a) SegNet (VGG16) (b) SegNet (VGG19) (c) SegNet (ResNet50)

Figura 6.4: Resultados da segmentação para SegNet com VGG16, VGG19 e ResNet50. Os rostos estão borrados por questões legais.

Comparando os resultados dos algoritmos de segmentação, a SegNet (VGG19) obteve a melhor acurácia pixel-a-pixel com 0,972. Por outro lado, o algoritmo

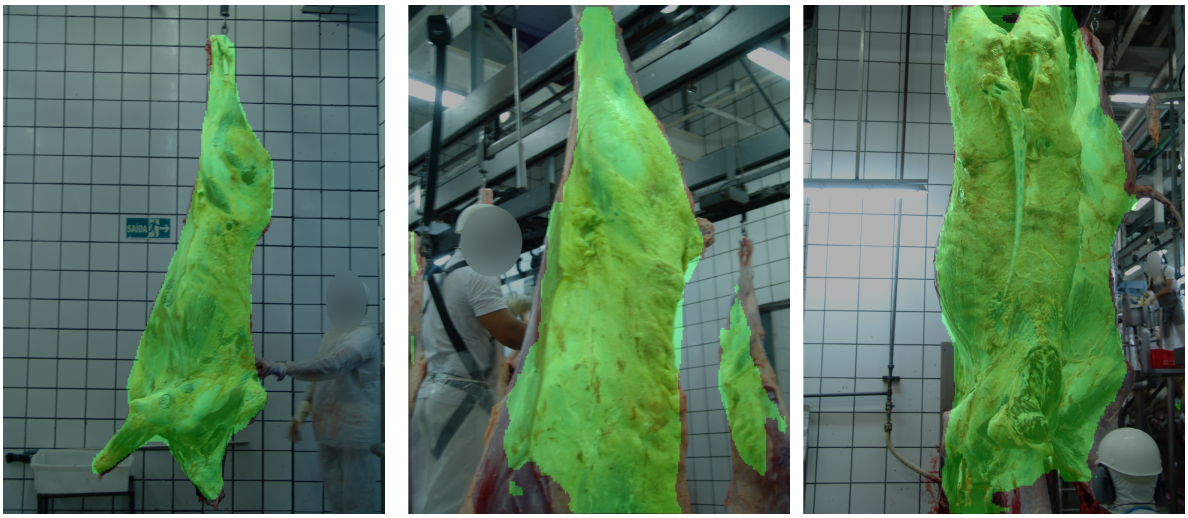
Superpixel+CNN obteve os melhores resultados para a métrico IoU, com 0,922 obtido pela VGG16 com $k = 1000$. A SegNet(VGG19) e o Superpixel+CNN(k = 1000, VGG16) obtiveram resultados melhores pois, a SegNet utiliza informações de toda a imagem para classificar um pixel, devido as camadas de convolução e *upsampling*. Já no algoritmo Superpixel+CNN com o $k = 1000$, as bordas são bem precisas, aumentando os resultados de IoU. Por outro lado, a SegNet não é tão precisa nas bordas se comparada com o Superpixel+CNN com um k grande. Porém, quanto maior o k utilizado pelo Superpixel+CNN, menor será a informação da imagem toda.

Tabela 6.3: Comparação entre os algoritmos de segmentação: Superpixel+CNN e SegNet para a base carcaça.

Algoritmo	**Acurácia pixel-a-pixel**	**IoU**
Superpixel+CNN (k = 100, VGG16)	0,836 (±0,07)	0,772 (±0,08)
Superpixel+CNN (k = 100, VGG19)	0,843 (±0,06)	0,770 (±0,08)
Superpixel+CNN (k = 100, ResNet50)	0,843 (±0,08)	0,773 (±0,08)
Superpixel+CNN (k = 500, VGG16)	0,956 (±0,01)	0,912 (±0,02)
Superpixel+CNN (k = 500, VGG19)	0,954 (±0,01)	0,910 (±0,02)
Superpixel+CNN (k = 500, ResNet50)	0,953 (±0,02)	0,911 (±0,02)
Superpixel+CNN (k = 1000, VGG16)	0,961 (±0,01)	**0,922 (±0,02)**
Superpixel+CNN (k = 1000, VGG19)	0,961 (±0,01)	0,921 (±0,02)
Superpixel+CNN (k = 1000, ResNet50)	0,957 (±0,01)	0,917 (±0,02)
SegNet (VGG16)	0,956 (±0,02)	0,920 (±0,02)
SegNet (VGG19)	**0,972 (±0,02)**	0,886 (±0,02)
SegNet (ResNet50)	0,938 (±0,02)	0,897 (±0,03)

A Figura 6.5 apresenta uma comparação qualitativa da SegNet e Superpixel+CNN, em que cada linha representa um algoritmo. Devido aos resultados superiores, as imagens para Superpixel+CNN correspondem ao $k = 1000$ e a melhor arquitetura. Apesar da complexidade de algumas imagens, com ângulos e escalas diferentes, além de algumas com mais de uma carcaça, os algoritmos obtiveram bons resultados qualitativos.

A Tabela 6.4 apresenta as acurácias no conjunto de testes para a base de imagens boi em pé. Também foram desconsiderados os resultados da classe fundo, mostrando os resultados apenas do objeto de interesse (boi). Os resultados mostram uma acurácia e um IoU mais baixos se comparado com a base carcaça, o que demonstra que a base boi em pé é mais complexa. Utilizando a abordagem Superpixel+CNN, também nota-se que a maior diferença nos resultados está quando o k é alterado, principalmente levando em consideração o $k = 100$, que obteve um resultado bem inferior tanto para a acurácia pixel-a-pixel quanto para o IoU se comparado com o $k = 500$ e $k = 1000$. Nesta base de imagens houve uma maior diferença entre os resultados das arquiteturas, sendo a VGG16 superior quando o $k = 100$ e a ResNet50 com o k igual a 500 e



(a) SegNet (VGG16)



(b) Superpixel+CNN (VGG16)

Figura 6.5: Resultados comparativos da SegNet e Superpixel+CNN com as melhores arquiteturas. Os resultados apresentados da Superpixel+CNN são para $k = 1000$. Os rostos e partes da imagem foram borrados por questões legais.

1000. Na comparação geral, o melhor resultado para a acurácia pixel-a-pixel foi obtido pela Superpixel+CNN (ResNet50, $k = 500$) com 0,822. Para o IoU o melhor resultado foi obtido pela ResNet50 e $k = 1000$ com 0,736. Para a SegNet, a melhor acurácia foi obtida pela VGG19 com 0,909. Por outro lado, o melhor IoU foi obtido pela VGG16 com 0,838. Comparando as duas abordagens, a SegNet obteve os melhores resultados se comparado com a Superpixel+CNN, sendo bem superior tanto para a acurácia quanto para o IoU.

A Figura 6.6 apresenta os resultados qualitativos desta base de imagens. As imagens da abordagem Superpixel+CNN foram obtidas com o $k = 500$, pois essa configuração obteve os melhores resultados na média. Cada coluna representa um algoritmo com a melhor arquitetura. É possível observar que a

Tabela 6.4: Comparação entre os algoritmos de segmentação: Superpixel+CNN e SegNet para a base boi em pé.

Algoritmo	**Acurácia pixel-a-pixel**	**IoU**
Superpixel+CNN (k = 100, VGG16)	0,654 (±0,17)	0,526 (±0,14)
Superpixel+CNN (k = 100, VGG19)	0,544 (±0,23)	0,467 (±0,21)
Superpixel+CNN (k = 100, ResNet50)	0,622 (±0,21)	0,508 (±0,19)
Superpixel+CNN (k = 500, VGG16)	0,799 (±0,07)	0,700 (±0,09)
Superpixel+CNN (k = 500, VGG19)	0,812 (±0,07)	0,706 (±0,08)
Superpixel+CNN (k = 500, ResNet50)	0,822 (±0,07)	0,722 (±0,08)
Superpixel+CNN (k = 1000, VGG16)	0,791 (±0,08)	0,709 (±0,09)
Superpixel+CNN (k = 1000, VGG19)	0,777 (±0,08)	0,701 (±0,08)
Superpixel+CNN (k = 1000, ResNet50)	0,821 (±0,07)	0,736 (±0,07)
SegNet (VGG16)	0,893 (±0,05)	**0,838** (±0,06)
SegNet (VGG19)	**0,909** (±0,04)	0,835 (±0,07)
SegNet (ResNet50)	0,874 (±0,04)	0,815 (±0,06)

SegNet apresenta resultados visualmente melhores, corroborando os resultados das métricas quantitativas.



(a) SegNet (VGG19)



(b) Superpixel+CNN (ResNet50)

Figura 6.6: Resultados comparativos da SegNet e Superpixel+CNN em 3 imagens da base boi em pé. Os resultados apresentados são para $k = 500$ e melhores arquiteturas.

6.3 Resultados do Método Proposto

Esta seção apresenta os resultados da combinação da SegNet com o superpixel+CNN proposta nesse trabalho. Para os resultados foram escolhidas as melhores configurações das abordagens, levando em consideração os resultados apresentados na sub-seção anterior. Para a base de imagens da carcaça, a melhor arquitetura da Superpixel+CNN e da SegNet foi a VGG16 e $k = 1000$. Para a base de imagens boi em pé, a configuração utilizada foi Superpixel+CNN (ResNet50, $k = 1000$) e SegNet (VGG19).

A Tabela 6.5 apresenta os resultados de todas as combinações da Superpixel+CNN e SegNet para a base de imagens da carcaça. A Tabela foi dividida em quatro blocos horizontais, em que o primeiro bloco corresponde aos melhores resultados do Superpixel+CNN e da SegNet, o segundo bloco corresponde às combinações usando o $k = 100$ para a Superpixel+CNN, o terceiro com $k = 500$ e o último com o $k = 1000$.

Considerando a acurácia pixel-a-pixel, a combinação OR obteve os melhores resultados para todos os valores de k, inclusive melhor que os melhores resultados obtidos pela Superpixel+CNN e SegNet separadamente. O melhor resultado geral da acurácia pixel-a-pixel foi de 0,989 obtido pela combinação OR com $k = 1000$. Para a métrica IoU os resultados entre as combinações em cada bloco são bem parecidos, em que os melhores resultados estão no bloco onde o $k = 1000$. Nessa métrica há um empate entre 3 combinações que obtiveram 0,936: MAX, MULT e MEAN, que também obtiveram melhores resultados se comparado com a Superpixel+CNN e SegNet.

A Figura 6.7 apresenta os resultados qualitativos para a SegNet, Superpixel+CNN e método proposto com combinações MEAN e OR. Essas duas combinações foram escolhidas pois forneceram bons resultados em ambas métricas e bases de imagens. Podemos observar que o método proposto refina os resultados dos dois algoritmos. A SegNet (Figura 6.7a) não fornece uma segmentação precisa nas bordas enquanto que a Superpixel+CNN (Figura 6.7b) melhora a segmentação nas bordas embora forneça falhas em algumas partes devido a falta de contexto do superpixel. Por outro lado, o método proposto melhora as deficiências conforme ilustrado na Figura 6.7d. Isso ocorre na combinação OR, pois na segmentação dos pixels das bordas é utilizado o resultado do Superpixel+CNN, que neste caso é mais preciso. Por outro lado, as falhas são resolvidas utilizando a SegNet, que possui maior contexto na segmentação.

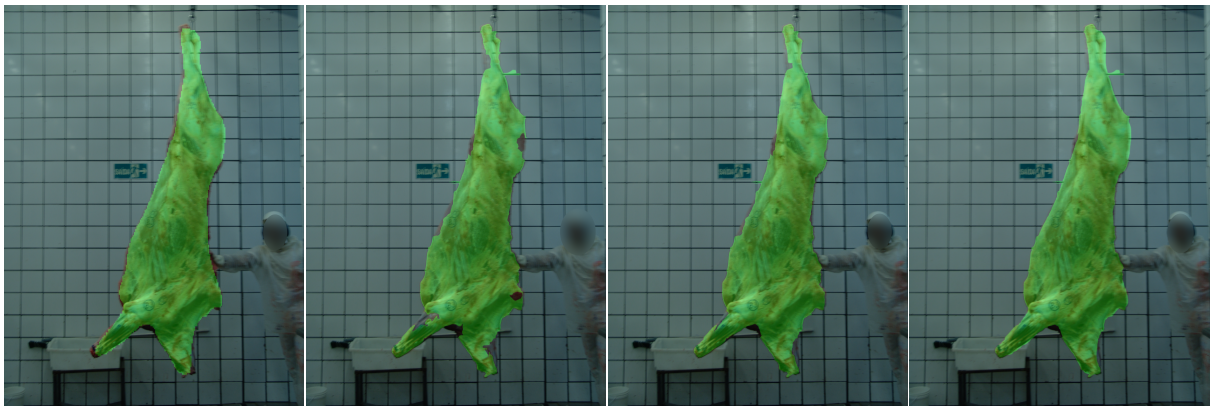
A Tabela 6.6 apresenta os resultados das combinações para a base de imagens boi em pé. Os melhores resultados mostraram o mesmo padrão que o da base da carcaça, onde a combinação OR forneceu os melhores resultados da acurácia pixel-a-pixel. Considerando essa métrica, o melhor resultado de

Tabela 6.5: Resultados das combinações das abordagens Superpixel+CNN e SegNet para a base de imagens carcaça. Foram utilizados a Superpixel+CNN (VGG16, $k = 1000$) e a SegNet (VGG16), pois foram as que obtiveram os melhores resultados nesta base.

k	**Combinação**	**Acurácia pixel-a-pixel**	**IoU**
	Superpixel+CNN (k = 1000, VGG16)	0,961 (±0,01)	0,922 (±0,02)
	SegNet (VGG16)	0,956 (±0,02)	0,920 (±0,02)
100	OR	0,982 (±0,01)	0,871 (±0,04)
	AND	0,813 (±0,07)	0,805 (±0,07)
	MAX	0,858 (±0,06)	0,793 (±0,06)
	MULT	0,858 (±0,06)	0,793 (±0,06)
	MEAN	0,854 (±0,06)	0,793 (±0,06)
500	OR	0,988 (±0,00)	0,915 (±0,02)
	AND	0,920 (±0,02)	0,911 (±0,02)
	MAX	0,960 (±0,01)	0,919 (±0,02)
	MULT	0,960 (±0,01)	0,919 (±0,02)
	MEAN	0,960 (±0,01)	0,919 (±0,02)
1000	OR	**0,989** (±0,00)	0,922 (±0,02)
	AND	0,928 (±0,02)	0,920 (±0,02)
	MAX	0,969 (±0,00)	**0,936** (±0,01)
	MULT	0,969 (±0,00)	**0,936** (±0,01)
	MEAN	0,969 (±0,00)	**0,936** (±0,01)

0,941 foi obtido com $k = 1000$. Esse resultado foi superior aos resultados obtidos pelos algoritmos separadamente. Para IoU também houve um empate entre as combinações MAX, MULT e MEAN com o $k = 1000$, obtendo 0,815. Porém, esse resultado foi inferior ao da SegNet.

A Figura 6.8 apresenta os resultados qualitativos da SegNet (VGG19), Superpixel+CNN (ResNet50, $k = 1000$) e as combinações MEAN e OR. Estas arquiteturas foram escolhidas porque obtiveram os melhores resultados na base de imagens boi em pé. Assim como observado nos resultados qualitativos das imagens de carcaça, observa-se uma melhoria nos resultados no método proposto. Este aperfeiçoamento fica evidente na parte do focinho, onde tanto a SegNet (6.8a) e a Superpixel+CNN (6.8b) forneceram resultados imprecisos. Além disso, também nota-se falhas em alguns superpixels classificado pela Superpixel+CNN, que pode ser explicado pela falta de contexto deste algoritmo. A grande maioria das imprecisões apontadas são aperfeiçoadas pelas combinações propostas, como mostra a Figura 6.8d que corresponde a combinação OR. Podemos notar que a classificação dos pixels da parte do focinho e as falhas do Superpixel+CNN foram aprimoradas por esta combinação, obtendo resultados expressivos e promissores.



(a) SegNet (b) Superpixel+CNN (c) MEAN (d) OR



(e) SegNet (f) Superpixel+CNN (g) MEAN (h) OR

Figura 6.7: Resultados comparativos da SegNet (VGG16), Superpixel+CNN (VGG16, $k = 1000$) e as combinações MEAN e OR em três imagens da base carcaça. Os rostos e partes da imagem foram borrados por questões legais.



(a) SegNet (b) Superpixel+CNN (c) MEAN (d) OR



(e) SegNet (f) Superpixel+CNN (g) MEAN (h) OR

Figura 6.8: Resultados comparativos da SegNet (VGG19), Superpixel+CNN (Resnet50, $k = 1000$) e as combinações MEAN e OR em três imagens da base boi em pé.

Tabela 6.6: Resultados das combinações das abordagens Superpixel+CNN e SegNet para a base de imagens boi em pé. Foi utilizado para os experimentos a Superpixel+CNN (Resnet50, k=1000) e a SegNet (VGG19), pois foram as que obtiveram os melhores resultados nesta base.

k	**Algoritmo**	**Acurácia pixel-a-pixel**	**IoU**
	Superpixel+CNN (k = 1000, ResNet50)	0,821 (±0,07)	0,736 (±0,07)
	SegNet (VGG19)	0,909 (±0,04)	**0,835** (±0,07)
100	OR	0,924 (±0,03)	0,774 (±0,09)
	AND	0,491 (±0,24)	0,482 (±0,24)
	MAX	0,688 (±0,15)	0,633 (±0,14)
	MULT	0,692 (±0,14)	0,638 (±0,13)
	MEAN	0,692 (±0,14)	0,508 (±0,13)
500	OR	0,939 (±0,02)	0,796 (±0,07)
	AND	0,750 (±0,10)	0,734 (±0,10)
	MAX	0,847 (±0,01)	0,779 (±0,07)
	MULT	0,848 (±0,06)	0,780 (±0,07)
	MEAN	0,848 (±0,06)	0,780 (±0,07)
1000	OR	**0,941** (±0,02)	0,799 (±0,08)
	AND	0,790 (±0,08)	0,773 (±0,08)
	MAX	0,878 (±0,04)	0,815 (±0,05)
	MULT	0,879 (±0,04)	0,815 (±0,05)
	MEAN	0,879 (±0,04)	0,815 (±0,05)

CAPÍTULO

7

Conclusão

A pecuária é uma das áreas mais importantes da economia brasileira e a inserção de tecnologias da pecuária de precisão podem incrementar a produtividade. Para a criação de tecnologias automáticas, métodos computacionais que utilizam a visão computacional estão sendo cada vez mais estudados e propostos na literatura. A segmentação é considerada uma das etapas mais importantes e complexas destes métodos, pois uma boa segmentação pode melhorar o desempenho das próximas etapas do sistema.

Neste trabalho foram avaliados três algoritmos de segmentação para imagens de carcaça e bovinos, Superpixel+CNN, SegNet e o método proposto. No Superpixel+CNN, os superpixels extraídos da imagem são classificados por uma CNN. Na SegNet, uma CNN foi incorporada para classificar cada pixel da imagem. Após estas avaliações, foi proposto um método com diversas formas de combinações entre as probabilidades dos dois algoritmos através de operadores lógicos e matemáticos. A ideia é que estas combinações recebam as vantagens de cada algoritmo, obtendo resultados superiores.

Para os experimentos, foram construídas duas bases de imagens, uma de carcaça composta por 226 imagens com dimensão 3264×2448 e outra de bovinos com 154 imagens com dimensão de 4032 por 3024. Nos algoritmos da literatura, foram avaliadas três arquiteturas de CNN: VGG16, VGG19 e ResNet50. Devido aos melhores resultados, as arquiteturas utilizadas no método proposto na base da carcaça foram a Superpixel+CNN ($k = 1000$, VGG16) e SegNet (VGG16). O melhor resultado para a métrica acurácia pixel-a-pixel foi de 98.9% obtido pela combinação OR. Para a métrica IoU, o melhor resultado foi de 93.6% obtido pelas combinações MAX, MULT e MEAN. Estes resultados foram superiores aos dos algoritmos da literatura. Já para a base de imagens

de bovinos, as arquiteturas utilizadas pelo método proposto foram a Superpixel+CNN ($k = 1000$, ResNet50) e a SegNet (VGG19). O melhor resultado para a métrica acurácia pixel-a-pixel foi obtida pela combinação OR com 94.1%, sendo superior as algoritmos da literatura. Para o IoU, o melhor resultado foi de 83.5% obtido pela SegNet (VGG19).

De acordo com os resultados, foi observado que Superpixel+CNN com um k adequado possui boa precisão nas bordas embora perca informação de contexto, classificando incorretamente superpixels da classe fundo como carcaça. Por outro lado, a SegNet apresentou boa informação de contexto na classificação dos pixels, mas perdendo precisão nas bordas. Os resultados do método proposto demonstraram que foi possível refinar os resultados do Superpixel+CNN e SegNet, obtendo as vantagens de cada algoritmo na segmentação de imagens.

Como trabalhos futuros, pretende-se propor e avaliar outros tipos de combinações, realizar experimentos para outras bases de imagens de problemas da pecuária de precisão e utilizar novas métricas para avaliar os resultados. Além disso, arquiteturas recentes propostas na literatura podem incrementar os resultado obtidos.

Referências Bibliográficas

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., e Süsstrunk, S. (2012). SLIC Superpixels Compared to State-of-the-art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274 – 2282. Citado nas páginas 4 e 7.

Associação Brasileira Indústrias Exportadoras Carnes (2017). Perfil da pecuária no brasil. Citado na página 1.

Badrinarayanan, V., Kendall, A., e Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495. Citado nas páginas 5, 22, e 36.

Beling, R. R. (2014). Anuário brasileiro da pecuária. *Editora Gazeta Santa Cruz Ltda*. Citado na página 1.

Comaniciu, D. e Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619. Citado na página 4.

Felzenszwalb, P. F. e Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181. Citado na página 4.

Ferreira, A. D. S., Freitas, D. M., da Silva, G. G., Pistori, H., e Folhes, M. T. (2017). Weed detection in soybean crops using convnets. *Computers and Electronics in Agriculture*, 143:314 – 324. Citado na página 19.

Frost, A., Schofield, C., Beaulah, S., Mottram, T., Lines, J., e Wathes, C. (1997). A review of livestock monitoring and the need for integrated systems. *Computers and Electronics in Agriculture*, 17(2):139 – 159. Citado na página 2.

Graves, A., r. Mohamed, A., e Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, páginas 6645–6649. Citado na página 5.

Hertem, T. V., Alchanatis, V., Antler, A., Maltz, E., Halachmi, I., Schlageter-Tello, A., Lokhorst, C., Viazzi, S., Romanini, C., Pluk, A., Bahr, C., e Berckmans, D. (2013). Comparison of segmentation algorithms for cow contour extraction from natural barn background in side view images. *Computers and Electronics in Agriculture*, 91:65 – 74. Citado na página 3.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., e Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 1725–1732. Citado na página 5.

Kist, B. B. (2017a). Anuário brasileiro do gado de corte. *Editora Gazeta Santa Cruz Ltda*. Citado na página 1.

Kist, B. B. (2017b). Anuário brasileiro do gado de leite. *Editora Gazeta Santa Cruz Ltda*. Citado na página 1.

Krizhevsky, A., Sutskever, I., e Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., e Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, páginas 1097–1105. Curran Associates, Inc. Citado nas páginas 5 e 11.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., e Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551. Citado na página 4.

Levinshtein, A., Stere, A., Kutulakos, K. N., Fleet, D. J., Dickinson, S. J., e Siddiqi, K. (2009). Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297. Citado na página 4.

Luo, P., Wang, X., e Tang, X. (2012). Hierarchical face parsing via deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 2480–2487. Citado na página 5.

Oquab, M., Bottou, L., Laptev, I., e Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, páginas 1717–1724, Washington, DC, USA. IEEE Computer Society. Citado na página 20.

Ren, S., He, K., Girshick, R., e Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., e Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, páginas 91–99. Curran Associates, Inc. Citado na página 15.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., e Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252. Citado na página 21.

Shelhamer, E., Long, J., e Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651. Citado nas páginas 5 e 36.

Shi, J. e Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905. Citado na página 4.

Simonyan, K. e Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *ArXiv e-prints*. Citado na página 14.

Sun, Y., Wang, X., e Tang, X. (2013). Deep convolutional network cascade for facial point detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 3476–3483. Citado na página 5.

Toshev, A. e Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 1653–1660. Citado na página 5.

Vedaldi, A. e Soatto, S. (2008). Quick shift and kernel methods for mode seeking. In Forsyth, D., Torr, P., e Zisserman, A., editors, *European Conference of Computer Vision*, volume 5305 of *Lecture Notes in Computer Science*, páginas 705–718. Springer Berlin Heidelberg. Citado na página 4.

Viazzi, S., Bahr, C., Hertem, T. V., Schlageter-Tello, A., Romanini, C., Halachmi, I., Lokhorst, C., e Berckmans, D. (2014). Comparison of a three-dimensional and two-dimensional camera system for automated measurement of back posture in dairy cows. *Computers and Electronics in Agriculture*, 100:139 – 147. Citado na página 3.

Vindis, P., Brus, M., Stajnko, D., e Janzekovic, M. (2010). Non invasive weighing of live cattle by thermal image analysis. In *New Trends in Technologies: Control, Management, Computational Intelligence and Network Systems*, chapter 13, páginas 243 – 257. Meng Joo Er (Ed.), InTech. Citado na página 3.