UNIVERSIDADE FEEVALE

ANDREI FILLMANN

SEGMENTAÇÃO AUTOMÁTICA DE PARTES DO CORPO DE BOVINOS

Novo Hamburgo

2021

ANDREI FILLMANN

SEGMENTAÇÃO AUTOMÁTICA DE PARTES DO CORPO DE BOVINOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Feevale

Orientador: Marta Rosecler Bez

Novo Hamburgo

2021

ANDREI FILLMANN

SEGMENTAÇÃO AUTOMÁTICA DE PARTES DO CORPO DE BOVINOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Feevale

APROVADO EM: ___ / ___ / _____

MARTA ROSECLER BEZ
Orientador – Feevale

GABRIEL DA SILVA SIMÕES
Examinador interno – Feevale

JOÃO BATISTA MOSSMANN
Examinador interno – Feevale

Novo Hamburgo
2021

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial: Minha orientadora, Marta Rosecler Bez. E principalmente a minha esposa Nadine e minha filha Lauren, que eu amo muito e foram a minha motivação e meu apoio emocional durante esta etapa.

RESUMO

Devido ao seu grande rebanho, o Brasil exerce um papel de destaque na criação de bovinos, porém sua produção apresenta ainda uma baixa eficiência. As formas de aumentar a produtividade estão ligadas a diminuição do custo, cuidados com alimentação, saúde e bem estar dos animais. A pecuária de precisão busca, através da utilização de tecnologias, melhorar de forma significativa a produção de uma fazenda. Diversas áreas da computação são utilizadas neste novo modelo de pecuária, entre elas, a visão computacional. Esta área tem como objetivo aproximar como as máquinas enxergam o meio à sua volta ao que o olho humano pode ver, reproduzindo a capacidade de reconhecimento de imagens através da utilização de técnicas computacionais. Desta forma, o presente trabalho tem como objetivo apresentar um protótipo de *software* que, através da técnica de segmentação de imagens, utilizando modelos criados nos *frameworks* Mask R-CNN e PointRend, seja capaz de identificar partes do corpo de bovinos em fotografias digitais dos animais, servindo seu resultado como fonte de entrada em modelos que se propõem em estimar pesos ou condições corporais destes animais. A pesquisa foi elaborada através da utilização de referências teóricas e de testes práticos utilizando o protótipo desenvolvido. Sendo esse dirigido a busca de soluções mais eficientes para extração de partes do corpo de bovinos em imagens digitais, já seu resultado foi avaliado de forma quantitativa. Para tanto, é apresentado o referencial teórico, incluindo técnicas de Processamento Digital de Imagens e Redes Neurais Artificiais, bem como uma análise de três trabalhos relacionados. Para avaliar o desempenho dos modelos a métrica mAP foi utilizada, para o Mask R-CNN o resultado foi de 88.529 e para o PointRend 90.576.

Palavras-chave: Segmentação de imagens. Visão computacional. Pecuária de precisão. Bovinos.

ABSTRACT

Due to its large herd, Brazil has an outstanding role in cattle farming; however, its production efficiency is still poor. The ways to increase productivity are linked to decreased cost, feeding care, and animal health and welfare. Precision livestock farming, with the use of technologies, aims at significantly improving farm production. Several computer areas are used in this new livestock farming model, including computer vision. This area has the purpose of approximating how machines see the surrounding environment to that seen by the human eye, reproducing the image recognition ability using computer techniques. Therefore, this work has the purpose of developing a software prototype that, by image segmentation technique, using models created in the Mask R-CNN and PointRend frameworks, is able to identify parts of the bovine body in digital photography of animals, presenting its result as a source of input in models aiming at estimating body weight or conditions of such animals. The research will be prepared by the use of theoretical references and practical tests using the prototype developed. This prototype targets the search for more efficient solutions to extract parts of the bovine body in digital images; its results will be quantitatively assessed. For this purpose, the theoretical reference is presented herein, including Image Digital Processing and Artificial Neural Networks techniques, as well as an analysis of three related works. To evaluate the performance of the models, the mAP metric was used, for Mask R-CNN the result was 88,529 and for PointRend 90,576

Keywords: Image segmentation. Computer vision. Precision livestock farming. Cattle

LISTA DE ILUSTRAÇÕES

Figura 1 – Etapas de PDI 16
Figura 2 – Resultado de um histograma. (a) imagem; (b) histograma correspondente 18
Figura 3 – Limiarização de uma imagem 18
Figura 4 – Brilho e contraste. (a) Imagem original; (b) Imagem com aumento de brilho; (c) Imagem com diminuição de brilho; (d) Imagem com aumento de contraste; (e) Imagem com diminuição de contraste 19
Figura 5 – Filtro de mediana 21
Figura 6 – Máscara 1D e a forma inteira 5x5 do filtro Gaussiano. 22
Figura 7 – Modelos de borda. (a) em degrau; (b) em rampa; (c) em telhado 23
Figura 8 – Operações morfológicas de dilatação e erosão. (a) imagem original binarizada; (b) dilatação; (c) erosão 24
Figura 9 – Operações morfológicas de abertura e fechamento. (a) imagem original binarizada; (b) abertura; (c) fechamento 25
Figura 10 – Principais operadores de descrição. 27
Figura 11 – Classificação de objetos. (a) supervisionado, (b) não supervisionado . . 28

Figura 12 – Neurônio biológico e artificial. (a) biológico; (b) artificial 29
Figura 13 – Arquitetura de uma rede Perceptron 30
Figura 14 – Arquitetura CNN 31
Figura 15 – Geração do mapa de recursos 32
Figura 16 – Máscara convolucional 2x2 32
Figura 17 – Filtro convolucional 33
Figura 18 – Processo de max pooling 33
Figura 19 – Ilustração gráfica da função sigmoid 35
Figura 20 – Ilustração gráfica da função ReLU 35
Figura 21 – Data-augmentation 38
Figura 22 – *Region-based Convolutional Network* 38
Figura 23 – *Non-maximum suppression* (NMS) 40
Figura 24 – *Region Proposal Network* (RPN). (a) Arquitetura RPN; (b) Caixas delimitadoras previstas 40
Figura 25 – ROI *pooling* 41
Figura 26 – *Feature Pyramid Network* (FPN) 42
Figura 27 – *Fully Convolutional Network* (FCN) 43
Figura 28 – *Arquitetura FCN* 43
Figura 29 – Bloco Residual 44
Figura 30 – Módulo residual 45

Figura 31 – Exemplo de máscara gerada pelo framework Mask R-CNN 46
Figura 32 – *Camada ROIAlign* 46
Figura 33 – Arquitetura do *framework* Mask R-CNN 47
Figura 34 – Arquitetura PointRend 47
Figura 35 – Comparação Mask R-CNN x PointRend 48

Figura 36 – Ilustração do dataset de imagens do trabalho de Danish(2018) 49
Figura 37 – Ilustração da representação da máscara da imagem do trabalho de Danish(2018). 50
Figura 38 – Ilustração da máscara prevista do trabalho de Danish(2018). 51
Figura 39 – Hardware utilizado por Qiao, Truman e Sukkarieh (2019) 54
Figura 40 – Ilustração do processo de melhoria da iluminação de imagem utilizado por Qiao, Truman e Sukkarieh (2019) 54
Figura 41 – Métricas ADE E MPA 55
Figura 42 – Máscaras previstas em imagens com dois animais no trabalho de Qiao, Truman e Sukkarieh (2019) 56
Figura 43 – Máscaras previstas no trabalho de Qiao, Truman e Sukkarieh (2019) . 57
Figura 44 – Detecções previstas no trabalho de Jiang *et al.*, 2019. (a) FLYOLOv3, (b) Faster R-CNN (c) YOLOv3. 59

Figura 45 – Exemplo de imagens utilizadas 62
Figura 46 – Ferramenta COCO Annotator 62
Figura 47 – Google Colab 64
Figura 48 – Configuração do Colab 66
Figura 49 – Código Python responsável pelo treinamento 67
Figura 50 – Treinamento com *overfitting* 67
Figura 51 – Resultado do treinamento 68

Figura 52 – Imagem Segmentada - Mask R-CNN X Point Rend 71

LISTA DE TABELAS

Tabela 1 – Hiperparâmetros utilizados no trabalho de Danish(2018) 52
Tabela 2 – Performance dos modelos propostos do trabalho de Danish(2018) . . . 53
Tabela 3 – Resultados da segmentação nos conjuntos de dados de imagens bruta e aprimorada no trabalho de Qiao, Truman e Sukkarieh (2019) 56
Tabela 4 – Resultados do cálculo do ADE no trabalho de Qiao, Truman e Sukkarieh (2019) 56
Tabela 5 – Resultado da detecção no trabalho de Jiang, *et al.*, 2019. 58
Tabela 6 – Resultado da detecção por partes do corpo no trabalho de Jiang, *et al.*, 2019. 60

Tabela 7 – Distribuição das partes do corpo dos animais nas imagens 65
Tabela 8 – Hiperparâmetros utilizados nos modelos 65

Tabela 9 – Resultados dos modelos 70
Tabela 10 – Resultados dos modelos por classes 71

LISTA DE ABREVIATURAS E SIGLAS

1D	Unidimensional
2D	Bidimensional
3D	Tridimensional
ADE	*Average Distance Erro*
BCS	*Body Condition Score*
BN	*Batch Normalization*
CAFFE	*Convolutional Architecture for Fast Feature Embedding*
CNN	*Convolutional Neural Networks*
CPU	*Central Process Unit*
dB	Decibel
DFT	*Discrete Fourier Transform*
FCN	*Fully Convolutional Network*
FFT	*Fast Fourier Transform*
FPN	*Feature Pyramid Networks*
FPS	*Frames Per Second*
GB	Gigabyte
GHz	Gigahertz
GPU	*Graphics Processing Unit*
IoU	*Intersection over Union*
JSON	*JavaScript Object Notation*
MAP	*Mean Average Precision*
MB	Megabyte
MPA	*Mean Pixel Accuracy*
NMS	*Non-maximum Suppression*

PDI	Processamento digital de imagens
R-CNN	*Region-based Convolutional Network*
RAM	*Random Access Memory*
RESNET	*Residual Networks*
RGB	*Red, Green, Blue*
RNA	Redes Neurais Artificiais
ROI	*Region of Interest*
RPN	*Region Proposal Network*
SVM	*Support Vector Machine*
TB	Terabyte

SUMÁRIO

1 INTRODUÇÃO 13

2 PROCESSAMENTO DIGITAL DE IMAGENS 16

2.1 PRÉ-PROCESSAMENTO 17

2.1.1 Histograma 17

2.1.2 Limiarização (*Thresholding*) 17

2.1.3 Brilho e contraste 19

2.1.4 Filtro no domínio espacial 20

2.1.4.1 Filtro de mediana 20

2.1.5 Filtro no domínio da frequência 20

2.1.5.1 Filtro Gaussiano 21

2.2 SEGMENTAÇÃO 22

2.2.1 Detecção de borda 22

2.3 PÓS-PROCESSAMENTO 23

2.3.1 Morfologia Matemática 23

2.3.1.1 Erosão 24

2.3.1.2 Dilatação 24

2.3.1.3 Abertura 25

2.3.1.4 Fechamento 25

2.4 EXTRAÇÃO DE ATRIBUTOS 25

2.4.1 Descritores de borda 26

2.4.2 Descritores de região 26

2.5 CLASSIFICAÇÃO E RECONHECIMENTO 27

3 REDES NEURAIS ARTIFICIAIS 29

3.1 REDES NEURAIS CONVOLUCIONAIS 31

3.1.1 *Convolutional Layer* 31

3.1.2 *Pooling Layer* 33

3.1.3 *Fully Connected Layer* 34

3.1.4 Função de Ativação 34

3.1.4.1 Sigmóide 34

3.1.4.2 ReLU 35

3.1.4.3 Softmax 36

3.1.5 Função de Perda 36

3.1.5.1 Entropia cruzada 36

3.1.6 Processo de Aprendizagem 36

3.1.6.1 Inicialização dos pesos . . . 36
3.1.6.2 Backpropagation . . . 37
3.1.6.3 Overfitting . . . 37
3.1.6.4 Dropout . . . 37
3.1.6.5 Batch normalization . . . 37
3.1.6.6 Data-augmentation . . . 37
3.2 *REGION-BASED CONVOLUTIONAL NETWORK* (R-CNN) . . . 38
3.2.1 *Region Proposal Network* (RPN) . . . 39
3.2.2 *Feature Pyramid Network* (FPN) . . . 41
3.3 *FULLY CONVOLUTIONAL NETWORK* (FCN) . . . 42
3.4 RESNET . . . 44
3.5 MASK R-CNN . . . 45
3.6 POINTREND . . . 47

4 TRABALHOS RELACIONADOS . . . 49
4.1 *BEEF CATTLE INSTANCE SEGMENTATION USING MASK R-CNN (DANISH, 2018)* . . . 49
4.2 *CATTLE SEGMENTATION AND CONTOUR EXTRACTION BASED ON MASK R-CNN FOR PRECISION LIVESTOCK FARMING (QIAO, TRUMAN E SUKKARIEH, 2019)* . . . 53
4.3 *FLYOLOV3 DEEP LEARNING FOR KEY PARTS OF DAIRY COW BODY DETECTION (JIANG, ET AL., 2019)* . . . 57

5 DESENVOLVIMENTO DO PROTÓTIPO . . . 61
5.1 BASE DE DADOS UTILIZADA . . . 61
5.2 TECNOLOGIAS . . . 62
5.2.1 Detectron2 . . . 63
5.2.2 Python . . . 63
5.2.3 PyTorch . . . 63
5.2.4 OpenCV . . . 64
5.2.5 Google Colab . . . 64
5.3 TREINAMENTO DO MODELO . . . 65

6 RESULTADOS E DISCUSSÃO . . . 70

7 CONCLUSÃO . . . 73

Referências . . . 75

1 INTRODUÇÃO

O Brasil possui o maior rebanho de gado bovino comercial do mundo, com cerca de 213,5 milhões de animais, produzindo 10,100 milhões de toneladas de carne (IBGE, 2020). Em contrapartida, os EUA, que possuem cerca de 94,4 milhões de cabeças de gado, produzem 12,397 milhões de toneladas de carne bovina (USDA, 2021). Os números demonstram que a eficiência na produção de gado de corte no país ainda é baixa, pois com menos da metade do rebanho os EUA conseguem produzir um volume de carne 20% maior que o Brasil.

A busca por maior produtividade dentro do sistema de produção de carne abrange o interesse em procurar alternativas que reduzam os custos de produção (ROSA *et al.*, 2014). Vale ressaltar ainda que a melhoria da produtividade e da eficiência dos sistemas de produção têm na alimentação animal seu principal componente (EUCLIDES, 2000).

Uma das formas de melhorar a produtividade na criação é conhecendo o peso vivo dos animais. Ao longo dos anos diversos estudos vêm sendo realizados e demonstrando com sucesso a correlação de medidas corporais com o peso real de animais, principalmente utilizando as medidas do perímetro torácico e o comprimento corporal (REIS *et al.*, 2008).

Outro ponto decisivo para o avanço na criação de gado são os cuidados com saúde e bem estar dos animais. Através de alguns indicadores visuais ou táteis, como condição corporal, dificuldade de andar, higiene e lesões físicas, é possível determinar a vitalidade dos bovinos. A análise destes indicadores geralmente necessita de um avaliador treinado, elevando assim os custos de produção, além disso, esse processo está sujeito a falhas devido a subjetividade da pessoa responsável por essa avaliação. Já os indicadores táteis muitas vezes acabam sendo trabalhosos e estressantes para o animal.

A pecuária de precisão busca automatizar processos através do uso de tecnologia em diversas áreas de uma fazenda. De acordo com Qiao, Truman e Sukkarieh (2019), abordagens baseadas em visão computacional, para análise e interpretação de fotos e vídeos, têm sido amplamente utilizadas neste novo modelo de pecuária, principalmente empregando técnicas de segmentação, detecção e classificação de imagens.

A visão computacional é a ciência que busca aproximar o que e como as máquinas enxergam o meio à sua volta ao que olho humano pode ver. Ou seja, a visão computacional tem por objetivo reproduzir a capacidade de reconhecimento de imagens a partir de diversas técnicas computacionais, partindo de imagens e chegando a modelos matemáticos (BORTH *et al.*, 2014).

Segundo Gonzalez e Woods (2010) o processamento de imagens pode ser dividido em três níveis: baixo-nível, médio-nível e alto-nível. Os processos de baixo-nível envol-

vem operações primitivas, tais como a redução de ruído ou melhoria no contraste de uma imagem. Os processos de nível-médio são operações de segmentação e classificação, já os processos de alto-nível estão relacionados com as tarefas de cognição associadas com a visão humana. Sendo assim, a visão computacional compreende os campos do processamento da imagem, modelos de reconhecimento, análises da cena, interpretação da cena, processamento óptico, processamento de vídeo e a compreensão da imagem (COHEN; FEIGENBAUM, 1984; HAVENS; MACKWORTH, 1986). Várias pesquisas têm sido realizadas relacionando a visão computacional e a pecuária, como os apresentados na sequência.

Uma das formas de avaliar a qualidade da alimentação dos animais é através da verificação do ganho de seu peso. Ozkaya e Bozkurt (2008), Ozkaya (2013), Ozkaya *et al.* (2016) estimaram automaticamente o peso de bovinos analisando imagens digitais. O seu método está relacionado a utilização das medidas da cernelha, a altura do quadril, comprimento do corpo e a largura do quadril do animal como variáveis em um modelo de regressão linear, que tem como objetivo estabelecer uma equação que possa ser usada para explicar a variação de uma variável dependente (peso) para valores dados pelas variáveis independentes (medidas corporais).

Conforme Alvarez *et al.* (2018), o principal problema em seu modelo de Rede Neural Convolucional que foi capaz de predizer o *Body Condition Score* (BCS) de gados da raça Holandesa, é o ruído e a influência de fatores não relacionados aos animais, mas específicos do local das imagens, concentrando esforços considerados na tentativa de reduzir ou eliminar essas interferências.

Existem diversas técnicas clássicas de segmentação de imagens, entre elas é possível citar: Segmentação por detecção de borda, por corte, por crescimento de região, além de segmentação de cores e intensidades e baseadas em histogramas, etc. Atualmente as técnicas de segmentação utilizando Redes Neurais Convolucionais são consideradas o estado da arte para essa tarefa, sendo assim, estas serão abordadas neste trabalho.

O presente trabalho tem como objetivo o desenvolvimento de um protótipo de *software* que, através da técnica de segmentação de imagens, seja capaz de identificar partes do corpo de bovinos em fotografias, servindo seu resultado como fonte de entrada em modelos que se propõem em estimar pesos ou condições corporais destes animais.

A pesquisa, sob o ponto de vista da sua natureza, é caracterizada como aplicada, pois teve como objetivo gerar conhecimentos para aplicação prática através do desenvolvimento de um protótipo, sendo este dirigido à solução de problemas específicos.

A forma de abordagem foi quantitativa, onde o valor médio da precisão dos resultados da segmentação automática foi avaliado em relação ao conjunto de imagens anotadas manualmente. As métricas utilizadas foram mAP e AP.

O objetivo do estudo foi exploratório, pois consiste na busca de soluções mais eficientes para extração automática de partes do corpo de bovinos em imagens digitais.

Do ponto de vista dos procedimentos técnicos, os dados necessários para a elaboração da pesquisa foram obtidos de duas formas. Bibliográfica, caracterizada através da utilização de livros e artigos científicos já publicados sobre o tema. E, experimental, já que através do protótipo do *software* foram realizados testes práticos.

Este trabalho está dividido em 7 capítulos, sendo o primeiro esta introdução. No Capítulo 2, o referencial teórico referente a PDI é demonstrado. Enquanto o referencial teórico referente a Redes Neurais Artificiais é descrito no Capítulo 3. No Capítulo 4, é apresentada uma análise referente a três trabalhos relacionados ao presente estudo. O desenvolvimento do protótipo é apresentado no Capítulo 5. O resultado e a conclusão são expostos no Capítulo 6 e Capítulo 7, respectivamente.

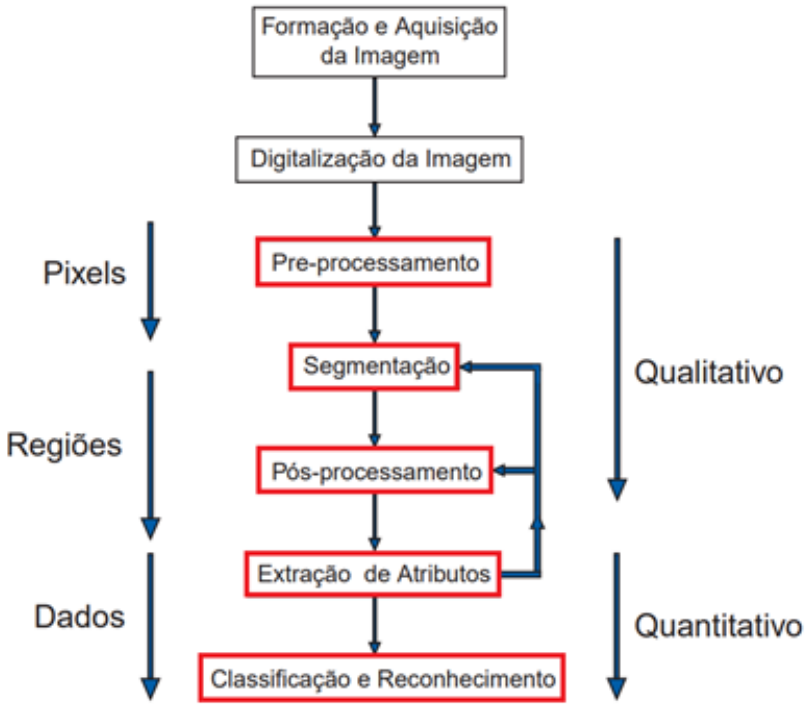
2 PROCESSAMENTO DIGITAL DE IMAGENS

Segundo Queiroz e Gomes (2016), os computadores não são capazes de processar imagens contínuas, mas apenas *arrays* de números digitais, sendo necessário representar imagens como arranjos bidimensionais de pontos. Cada ponto na grade bidimensional que representa a imagem digital é denominado *pixel*.

Uma imagem digital monocromática é definida por uma função bidimensional contínua *f(x,y)*, na qual x e y são coordenadas espaciais e o valor de f em qualquer ponto (x,y) é proporcional à intensidade de brilho ou nível de cinza no ponto considerado. Já em uma imagem digital colorida no sistema RGB, um *pixel* pode ser visto como um vetor cujas componentes representam as intensidades de vermelho, verde e azul de sua cor (QUEIROZ; GOMES, 2016). Cabe ressaltar que os valores dos *pixels* em imagens deste sistema vão de 0 (preto) a 255 (branco).

O processamento digital de imagens, ou PDI, envolve processos de manipulação de uma imagem por computador, onde a entrada e saída são imagens, além disso, de acordo com Gonzalez e Woods (2010), envolve processos de extração de atributos das imagens e reconhecimento de objetos individuais. A Figura 1 demonstra as etapas de PDI, destacadas em vermelho as que serão abordadas no presente trabalho.

Figura 1 – Etapas de PDI



Fonte: Adaptado de Esquef, Albuquerque e Albuquerque (2003)

2.1 PRÉ-PROCESSAMENTO

As imagens brutas podem apresentar diversas imperfeições. A presença de ruído ocasionado nos processos de aquisição, transmissão ou compressão, assim como, contraste ou brilho inadequados, são exemplos de problemas encontrados em imagens. Desta forma, segundo Esquef, Albuquerque e Albuquerque (2003), as técnicas de pré-processamento tem a função de melhorar a qualidade de uma imagem, ou seja, após a realização dos processos a imagem resultante se torna mais adequada que a imagem original para uma aplicação específica.

De acordo com Marques Filho e Vieira Neto (1999), as técnicas são por natureza orientadas a um problema que se deseja resolver. Portanto, não existem técnicas capazes de resolver todos os problemas que uma imagem digital possa apresentar, como também, nem sempre uma técnica produz os resultados esperados. Nas seções seguintes serão destacadas algumas das técnicas empregadas nesta etapa.

2.1.1 Histograma

Os resultados de um histograma geralmente são apresentados em forma de um gráfico de barras, onde seus valores fornecem uma visão estatística sobre a distribuição dos *pixels*, o contraste da imagem e os níveis de iluminação. Os histogramas são utilizados como base de diversas técnicas de pré-processamento de imagens, além disso, o histograma também é muito utilizado na etapa de segmentação. Uma imagem e o histograma correspondente são apresentados na Figura 2.

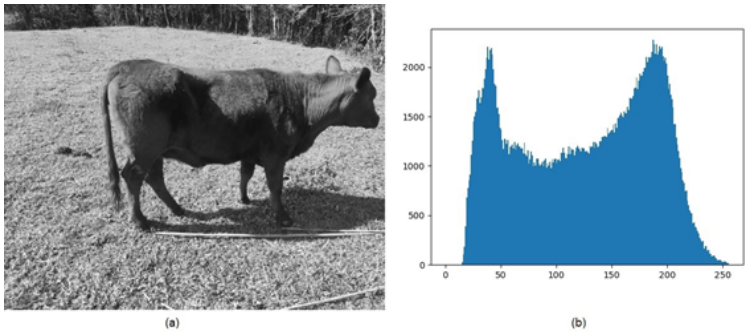
> Em uma imagem escura os componentes do histograma estão concentrados no lado inferior da escala de intensidades. De forma similar, os componentes do histograma de uma imagem clara tendem a direção do lado superior da escala. Uma imagem com baixo contraste tem um histograma estreito, normalmente localizado no meio da escala. Já os componentes do histograma em uma imagem de alto contrastes cobrem uma faixa bem ampla da escala de intensidades (GONZALEZ; WOODS, 2010).

2.1.2 Limiarização (*Thresholding*)

A técnica de limiarização consiste na separação de uma imagem quando esta apresenta várias classes ou pelo menos duas classes, fundo e objeto. Uma das formas de fazer a limiarização é utilizando o resultado de um histograma. Quando um histograma apresenta dois picos, um representando o objeto e o outro o fundo, um valor central entre eles pode ser usado para gerar uma binarização da imagem. Os *pixels* cujo tom de cinza é maior ou igual ao valor de limiar encontrado, são transformados em brancos e os demais em pretos.

Muitas vezes a escolha do valor de limiar se torna subjetivo, porém existem outras técnicas capazes de determinar o valor de limiar, uma delas é denominada limiarização

Figura 2 – Resultado de um histograma. (a) imagem; (b) histograma correspondente

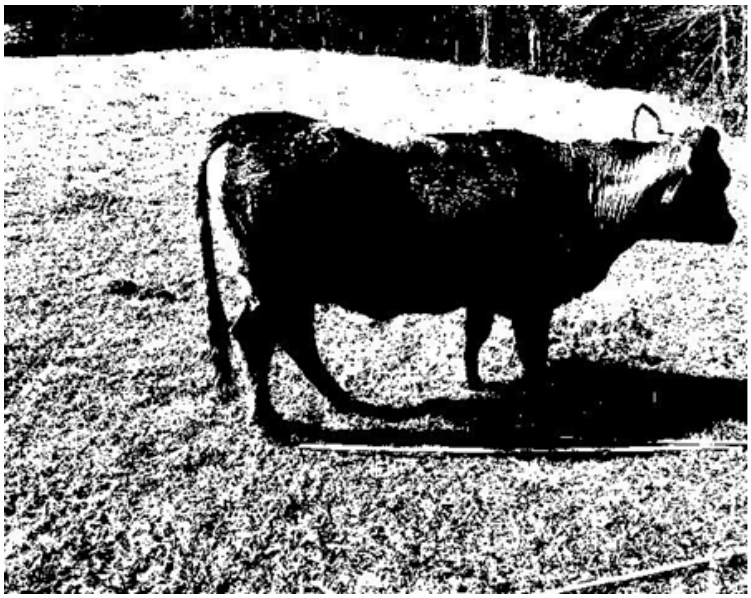


Fonte: Do autor (2021)

ótima, onde os cálculos do valor ótimo de limiar são feitos com base nas propriedades estatísticas da imagem. Este valor é determinado por uma função matemática utilizando a média dos tons de cinza da região de interesse, média dos tons de cinza da região de fundo, desvio padrão e probabilidade de ocorrência dos *pixels* pertencentes a esta ou aquela região.

A iluminação desempenha um papel significativo no processo de limiarização, uma vez que provoca alterações no histograma original da imagem, eventualmente eliminando uma região de vale entre dois picos, naturalmente propícia para a definição de um limiar global (MARQUES FILHO; VIEIRA NETO, 1999). A Figura 3 apresenta um exemplo de limiarização.

Figura 3 – Limiarização de uma imagem



Fonte: Do autor (2021)

2.1.3 Brilho e contraste

O brilho em uma imagem digital representa a intensidade luminosa emitida por fontes de luz, como o sol ou uma lâmpada. Já o contraste pode ser definido como uma medida da variação relativa da luminância, ou seja, é definido como a diferença de cor, ou intensidade dos níveis de cinza que tornam um objeto distinguível em uma imagem. O objetivo das transformações de contraste é melhorar a qualidade das imagens sob critérios subjetivos ao sistema visual humano, tornando mais fácil a percepção de informações contidas nas imagens (PEDRINI; SCHWARTZ, 2008).

O brilho e contraste são regidos pela fórmula: $C*f(x,y)+B$, onde o C corresponde ao contraste, o B ao brilho e o $f(x,y)$ ao valor do *pixel*. Se o contraste não for alterado, o valor de C deve ser um e se o brilho não for alterado, seu valor deve ser zero. A Figura 4 apresenta exemplos de imagens aplicando a fórmula do brilho e contraste.

Figura 4 – Brilho e contraste. (a) Imagem original; (b) Imagem com aumento de brilho; (c) Imagem com diminuição de brilho; (d) Imagem com aumento de contraste; (e) Imagem com diminuição de contraste

Fonte: Do autor (2021)

2.1.4 Filtro no domínio espacial

As técnicas de filtragem no domínio espacial atuam diretamente sobre a matriz de *pixels* de uma imagem, produzindo uma nova imagem a partir da imagem de entrada. Conforme Marques Filho e Vieira Neto (1999), as funções de processamento de imagens no domínio espacial podem ser expressas como: $g(x,y) = T\ [\ f(x,y)\ ]$. Onde $g(x,y)$ é a imagem processada, $f(x,y)$ é a imagem original e T é um operador em f, definido em uma certa vizinhança de (x,y).

No domínio espacial, o processo de filtragem normalmente é realizado por meio de matrizes denominadas máscaras (ou *kernel*), que são aplicadas sobre a imagem. A cada posição da máscara está associado um valor numérico, chamado de peso ou coeficiente (PEDRINI; SCHWARTZ, 2008).

A letra x representa coluna enquanto y representa a linha em uma imagem. O valor do *pixel* no ponto (x,y) é substituído por um novo valor, este depende dos valores dos *pixels* vizinhos e dos pesos da máscara. A máscara então desliza para o próximo *pixel*, repetindo este processo que é chamado de correlação, até que todos os pontos da imagem tenham sidos processados. Outra operação muito similar é a convolução, que se diferencia da correlação apenas pelo fato de a máscara ser rotacionada em 180º.

2.1.4.1 Filtro de mediana

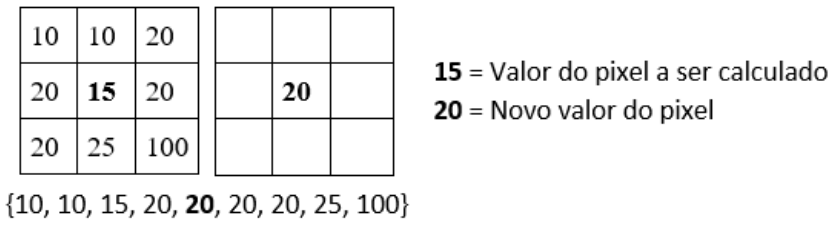
O filtro de mediana utiliza uma máscara para substituir o valor de um *pixel* pelo valor de mediana dos pixels vizinhos. Os filtros de mediana são particularmente eficazes na presença de ruído impulsivo, também chamado de ruído sal e pimenta, em razão de sua aparência, como pontos brancos e pretos sobrepostos em uma imagem (GONZALEZ; WOODS, 2010).

Para aplicar um filtro de mediana em uma imagem, primeiramente os *pixels* precisam ser ordenados de forma crescente. Logo após, é calculada a mediana. Em um conjunto de valores numéricos ímpar, a mediana será o valor central do conjunto e quando ímpar, um dos centrais. Por último, o valor do *pixel* correspondente é substituído na imagem filtrada. Desta forma, o filtro acaba forçando com que os níveis de intensidades distintos se tornem mais semelhantes ao de seus vizinhos. A Figura 5 ilustra este processo.

2.1.5 Filtro no domínio da frequência

Para realizar os filtros no domínio da frequência é necessário primeiramente passar a imagem do domínio espacial para o de frequência. Para realizar este processo é utilizada a transformada de Fourier, mais precisamente utilizando a forma discreta (DFT – *Discrete Fourier Transform*) e rápida (FFT - *Fast Fourier Transform*). Logo após, operações de

Figura 5 – Filtro de mediana



Fonte: Adaptado de Gonzalez e Woods (2010)

filtragem são realizadas na imagem. E por último, ocorre o processo inverso, onde a imagem no domínio da frequência é transformada para o domínio espacial.

Na transformada de Fourier, não há perda de informação durante a mudança de domínios, apenas a informação visual da imagem está representada de uma outra forma. Esta forma pode conter informações sobre toda a imagem no domínio espacial, indicando quanto desta frequência há na imagem (VIOLA, 2007). As baixas frequências no resultado da transformada correspondem a uma variação suave de intensidade. Já as altas frequências, significam mudanças repentinas de intensidade, como bordas e ruídos.

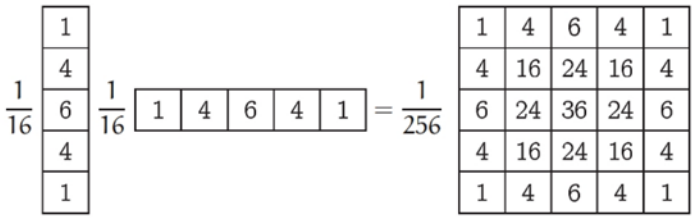
2.1.5.1 Filtro Gaussiano

A função Gaussiana é utilizada para definir os pesos da máscara que será utilizada como um filtro de suavização, onde as altas freqüências que correspondem às transições abruptas são atenuadas, do mesmo modo é utilizada para diminuir o ruído nas imagens. A função Gaussiana com média zero e desvio padrão σ é definida como:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right)$$

A máscara que define a função Gaussiana é teoricamente infinita, mas na prática se zera a uma distância de três *pixels* do centro. Para melhorar o desempenho do algoritmo, em vez de se utilizar uma convolução em 2D, podemos fazer primeiro uma convolução em 1D na direção x e depois usar a mesma máscara 1D da Gaussiana na direção y. A Figura 6 mostra as máscaras utilizadas para os eixos x e y, e a forma inteira 5 x 5 da máscara da Gaussiana para o desvio padrão σ= 1.0.

Figura 6 – Máscara 1D e a forma inteira 5x5 do filtro Gaussiano.



Fonte: Adaptado de Pedrini e Schwartz (2008)

2.2 SEGMENTAÇÃO

A segmentação é o processo de separar a imagem em pedaços menores a fim de se identificar e diferenciar os objetos de interesse presentes em uma imagem. Este processo é considerado o mais crítico, pois por consequência de qualquer erro ou distorção presentes nesta etapa, os resultados serão negativamente influenciados nas etapas posteriores.

Deve ser ressaltado que não existe um modelo formal para a segmentação de imagens. A segmentação é um processo empírico e adaptativo, procurando sempre se adequar às características particulares de cada tipo de imagem e aos objetivos que se pretende alcançar (ESQUEF; ALBUQUERQUE; ALBUQUERQUE, 2003).

Os algoritmos convencionais de segmentação são baseados em dois princípios: A similaridade entre os *pixels* e a descontinuidade entre eles. As técnicas baseadas na similaridade buscam unir os *pixels* da imagem que apresentam valores e características semelhantes de acordo com um conjunto de critérios definidos. A binarização, que foi abordada na subseção 2.1.1, é um exemplo de técnica utilizada por este princípio. Já a descontinuidade visa separar a imagem de acordo com mudanças bruscas de intensidades, como a borda de um objeto. A técnica de detecção de bordas utilizada por este princípio é detalhada na seção a seguir.

2.2.1 Detecção de borda

De acordo com Pedrini e Schwartz (2008), uma borda é o limite entre duas regiões com propriedades de nível de cinza dissemelhantes. Sendo assim, a detecção de borda tem como objetivo segmentar um objeto através das mudanças significativas dos níveis de cinza da imagem.

As bordas são classificadas conforme seu perfil de intensidade. Uma borda em degrau envolve uma transição entre dois níveis de intensidade que ocorrem idealmente com uma distância de 1 *pixel*. Já uma borda com perfil em rampa, não tem-se mais uma borda fina de 1 *pixel* de espessura. Em vez disso, um ponto de borda agora é qualquer ponto contido na rampa, e um segmento de borda seria, então, um conjunto desses pontos

conectados. Um terceiro modelo de borda é chamado de borda em formato de telhado. As bordas em forma de telhado, são nada mais do que uma linha com 1 *pixel* de espessura que atravessa uma região da imagem (GONZALEZ; WOODS, 2010). A Figura 7 ilustra os três tipos apresentados.

Figura 7 – Modelos de borda. (a) em degrau; (b) em rampa; (c) em telhado

Fonte: Gonzalez e Woods (2010)

2.3 PÓS-PROCESSAMENTO

O pós-processamento sucede a segmentação. Nesta etapa são corrigidos eventuais problemas e imperfeições ocorridas durante a etapa anterior. Normalmente, estes defeitos da segmentação são corrigidos através de técnicas de Morfologia Matemática, com a aplicação em sequência de filtros morfológicos que realizam uma análise quantitativa dos *pixels* da imagem (ESQUEF; ALBUQUERQUE; ALBUQUERQUE, 2003).

2.3.1 Morfologia Matemática

A morfologia matemática permite a construção de operadores úteis para a descrição de objetos na imagem. Os operadores morfológicos podem ser utilizados em um grande número de aplicações em processamento digital de imagens, tais como extração de componentes conexos, busca de padrões específicos na imagem, delimitação do fecho convexo, extração de bordas dos objetos, afinamento de bordas e muitas outras (PEDRINI; SCHWARTZ, 2008).

Os métodos utilizados para representar os objetos são baseados na teoria dos conjuntos (união, interseção, complemento etc.) e a noção de forma básica, também chamada de elemento estruturante. Nas imagens binarizadas cada elemento do conjunto é representado por um vetor bidimensional, onde os valores são as coordenadas *(x,y)* de um *pixel*. Por convenção, os objetos em uma imagem binária são representados por *pixels* pretos (valor 1), enquanto o fundo será formado por *pixels* brancos (valor 0).

Já as imagens em escala de cinza podem ser representadas por um conjunto onde cada elemento possui 3 valores. Dois componentes de cada elemento referem-se as coordenadas *(x,y)*, e o terceiro corresponde ao seu valor de intensidade. A seguir alguns dos operadores morfológicos utilizados serão descritos.

2.3.1.1 Erosão

A erosão é uma operação onde os detalhes menores que o elemento estruturante são afinados, diminuídos ou removidos da imagem. A erosão de A por B, indicada por A B, é definida como:

$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$

Essa equação indica que a erosão de A por B é o conjunto de todos os pontos Z de forma que B, translado por Z está contido em A. Como a afirmação de que B tem de estar contido em A é equivalente a dizer que B não tem elementos comuns com o fundo (GONZALEZ; WOODS, 2010). É possível expressar a erosão da seguinte forma:

$$A \ominus B = \{z \mid (B)_z \cap A^c = \varnothing\}$$

2.3.1.2 Dilatação

Na dilatação pequenos espaços são preenchidos e áreas são conectadas, aumentando, ou engrossando objetos na imagem. A operação de dilatação entre o conjunto A e o elemento estruturante B é definida como:

$$\mathcal{D}(A, B) = A \oplus B = \bigcup_{b \in B} (A + b)$$

O processo de dilação entre A e B corresponde ao conjunto de todas as translações de B com os pontos da imagem em que há pelo menos um elemento não nulo em comum com o conjunto A (PEDRINI; SCHWARTZ, 2008). Na Figura 8 é apresentado o resultado das operações de dilatação e erosão em imagens.

Figura 8 – Operações morfológicas de dilatação e erosão. (a) imagem original binarizada; (b) dilatação; (c) erosão

Fonte: Do autor (2020)

2.3.1.3 Abertura

A operação de abertura é utilizada para suavizar contornos, eliminar conexões estreitas e remover saliências nos objetos da imagem. A abertura de A por B é simplesmente a erosão de A por B, seguida de uma dilatação do resultado por B, ou seja, é a união de todas as translações de B que estão contidas na imagem A (PEDRINI; SCHWARTZ, 2008). Sendo definida como:

$$A \bigcirc B = (A \ominus B) \oplus B$$

2.3.1.4 Fechamento

O fechamento é utilizado para unir separações estreitas, eliminar pequenos buracos e preencher lacunas no contorno de objetos da imagem. O fechamento de A por B é a dilatação de A por B, seguida de uma erosão do resultado por B, ou seja, a união de todas as translações de B que não estão contidas em A (PEDRINI; SCHWARTZ, 2008). Sendo definida como:

$$A \bullet B = (A \oplus B) \ominus B$$

Na Figura 9 é apresentado o resultado das operações de abertura e fechamento em imagens.

Figura 9 – Operações morfológicas de abertura e fechamento. (a) imagem original binarizada; (b) abertura; (c) fechamento

Fonte: Do autor (2020)

2.4 EXTRAÇÃO DE ATRIBUTOS

Nesta etapa ocorre uma seleção de características extraídas dos objetos contidos nas imagens, onde os *pixels* resultantes são representados e descritos de forma apropriada, a fim de serem classificadas e reconhecidas na próxima etapa. Segundo Pedrini e Schwartz (2008), um objeto pode ser representado em termos de suas características externas, neste caso as bordas, ou características internas, que são os *pixels* que compõem o objeto. Já

a descrição deve permitir a caracterização de sua forma sem ambiguidades e com um número reduzido de características ou medidas extraídas do objeto.

Nas subseções a seguir serão abordadas técnicas utilizadas para descrição de objetos em imagens segmentadas. Em muitos dos casos é necessário a combinação dos descritores de borda e região. A Figura 10 apresenta alguns dos atributos utilizados por estes descritores.

2.4.1 Descritores de borda

Pedrini e Schwartz (2008) citam que os descritores de borda são baseados em propriedades geométricas, sendo esses sensíveis a resolução da imagem. Entre os principais atributos destes descritores podemos citar o diâmetro, perímetro e a curvatura.

O diâmetro é a maior distância entre dois pontos que pertencem a borda, sendo que este valor não se altera devido a rotação ou translação do objeto. O perímetro ou comprimento pode ser obtido pela contagem dos *pixels* ao longo da borda do objeto. Já a curvatura mede a taxa de inclinação do contorno ou borda entre pontos do objeto, de modo que os pontos em uma reta tenham curvatura zero e uma circunferência detenha curvatura constante

2.4.2 Descritores de região

Nos descritores de região todos os *pixels* que compõem o objeto são considerados, não apenas os que formam a borda da região. Alguns dos principais atributos dos descritores de região são: Área, circularidade, projeção vertical e horizontal etc.

A Área corresponde ao total dos pixel existentes em um objeto ou região, sendo aplicada geralmente em situações em que a escala dos objetos não varia. Segundo Pedrini e Schwartz (2008), em uma imagem binária, a área de um objeto em um retângulo com dimensões de M x N *pixels*, em que os *pixels f(x,y)* do objeto são representados pelo valor 1 é definido como:

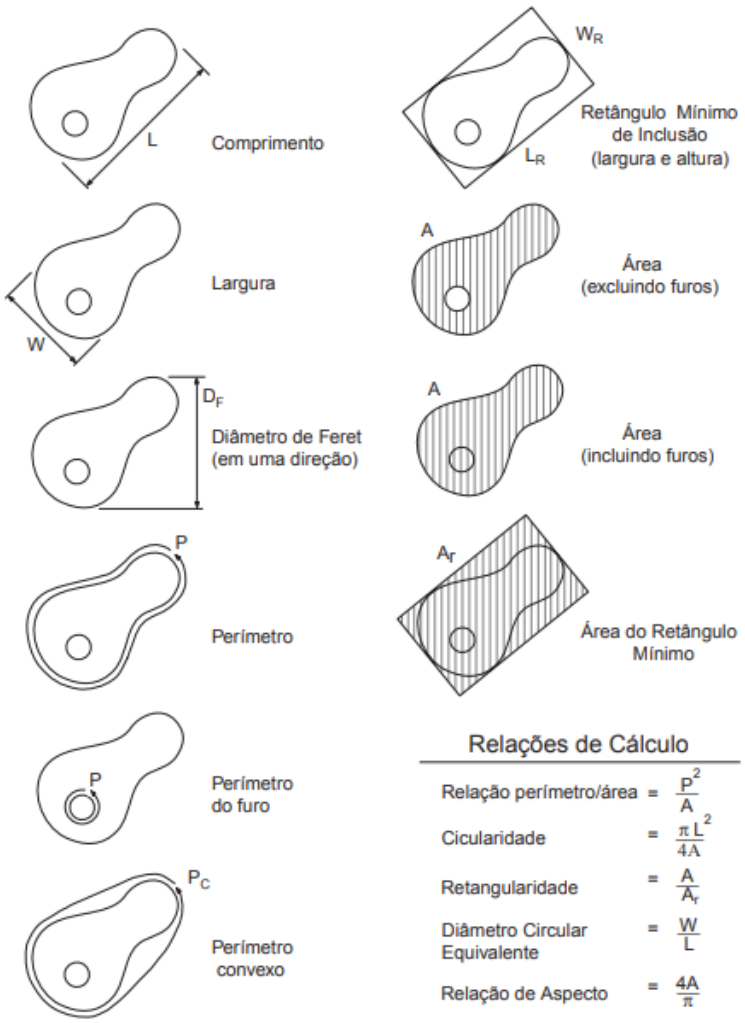
$$\text{Área} = \frac{1}{2}\left|\sum_{k=0}^{n-1}(x_k y_{k+1} - x_{k+1} y_k)\right|$$

Já a circularidade, é definida pela expressão abaixo, onde P significa o perímetro e A representa a área em medidas de *pixel*.

A projeção vertical é definida como a soma dos *pixels* existentes em cada coluna do objeto. Enquanto a projeção horizontal é definida como a soma dos *pixels* em cada linha do objeto.

$$C = \frac{P^2}{A}$$

Figura 10 – Principais operadores de descrição.



Fonte: Esquef, Albuquerque e Albuquerque (2003)

2.5 CLASSIFICAÇÃO E RECONHECIMENTO

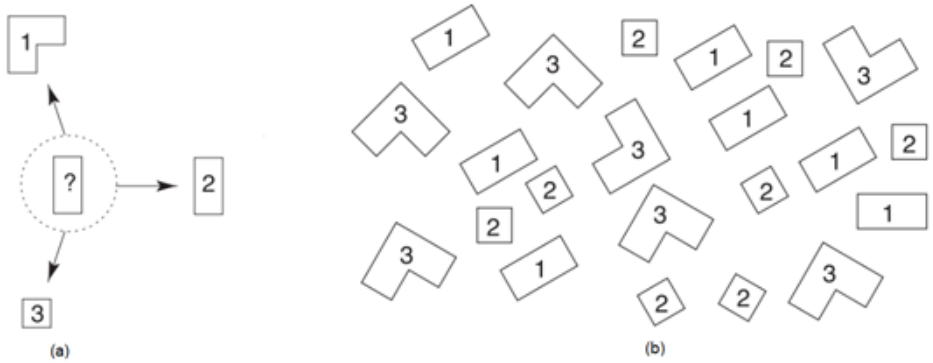
O objetivo desta etapa é reconhecer e rotular os objetos segmentados da imagem, relacionando os atributos extraídos na etapa anterior com amostras que compartilham propriedades semelhantes. Os algoritmos que visam estabelecer o mapeamento entre as propriedades das amostras e o conjunto de rótulos são denotados como algoritmos de classificação ou classificadores (PEDRINI; SCHWARTZ, 2008).

Antes de serem classificados os objetos, é necessário uma etapa de treinamento. Esta etapa utiliza um conjunto de dados chamado conjunto de treinamento. De acordo

com Pedrini e Schwartz (2008), este conjunto apresenta dois componentes, um composto de medidas responsáveis pela descrição de suas propriedades e o segundo representando a classe a qual o objeto pertence.

Quando o treinamento e classificação consideram classes previamente definidas, é chamado de classificação supervisionada. Quando as informações de interesse são obtidas a partir das próprias amostras a serem rotuladas, a classificação e o treinamento são denominados como não supervisionado. Na Figura 11 são demonstrados exemplos de classificação supervisionada e não supervisionada. Atualmente existem diversos classificadores como o classificador Bayesiano e a regra dos k-vizinhos mais próximos. Outra forma de classificação é utilizando Redes Neurais Artificiais, no qual os conceitos aplicados serão detalhados no Capítulo 3.

Figura 11 – Classificação de objetos. (a) supervisionado, (b) não supervisionado



Fonte: Adaptado de Pedrini e Schwartz (2008)

Neste capítulo foram expostos os principais conceitos relacionados a PDI e suas etapas. A etapa de pré-processamento está relacionada à aplicação de técnicas que melhoram a qualidade das imagens brutas, removendo possíveis imperfeições. Já a segmentação, baseada nos princípios da similaridade e descontinuidade dos *pixels*, busca separar a imagem em pedaços menores a fim de se identificar os objetos de interesse nela contidos. Na etapa de pós-processamento, através da aplicação em sequência de filtros morfológicos, são corrigidos eventuais problemas e imperfeições ocorridas durante a etapa anterior. A etapa de extração de atributos consiste na extração de características dos objetos a fim de serem reconhecidos na etapa posterior. Os objetos podem ser representados por suas características externas (bordas), ou características internas (*pixels* que compõem o objeto). Entre os atributos externos podem ser citados o diâmetro, perímetro e a curvatura. Enquanto a área, circularidade, projeção vertical e horizontal são exemplos de atributos internos dos objetos. Na última etapa, de classificação e reconhecimento, os objetos segmentados são rotulados, relacionando os atributos extraídos com amostras que compartilham propriedades semelhantes.

3 REDES NEURAIS ARTIFICIAIS

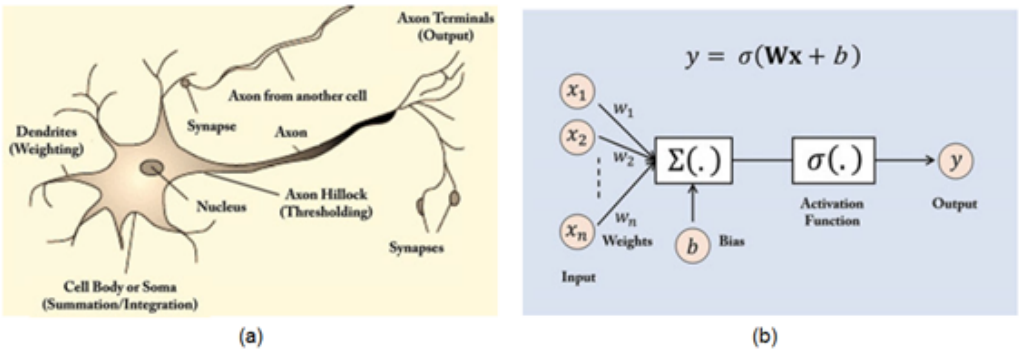
As Redes Neurais Artificiais (RNA) buscam, através de modelos computacionais, reproduzir o modo com que o cérebro humano processa as informações, além da forma de adquirir, armazenar e utilizar o conhecimento experimental. Segundo Haykin (2011), o cérebro é comparado a um computador altamente complexo, não linear e paralelo. Ele tem a capacidade de organizar seus neurônios de forma a realizar processos como reconhecimento de padrões, percepção e controle motor.

Uma RNA se assemelha ao cérebro em dois principais aspectos. O primeiro é a forma de adquirir o conhecimento, que se dá através de um processo de aprendizagem. E o segundo, conforme Haykin (2011), é a força de conexão entre os neurônios, também conhecida como pesos sinápticos, que são utilizados para armazenar o conhecimento adquirido.

A principal característica de uma rede neural é a habilidade de aprender e melhorar o seu desempenho através de um algoritmo de aprendizagem. Neste, um processo iterativo de ajuste é aplicado aos pesos sinápticos, tornando a rede mais inteligente a cada interação.

O cérebro humano contém aproximadamente 100 bilhões de neurônios, sendo esse uma célula nervosa que processa informações. De acordo com Khan *et al.* (2018), é possível traçarmos um paralelo entre um neurônio biológico e um artificial. Um neurônio artificial, também chamado de unidade, possui várias conexões de entrada, assim como os dendritos do neurônio biológico. A estes são atribuídos determinados pesos (análogo às sinapses). A unidade então calcula a soma das entradas ponderadas e aplica uma função de ativação, conforme o corpo celular em um neurônio biológico. O resultado da unidade é então transmitido usando a conexão de saída (função do axônio). Na Figura 12 é demonstrado o funcionamento do neurônio biológico e artificial.

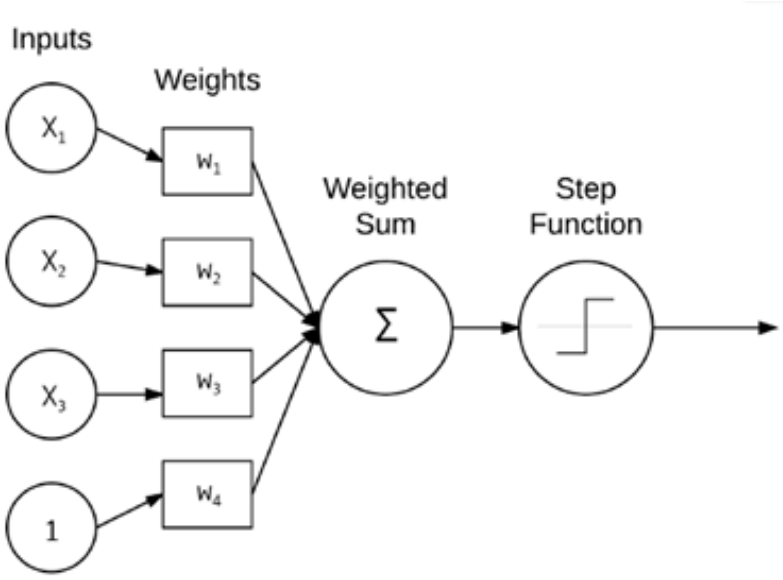
Figura 12 – Neurônio biológico e artificial. (a) biológico; (b) artificial



Fonte: Khan *et al.* (2018)

No ano de 1958 Rosenblatt propôs um novo conceito chamado Perceptron (ROSENBLATT, 1958). Sendo esse o primeiro modelo de aprendizagem supervisionado em uma rede neural. Segundo Haykin (2011), o Perceptron é considerado a rede neural artificial mais simples a ser utilizada com o propósito de classificação de padrões. Basicamente, a rede recebe uma ou mais entradas e gera uma única saída binária, essas entradas são multiplicadas por pesos. Os pesos indicam a importância dada a cada valor de entrada. A saída é determinada pela soma ponderada das entradas multiplicadas pelos pesos, menor ou maior do que algum valor de limiar (*threshold*). Na Figura 13 é demonstrada a arquitetura básica deste tipo de rede.

Figura 13 – Arquitetura de uma rede Perceptron



Fonte: Rosebrock (2017a)

Muitos dos problemas do mundo real não podem ser resolvidos com redes simples que são capazes apenas de separar os dados de forma linear. Para resolver problemas complexos diversas outras redes que receberam o nome de Redes de Aprendizagem Profunda ou *Deep Learning* foram propostas. Dentre elas é possível citar Redes Neurais Convolucionais, Redes Neurais Recorrentes, *Auto Encoders*, *Generative Adversarial Networks*, *Memory Networks*, entre outras.

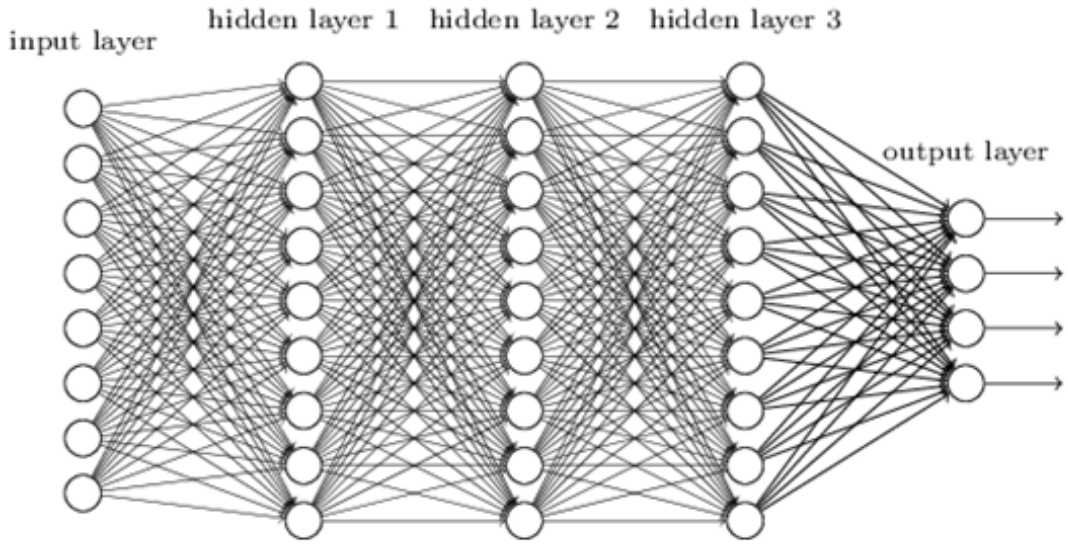
Segundo Khan *et al.* (2018), técnicas de aprendizado profundo são atualmente o estado da arte para a detecção, segmentação, classificação e reconhecimento de objetos em imagens, sendo as Redes Neurais Convolucionais muito utilizadas para este propósito. Na seção a seguir diversos termos relacionados a esse tipo de rede serão abordados.

3.1 REDES NEURAIS CONVOLUCIONAIS

As Redes Neurais Convolucionais, do inglês *Convolutional Neural Networks* (CNN), foram utilizadas com sucesso pela primeira vez por Yann LeCun (LECUN *et al.*, 1998). Desde então, os avanços na área da Visão Computacional foram notórios utilizando este tipo de rede. Classificação de rostos, detecção e segmentação de objetos, conversão de imagens de baixa resolução para alta resolução são exemplos de aplicações de uma CNN.

Uma das característica das redes neurais de aprendizado profundo é possuir diversas camadas. A informação é passada através de cada camada, com a saída da camada anterior fornecendo entrada para a camada posterior. A primeira camada recebe o nome de camada de entrada, enquanto a última é denominada camada de saída. Já as camadas intermediárias são referidas como camadas ocultas (Figura 14). Cada camada pode ter diversos neurônios que podem ser ativados ou desativados através de uma função de ativação.

Figura 14 – Arquitetura CNN

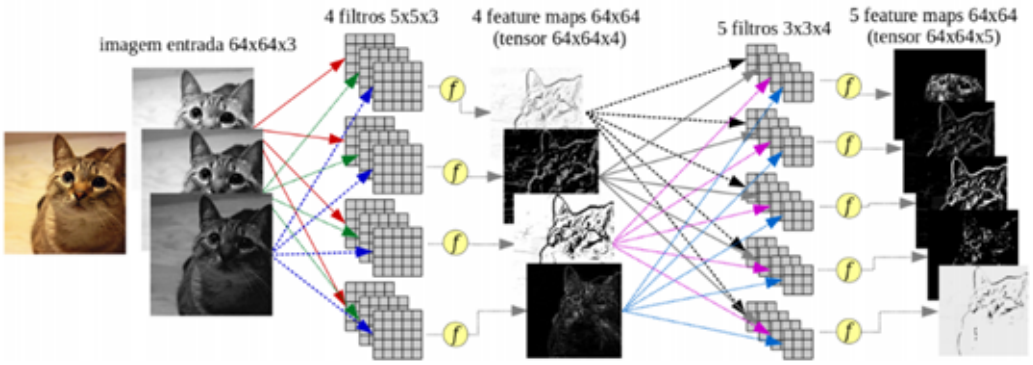


Fonte: Nielsen (2015)

3.1.1 *Convolutional Layer*

Para Khan *et al.* (2018), uma camada convolucional é o componente mais importante de uma CNN. Esta inclui um conjunto de máscaras, também chamadas de filtro ou *kernel*, que são aplicados em uma determinada entrada para gerar como saída um mapa de recursos (*feature map*). A Figura 15 ilustra o processo descrito.

Figura 15 – Geração do mapa de recursos



Fonte: Ponti e Costa (2017)

Em uma camada convolucional os filtros são representados por uma matriz de números discretos (Figura 16). Os pesos dos filtros são inicializados de forma aleatória, depois os pesos são ajustados em um processo chamado de *backpropagation*, conforme às várias iterações diferentes que ocorrem no procedimento de aprendizagem.

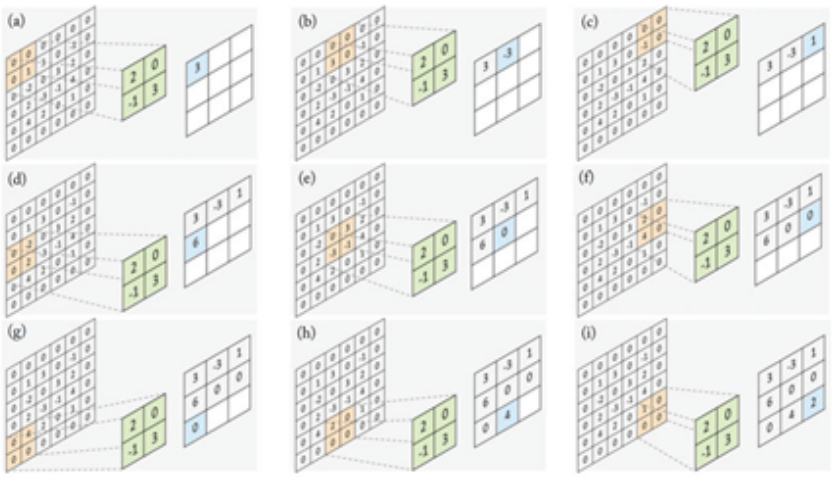
Figura 16 – Máscara convolucional 2x2

2	0
-1	3

Fonte: Khan *et al.* (2018)

Muito semelhante ao processo descrito na subseção 2.1.4, os filtros geralmente são quadrados que vão deslizando por toda a imagem, multiplicando os valores da máscara por seu correspondente na matriz de entrada, somando o resultado da multiplicação de seus vizinhos e armazenando em um mapa de recursos, repetindo este procedimento até que toda imagem tenha sido percorrida. Desta forma, realça características importantes presentes na imagem. Na Figura 17 é ilustrado este processo aplicando a máscara presente na Figura 16.

Figura 17 – Filtro convolucional



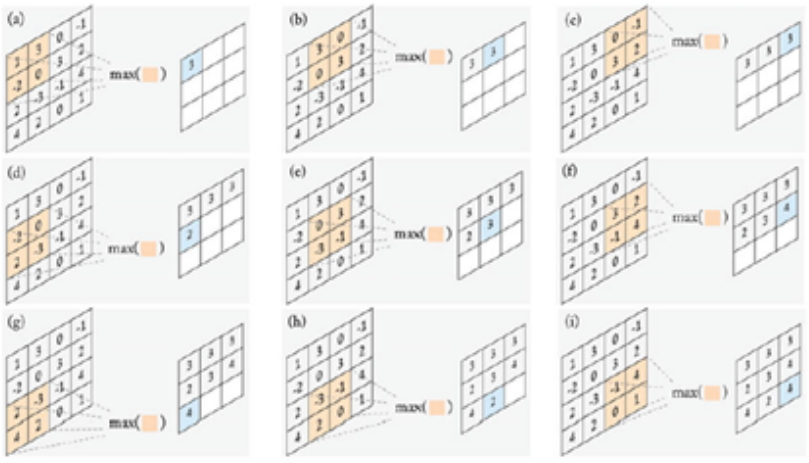
Fonte: Khan *et al.* (2018)

3.1.2 *Pooling Layer*

Segundo Rosebrock (2017a), a função principal da camada *pooling* é reduzir progressivamente o tamanho do espaço (largura x altura) do mapa de recursos de entrada. Permite, desta forma, uma redução na quantidade de parâmetros e no esforço computacional da rede.

As operações realizadas neste camada são definidas por uma função de agrupamento como média ou máxima. O *pooling* máximo é normalmente utilizado nas camadas do meio da arquitetura, enquanto o *pooling* médio é mais usado na camada final da rede. Na Figura 18 é apresentado um exemplo do processo de *pooling* máximo, onde em uma máscara de 2x2 o valor em evidência é substituído pelo valor máximo de seus vizinhos.

Figura 18 – Processo de max pooling



Fonte: Khan *et al.* (2018)

3.1.3 *Fully Connected Layer*

Camadas totalmente conectadas, do inglês *fully connected layer*, correspondem essencialmente a camadas de convolução com filtros de tamanho 1 x 1. Cada unidade em uma camada totalmente conectada é densamente conectada a todas as unidades da camada anterior (KHAN *et al.*, 2018). Geralmente esta camada é colocada no final da arquitetura. Por sua vez, esta é essencialmente idêntica aos pesos de um *Perceptron*, onde uma função de ativação é utilizada para predizer a classe de um objeto.

3.1.4 Função de Ativação

As camadas de peso como as convolucionais e totalmente conectadas geralmente são seguidas por uma função de ativação não linear. A aplicação de uma função não linear após as camadas de peso é muito importante, pois permite uma rede neural aprender mapeamentos não lineares. Na ausência de não linearidades, uma rede empilhada de camadas de peso é equivalente a um mapeamento linear do domínio de entrada para o domínio de saída (KHAN *et al.*, 2018).

Basicamente uma rede neural sem a função de ativação se torna apenas um modelo de regressão linear. Na prática, a função de ativação decide se a informação que o neurônio está recebendo é relevante ou deve ser ignorada. A seguir serão exibidas algumas das funções de ativação mais utilizadas em *deep learning*.

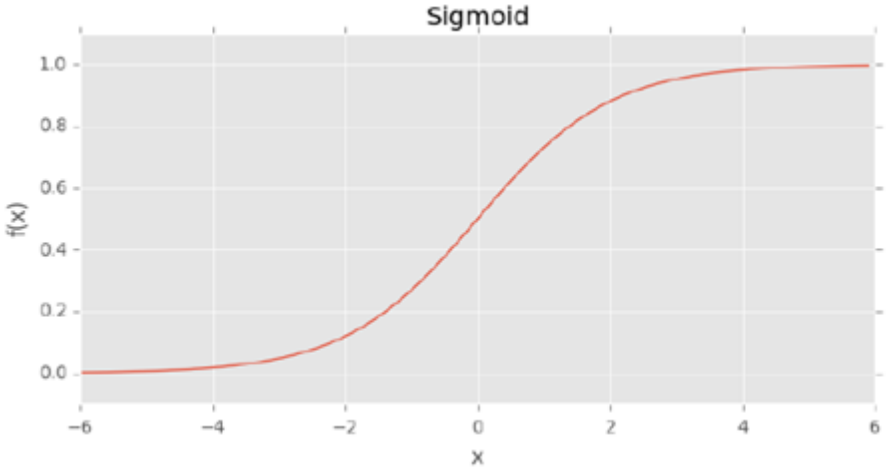
3.1.4.1 Sigmóide

A função sigmóide recebe um número real como sua entrada, e gera um número entre 0 e 1. Sendo definida por:

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}.$$

De acordo com Nielsen (2015), se a entrada for um número grande positivo, a saída da função será aproximadamente 1, já nos casos onde a entrada for negativa, o resultado será próximo a 0, semelhante ao resultado de um *Perceptron*. Somente quando a entrada é de tamanho modesto é que há muito desvio do modelo *Perceptron*. A função sigmoid pode ser plotada como:

Figura 19 – Ilustração gráfica da função sigmoid



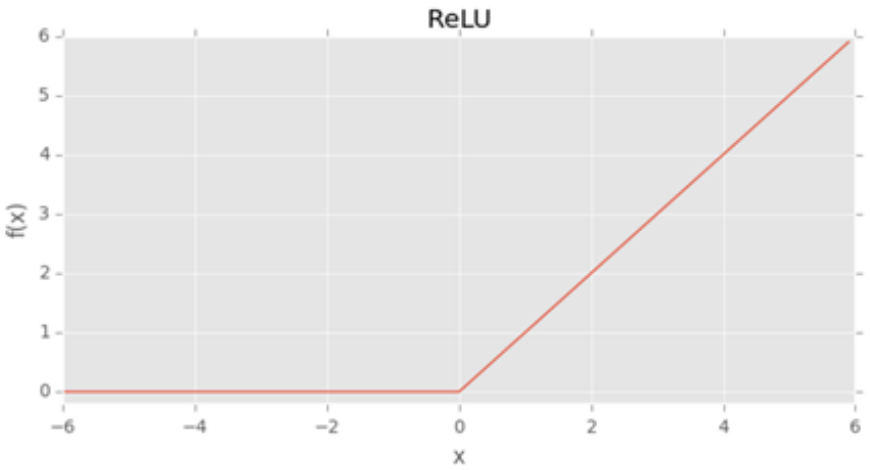
Fonte: Adaptado de Rosebrock (2017a)

3.1.4.2 ReLU

A função ReLU e suas variações são amplamente utilizadas em *deep learning*. Uma função ReLU mapeia a entrada para 0 se a entrada for negativa e mantém seu valor inalterado se for positivo (KHAN *et al.*, 2018). Desta forma, apenas alguns neurônios são ativados, tornando a rede esparsa e computacionalmente eficiente. A Figura 20 demonstra graficamente a função, enquanto matematicamente pode ser representada por:

$$f_{relu}(x) = \max(0, x).$$

Figura 20 – Ilustração gráfica da função ReLU



Fonte: Adaptado de Rosebrock (2017a)

3.1.4.3 Softmax

A função Softmax é extremamente útil em problemas que lidam com classificação. É utilizada na camada de saída para gerar as probabilidades dos objetos de se enquadrarem em cada classe. Segundo Chollet (2017), para um problema que existam N classes, a função retorna um vetor de N posições, contendo probabilidades para cada uma destas N classes. A posição do vetor que possuir a maior probabilidade representa a classe de saída.

3.1.5 Função de Perda

No contexto de classificação a função de perda quantifica o quão bom ou ruim foram as previsões das classes. Este processo é de extrema importância durante o treinamento e demonstra a curva de aprendizado da rede. A seguir será abordada uma função de custo muito utilizada na construção de redes de aprendizado profundo.

3.1.5.1 Entropia cruzada

Entropia cruzada, do inglês *cross entropy*, é geralmente utilizada para resolver o problema da desaceleração do aprendizado, onde pretende quantificar a diferença entre duas distribuições de probabilidade. É matematicamente expressa por:

$$C = -\frac{1}{n}\sum_{x}[y \ln a + (1-y)\ln(1-a)]$$

Onde n é o número total de itens de dados de treinamento, a soma é sobre todas as entradas de treinamento x, e y é a saída desejada correspondente. Em suma, a entropia cruzada é positiva e tende a zero, à medida que o neurônio melhora a computação da saída desejada, y, para todas as entradas de treinamento x (NIELSEN, 2015).

3.1.6 Processo de Aprendizagem

Como citado na seção 2.5, o treinamento pode ser supervisionado ou não supervisionado. Em uma CNN outros fatores incidem no treinamento. A seguir aspectos referentes ao processo de treinamento e aprendizagem das redes serão abordados.

3.1.6.1 Inicialização dos pesos

Uma inicialização de peso correta é a chave para treinar de forma estável em redes muito profundas. Uma inicialização inadequada pode levar ao problema de gradiente desaparecendo ou explodindo durante a retropropagação do erro (KHAN *et al.*, 2018).

Atualmente a forma mais comum de gerar os pesos iniciais é através de números aleatórios sorteados a partir de uma distribuição Gaussiana. Mas outras técnicas também

são utilizadas como a *uniform random, orthogonal random, unsupervised pre-training*, entre outras.

3.1.6.2 Backpropagation

O *Backpropagation* é utilizada para atualizar os pesos nas camadas convolucionais e totalmente conectada durante o treinamento da rede. Basicamente consiste em dois passos. O passo para frente ou *forward pass*, onde as entradas são passadas através da rede e as previsões de saída obtidas. E o passo para trás, ou *backward pass*, onde é calculada a função de perda da camada e seu resultado é usado recursivamente para atualizar os pesos das camadas anteriores.

3.1.6.3 Overfitting

O *Overfitting* ocorre quando a rede perde a capacidade de generalizar, ou seja, está apenas memorizando o conjunto de treinamento. Perde, assim, a capacidade de classificar informações que não estão no conjunto de dados de treinamento.

3.1.6.4 Dropout

Segundo Ponti e Costa (2017), o *dropout* é uma técnica para minimizar *overfitting* que, na fase de treinamento e durante o passo de propagação dos dados pela rede, aleatoriamente desativa proporcionalmente alguns neurônios. Isso faz com que os neurônios que sobram tenham que modificar os seus pesos para encontrar a resposta, evitando assim, a memorização das imagens.

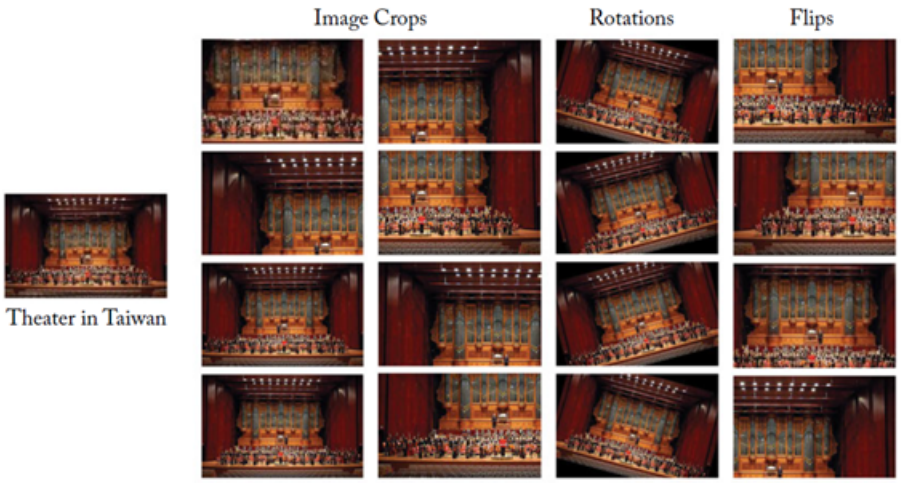
3.1.6.5 Batch normalization

De acordo com Ioffe e Szegedy (2015), a *Batch normalization* permite usar taxas de aprendizagem muito mais altas e que se tenha menos problemas com a inicialização dos pesos. Também atua como regularizador, em alguns casos eliminando a necessidade da utilização do *Dropout*.

3.1.6.6 Data-augmentation

A melhor forma de generalizar uma rede é obtendo um grande número de dados para treinamento. Uma das formas de aumentar o conjunto de treinamento é transformando os dados já disponíveis. Se tratando de imagem, operações simples como rotações, recorte, inversão, dimensionamento, translação, entre outras podem ser utilizadas. A Figura 21 demonstra algumas das operações que podem ser aplicadas.

Figura 21 – Data-augmentation



Fonte: Khan *et al.* (2018)

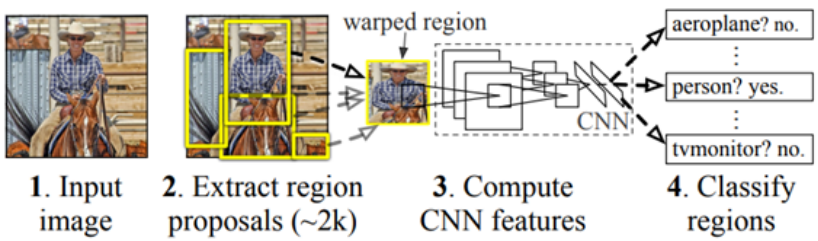
3.2 *REGION-BASED CONVOLUTIONAL NETWORK* (R-CNN)

A R-CNN (GIRSHICK, 2015) é uma rede com o propósito de resolver basicamente dois objetivos. Localizar objetos em imagens e treinar um modelo de alta capacidade com uma pequena quantidade de imagens anotadas. Além de classificar os objetos na imagem de entrada, a rede busca detectar a localização dos objetos classificados, gerando assim, uma caixa delimitadora ao seu redor.

O método gera cerca de 2.000 propostas de regiões independentes de categoria para a imagem de entrada, extrai um vetor de característica de comprimento fixo de cada proposta usando uma CNN e, em seguida, classifica cada região com SVMs lineares de categorias específicas (GIRSHICK, 2015). As propostas de regiões são locais de uma imagem em que existe uma grande probabilidade de conter um objeto relevante. A Figura 22 demonstra o método R-CNN.

De acordo com Girshick (2015), os métodos de proposta de região normalmente dependem de recursos baratos e esquemas de inferência econômicos como a busca seletiva, que utiliza *superpixels* com base em recursos de baixo nível para realizar esta tarefa.

Figura 22 – *Region-based Convolutional Network*



Fonte: Girshick (2015)

A partir deste método proposto, surgiram arquiteturas que trouxeram melhorias consideradas para os processos de classificação, detecção e segmentação de imagens. Entre elas é possível citar o Fast R-CNN (GIRSHICK, 2015), Faster R-CNN (REN *et al.*, 2015) e Mask R-CNN (HE *et al.*, 2017), entre outros. Estas arquiteturas introduziram conceitos e técnicas que são utilizadas atualmente em Visão Computacional, as quais algumas serão explicadas a seguir.

3.2.1 *Region Proposal Network* (RPN)

As redes baseadas em região, como citado anteriormente, originalmente utilizavam, na maioria dos casos, uma busca seletiva para determinar as propostas de regiões. Porém, este método necessitava de um grande esforço computacional. Pensando nisso, Ren *et al.* (2015), propuseram uma mini rede chamada RPN, que melhorou de forma satisfatória o tempo de processamento para determinar as propostas de regiões.

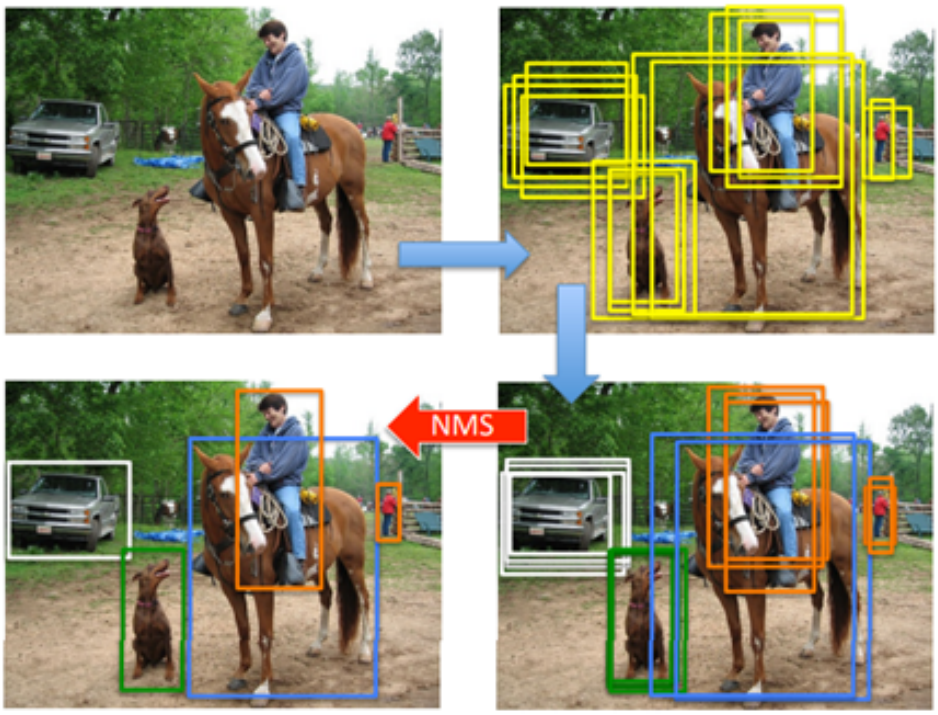
A rede recebe como entrada um *feature map* gerado em uma etapa anterior, desta forma, acaba não processando a imagem em si, mas sim recursos compartilhados entre as camadas. Permite uma reutilização eficiente dos recursos, evitando, portanto, cálculos desnecessários. Isso ocasiona uma diminuição do esforço computacional da arquitetura como um todo.

Cada ponto no mapa de recurso é chamado de âncora. A rede é treinada para gerar caixas de âncora para cada âncora do mapa de recursos. Segundo Ren *et al.* (2015), a rede é implementada com uma camada convolucional n×n seguida por duas camadas convolucionais 1×1. Uma para gerar a camada REG (*Regression coefficients for boxes layer*) e a outra para gerar a camada CLS (*Classification Scores Layer*).

A camada CLS indica a probabilidade da caixa de âncora conter um determinado objeto, enquanto a camada REG é formada pelas coordenadas x, y, largura e altura do canto superior esquerdo da imagem até a caixa delimitadora. Estas coordenadas são necessárias para que seja possível gerar as caixas delimitadoras na imagem original.

No caso de haver âncoras sobrepostas, o algoritmo de supressão não máxima, *non-maximum suppression* (NMS) é utilizado para determinar que fiquem apenas as âncoras com probabilidades mais altas de conter um objeto de primeiro plano. A Figura 23 demonstra a sua utilização para melhorar a precisão na predição das caixas delimitadoras. A cada proposta de região é atribuída uma pontuação por um classificador e logo após, a supressão não máxima é aplicada para determinar as detecções.

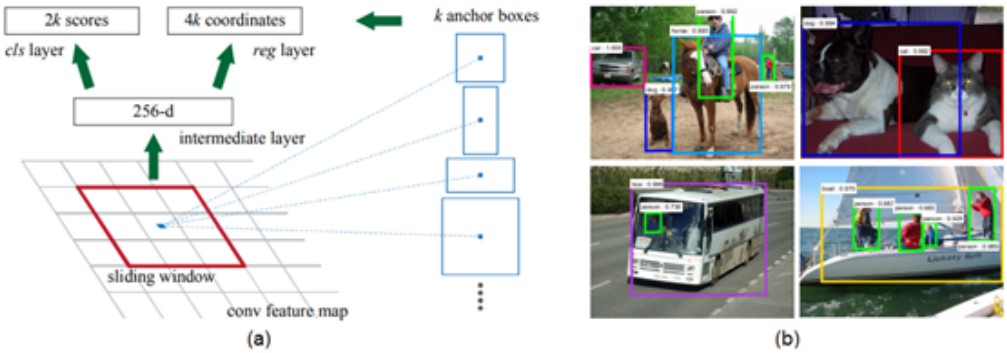
Figura 23 – *Non-maximum suppression* (NMS)



Fonte: Bodla *et al.* (2017)

De acordo com Ren *et al.* (2015), a RPN é, portanto, um tipo de rede totalmente convolucional, do inglês, *fully convolutional network* (FCN), que pode ser treinada de ponta a ponta, especificamente para a tarefa de geração de propostas de regiões, também chamadas de *Region of Interest* (ROI). A Figura 24 representa o processo de geração das propostas de regiões.

Figura 24 – *Region Proposal Network* (RPN). (a) Arquitetura RPN; (b) Caixas delimitadoras previstas

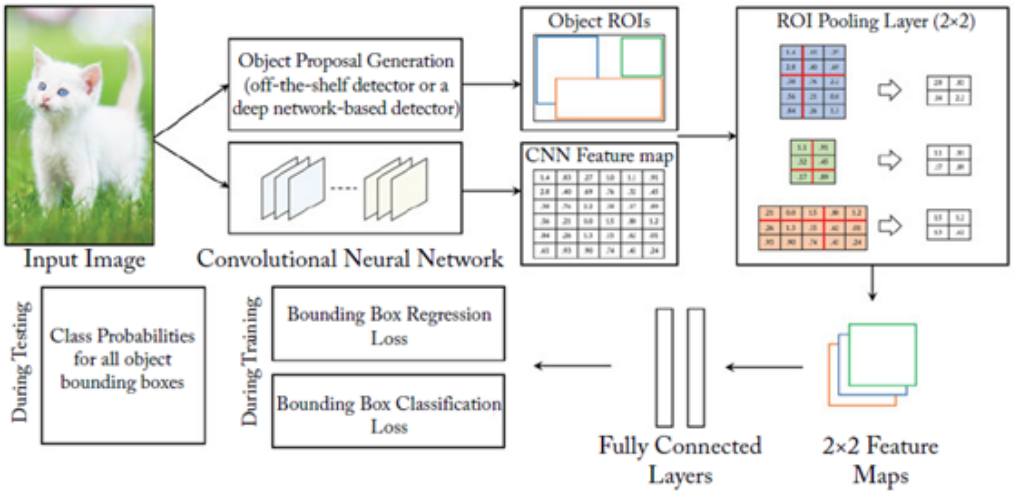


Fonte: Ren *et al.* (2015)

Os recursos obtidos podem ter tamanhos espaciais diferentes porque cada ROI pode ser de uma dimensão diferente. Uma vez que as camadas CNN só podem operar

como entradas dimensionais fixas, uma camada de ROI *pooling* converte esses mapas de recursos de tamanho variável para um tamanho fixo (KHAN *et al.*, 2018). A Figura 25 ilustra a arquitetura da rede.

Figura 25 – ROI *pooling*



Fonte: Khan *et al.* (2018)

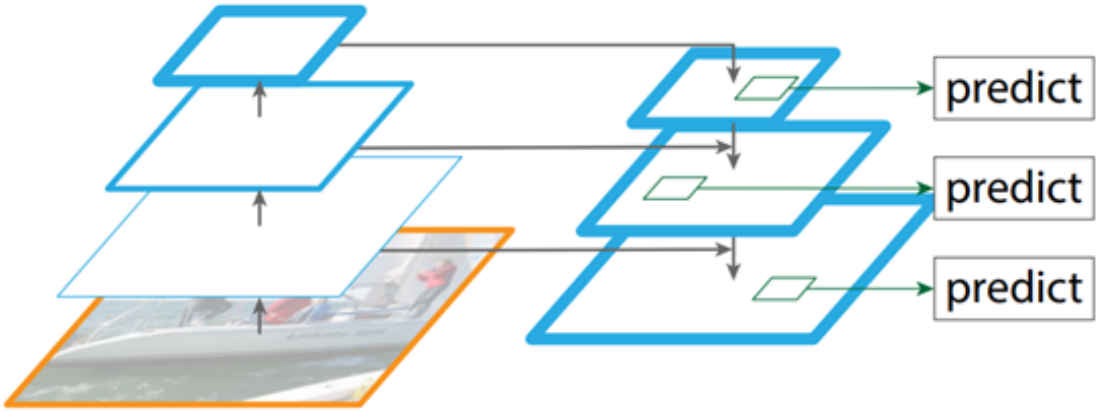
3.2.2 *Feature Pyramid Network* (FPN)

Detectar objetos em escalas diferentes sempre foi considerado um problema na Visão Computacional. Até então este problema era resolvido através da utilização de uma pirâmide da mesma imagem em diferentes escalas. No entanto, o processamento de imagens em várias escalas é demorado e consome uma alta quantidade de memória.

O FPN (LIN *et al.*, 2016) é um extrator de recursos focado em precisão e velocidade que foi desenvolvido para substituir o extrator de recursos de detectores como o Faster R-CNN. Ao contrário dos extratores comuns, que geram uma pirâmide da mesma imagem em diferentes escalas para detectar os objetos, a rede gera os mapas de recursos em diversas escalas. Constrói, desta forma, mapas de recursos semânticos de alto nível em todas as escalas.

De acordo com Lin *et al.* (2016), a construção da pirâmide envolve um caminho de baixo para cima, um caminho de cima para baixo e conexões laterais. O caminho de baixo para cima é utilizado para extração dos recursos. A medida que o caminho é avançado, a dimensão espacial é reduzida pela metade da camada anterior e o valor semântico de cada camada aumenta. Já o caminho de cima para baixo é utilizado para reconstruir as camadas com resolução mais alta a partir de uma camada com alto valor semântico. Mesclando os mapas de recursos do caminho de baixo para cima e de cima para baixo por adição de elementos nas conexões laterais. Desta forma, gerando predições em todas as camadas da rede. A Figura 26 ilustra o processo descrito.

Figura 26 – *Feature Pyramid Network* (FPN)



Fonte: Lin *et al.* (2016)

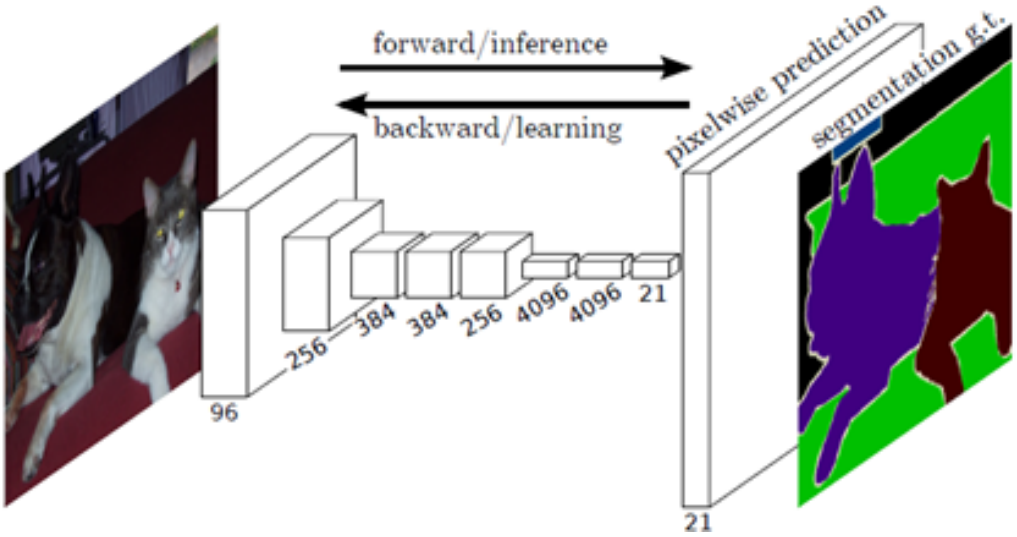
3.3 *FULLY CONVOLUTIONAL NETWORK* (FCN)

De acordo com Khan *et al.* (2018), as redes de classificação típicas recebem como entrada imagens de tamanho fixo e produzem mapas de saída não espaciais e logo após utilizam uma função softmax para realizar a classificação. Neste caso, a informação espacial é perdida devido a essas redes usarem camadas totalmente conectadas de dimensão fixa em sua arquitetura.

Já as redes totalmente convolucionais, do inglês, *Fully Convolutional Network* (FCN), transformam as camadas totalmente conectadas em uma camada convolucional 1x1 com *kernels* que cobrem todas as suas regiões de entrada. Produzem assim, mapas espaciais a partir de imagens de entrada de qualquer tamanho. Desta forma, torna este tipo de rede ideal para a tarefa de segmentação semântica de imagens.

A rede proposta por Long, Shelhamer e Darrell (2014) é utilizada para segmentar objetos em imagens através de predições *pixels* por *pixels*. Onde camadas convolucionais são usadas para extrair recursos da imagem, em seguida, uma camada convolucional 1x1 é aplicada e, finalmente, são transformadas a altura e largura do mapa de recursos para o tamanho da imagem de entrada usando a camada de convolução transposta (também chamada de deconvolução).

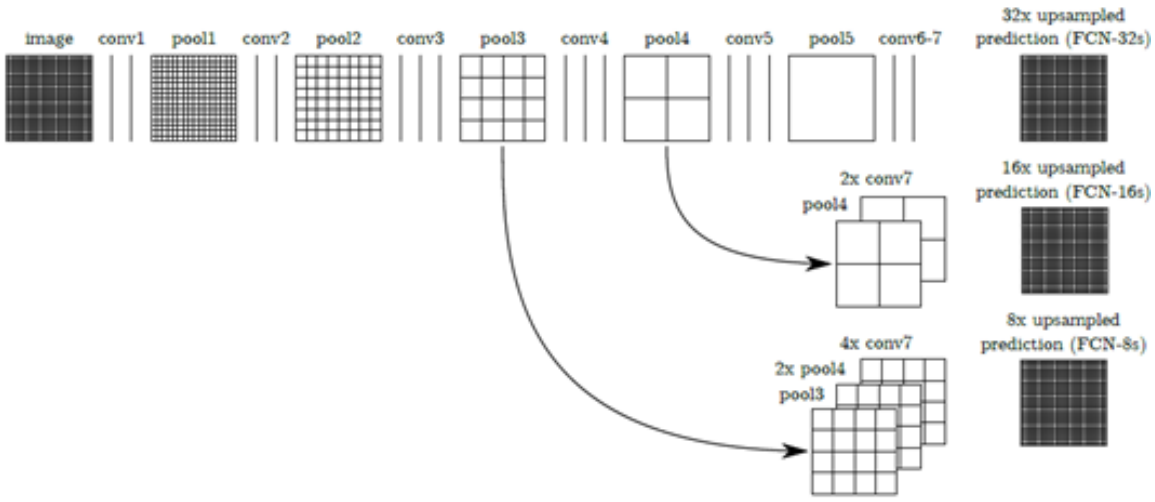
Figura 27 – *Fully Convolutional Network* (FCN)



Fonte: Long, Shelhamer e Darrell (2014)

Os mapas de recursos gerados pela rede são expandidos em múltiplos de 32. O que limita o nível de detalhes na saída. De acordo com Long, Shelhamer e Darrell (2014), para resolver este problema, o FCN aprende a combinar as informações da camada rasa, que contém recursos grosseiros e da camada profunda, que possui recursos finos. Desta forma, os recursos são expandidos em 16x e 8x. O que melhora de forma considerável a precisão da máscara prevista. A Figura 28 ilustra a arquitetura da rede.

Figura 28 – *Arquitetura FCN*

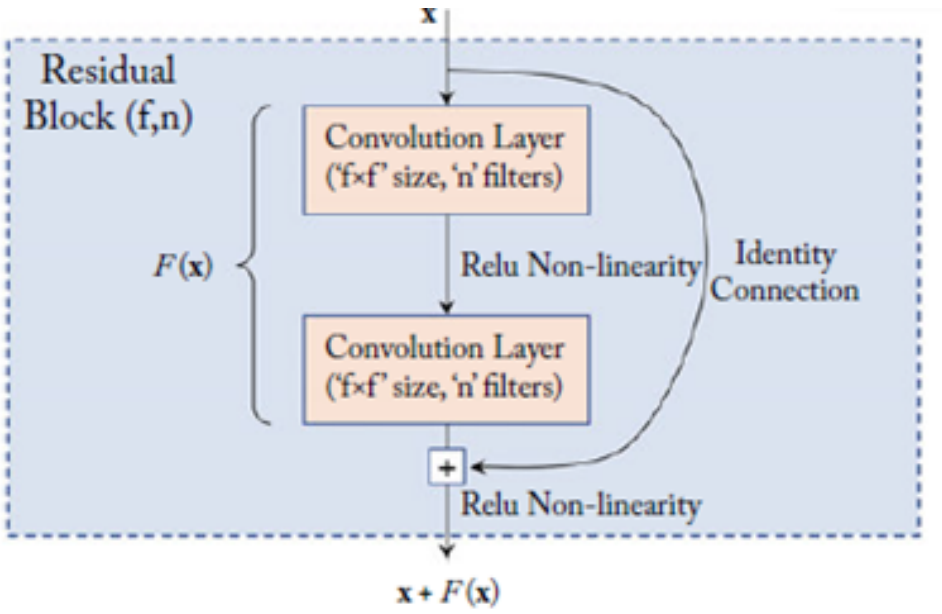


Fonte: Long, Shelhamer e Darrell (2014)

3.4 RESNET

De acordo com Rosebrock (2017b), ResNet é extremamente importante na história da aprendizagem profunda, pois introduziu o conceito de módulos residuais e mapeamentos de identidade. Esses dois conceitos permitiram treinar redes muito profundas, o que não era possível até então. A rede foi Introduzida pela primeira vez por He *et al.* (2015), em um artigo chamado *Deep Residual Learning for Image Recognition* e posteriormente atualizado em He *et al.* (2016), no artigo chamado *Identity Mappings in Deep Residual Networks*.

Figura 29 – Bloco Residual

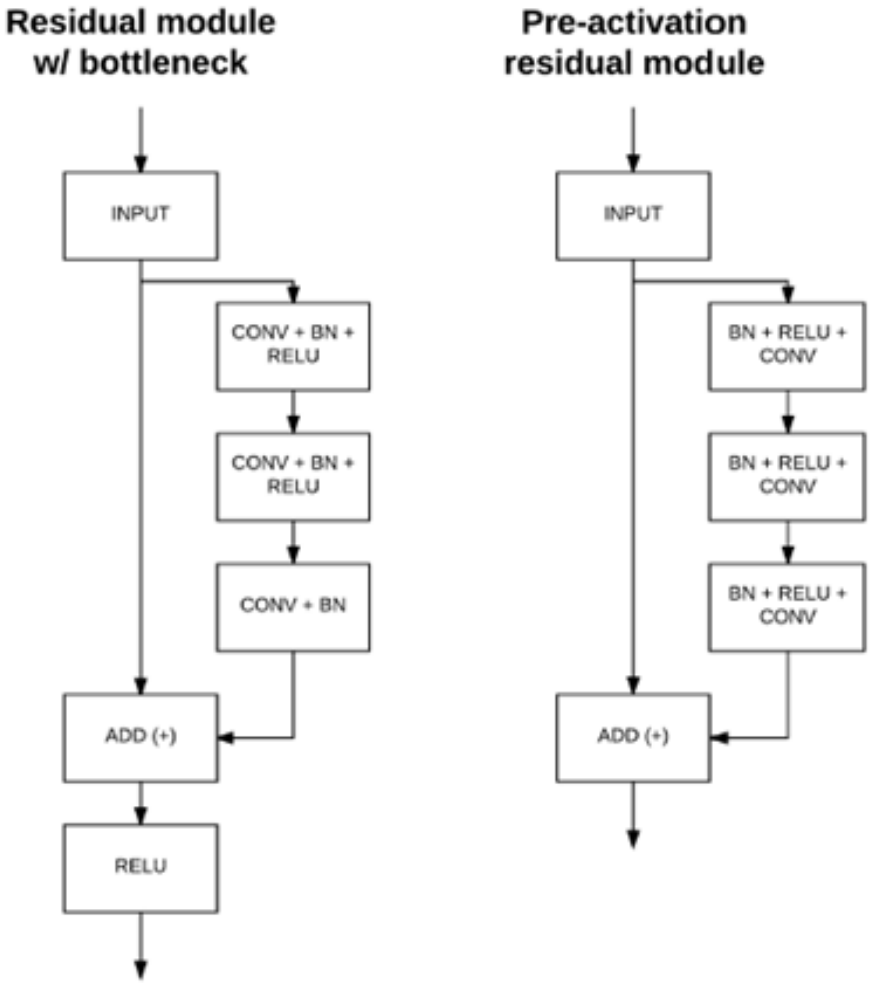


Fonte: Khan *et al.* (2018)

Os blocos residuais (Figura 29) implementam uma função de transformação que adiciona na entrada original a saída gerada pela camada convolucional junto com a função ReLU e de uma normalização através de BN. O processo responsável pelas operações de convolução, ReLU e BN recebem o nome de mapeamento de identidade. Enquanto a operação de adição é chamada de conexão residual.

A Figura 30 demonstra a função de transformação em um bloco residual no módulo original e no módulo que utiliza uma pré-ativação, que nada mais é do que uma alteração na ordem das operações no mapeamento de identidade. Tal alteração permitiu redes mais profundas ainda.

Figura 30 – Módulo residual



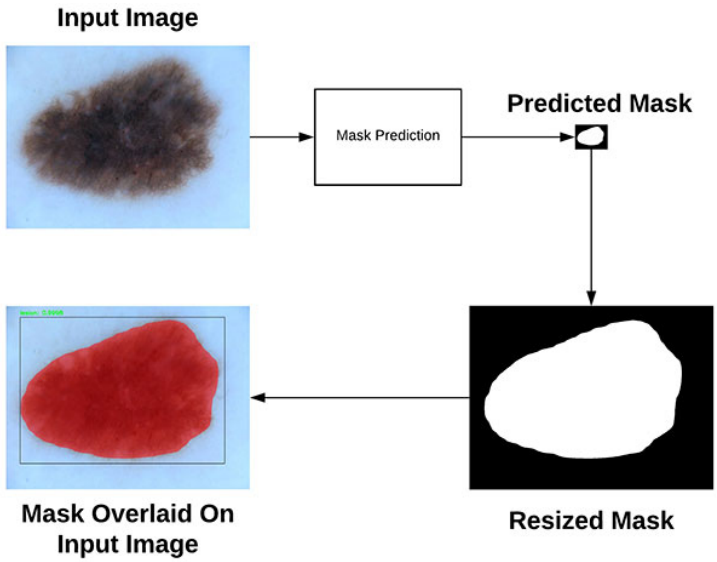
Fonte: Rosebrock (2017b)

Na prática, a arquitetura alcança um aprendizado estável em modelos muito profundos. A razão é que o mapeamento de recurso residual é muitas vezes mais simples do que o mapeamento não referenciado aprendido nas arquiteturas convencionais (KHAN *et al.*, 2018).

3.5 MASK R-CNN

Mask R-CNN (HE *et al.*, 2017) é um *framework* para segmentação de instância de objetos que estende o *framework* Faster R-CNN (REN *et al.*, 2015), adicionando uma ramificação para prever uma máscara dos objetos existentes, além do reconhecimento da caixa delimitadora dos objetos detectados. O Mask R-CNN (HE *et al.*, 2017), tem como resultado três saídas para cada objeto detectado. O rótulo da classe, a localização da caixa delimitadora e a máscara do objeto. Diferente do Faster R-CNN (REN *et al.*, 2015) que apresenta duas saídas para cada objeto detectado (rótulo da classe e a localização da caixa delimitadora).

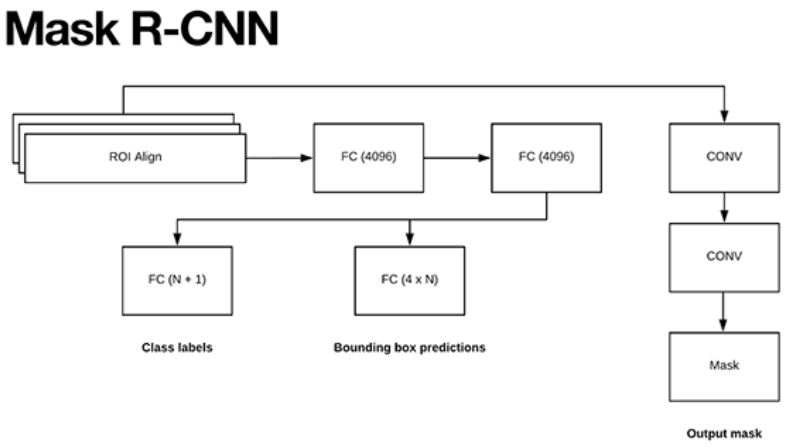
Figura 31 – Exemplo de máscara gerada pelo framework Mask R-CNN



Fonte: Rosebrock (2017b)

Primeiro utilizando a rede RPN, o framework gera a proposta de regiões onde pode haver um objeto. Logo após, prevê a classe do objeto e a caixa delimitadora. Por último, gera uma máscara em nível de *pixel* do objeto com base na proposta da primeira etapa. Nestas três últimas etapas é utilizada uma camada chamada *ROIAlign*, que preserva as informações espaciais que ficam desalinhadas no caso do *ROI pooling* utilizado pelo Faster R-CNN (REN *et al.*, 2015). A Figura 32 ilustra a camada *ROIAlign*.

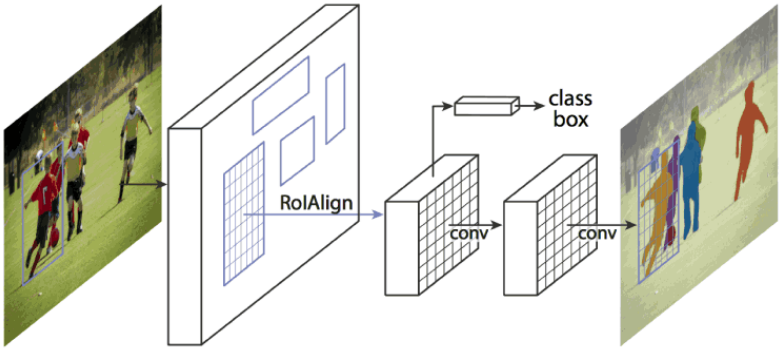
Figura 32 – *Camada ROIAlign*



Fonte: Rosebrock (2017b)

De acordo com He *et al.* (2017), os melhores resultados do *framework* foram utilizando em seu *backbone* as redes de aprendizado profundo ResNet para extração de recursos brutos e FPN para extração de recursos ricos semanticamente em várias escalas de resolução. A Figura 33 demonstra a arquitetura deste *framework*.

Figura 33 – Arquitetura do *framework* Mask R-CNN

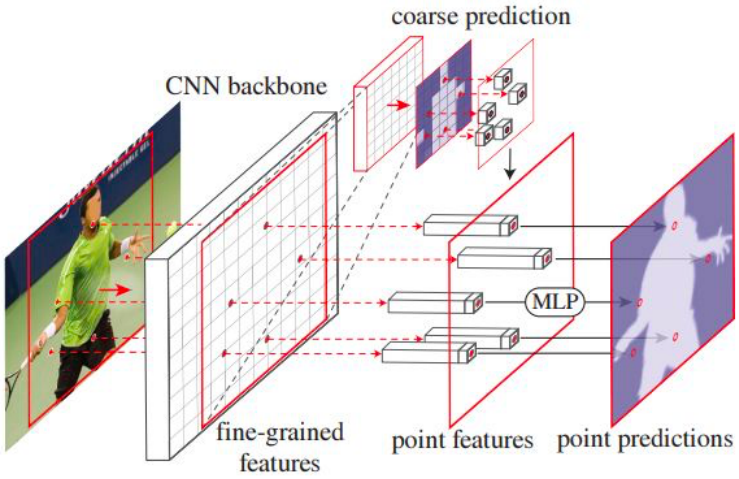


Fonte: He *et al.* (2017)

3.6 POINTREND

PointRend (KIRILLOV *et al.*, 2019) é um método desenvolvido por pesquisadores do Facebook, que pretende melhorar a eficiência na segmentação de imagens em objetos complexos, como dedos, por exemplo. Introduziu o conceito de renderização baseada em pontos, que executa as previsões de segmentação em locais selecionados de forma adaptativa com base em um algoritmo de subdivisão iterativa. PointRend pode ser aplicado de forma flexível tanto à segmentação de instância quanto à semântica, permitindo a integração a modelos como o Mask R-CNN He *et al.* (2017). A Figura 34 demonstra a arquitetura proposta pelos autores.

Figura 34 – Arquitetura PointRend

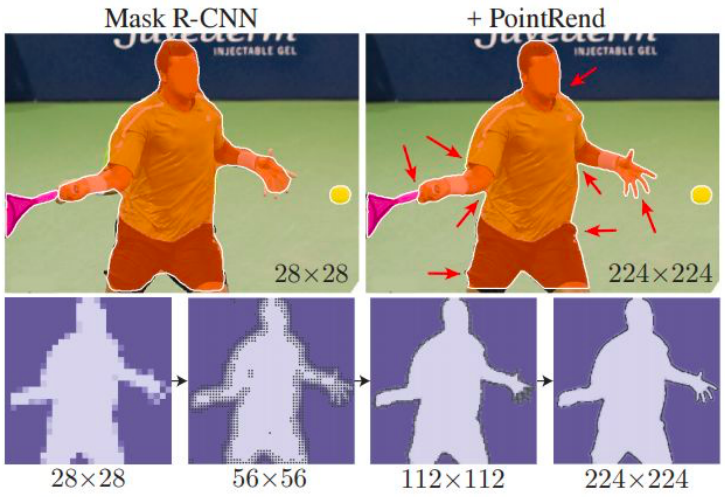


Fonte: Kirillov *et al.* (2019)

PointRend encara a tarefa de segmentar imagens como um problema de renderização, adaptando ideias clássicas de renderização gráfica, como a subdivisão. A técnica de subdivisão produz um padrão de amostragem do tipo *quadtree* que renderiza de forma

eficiente uma imagem com suavização do serrilhado em alta resolução (KIRILLOV *et al.*, 2019). A Figura 35 demonstra os resultados do modelo Mask R-CNN tradicional e o PointRend proposto por Kirillov *et al.* (2019).

Figura 35 – Comparação Mask R-CNN x PointRend



Fonte: Kirillov *et al.* (2019)

Neste capítulo foram abordados conceitos pertinentes a Redes Neurais Artificiais, bem como a sua relação com o cérebro humano. A RNA se assemelha ao cérebro em dois principais aspectos. A forma de adquirir o conhecimento, que se dá através de um processo de aprendizagem. E a força de conexão entre os neurônios, que são utilizados para armazenar o conhecimento adquirido. A CNN é um tipo de rede de aprendizado profundo amplamente utilizada em Visão Computacional. Uma das características desse tipo de rede é possuir diversas camadas. Entre elas é possível citar: *Convolutional Layer*, *Pooling Layer* e *Fully Connected Layer*. Neste capítulo também foram citadas algumas arquiteturas que introduziram conceitos e técnicas que são utilizadas atualmente em Visão Computacional. Entre elas, a arquitetura R-CNN (GIRSHICK, 2015), que propõe uma mini rede chamada RPN, que melhorou o tempo de processamento para determinar as propostas de regiões onde pudesse conter um objeto. O extrator de recursos FPN (LIN *et al.*, 2016), que gera mapas de recursos semânticos de alto nível em todas as escalas, combinado com a rede FCN (LONG; SHELHAMER; DARRELL, 2014), que permitiu segmentar objetos em imagens através de predições *pixels* por *pixels*. A utilização destas técnicas permitiu melhorar de forma satisfatória a segmentação dos objetos em imagens. Por último, foram apresentados os *frameworks* Mask R-CNN (HE *et al.*, 2017) e PointRend (KIRILLOV *et al.*, 2019) que utilizam os conceitos e técnicas citadas e são utilizados na elaboração do protótipo descrito no Capítulo 5.

4 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados à área de pesquisa deste projeto. O objetivo é identificar e avaliar os métodos propostos por outros autores que tratam do mesmo tema.

4.1 *BEEF CATTLE INSTANCE SEGMENTATION USING MASK R-CNN (DANISH, 2018)*

Segundo Danish (2018), existe uma complexidade geral das atividades do dia a dia de uma fazenda, sendo no manejo de animais onde são empregados os maiores esforços. A avaliação visual individual nos animais do rebanho se torna ineficiente e requer grande investimento de tempo.

Esta é uma situação em que as aplicações baseadas em visão computacional podem resolver problemas como o rastreamento das atividades dos bovinos, encontrando a distribuição espacial deles, fornecendo assim, informações valiosas que podem ajudar a detectar o comportamento social dos animais. De acordo como Danish (2018), a aplicação da segmentação de instância usando o algoritmo Mask R-CNN (HE *et al.*, 2017) tem potencial para alcançar bons resultados no monitoramento em tempo real na pecuária.

O objetivo da pesquisa de Danish (2018) foi construir um modelo de segmentação de instância para a segmentação do corpo de vacas por meio de ajuste fino do algoritmo Mask R-CNN. Executou o seu modelo em um conjunto de dados de vacas que foi preparado e extraído através da utilização de uma câmera de monitoramento (Figura 36).

Figura 36 – Ilustração do dataset de imagens do trabalho de Danish(2018)

Fonte: Danish (2018)

As imagens utilizadas por Danish (2018) foram preparadas a partir de animais em confinamento, com baixa iluminação e oclusão parcial entre as vacas. 27 imagens com dimensões de 720x1280 com 10 vacas cada foram utilizadas, totalizando 270 animais, das quais 70% foram usadas para o treinamento do modelo e 30% para a validação. Após a sua aquisição, as imagens foram anotadas utilizando uma ferramenta de anotação gráfica. O resultado é demonstrado na Figura 37 .

Figura 37 – Ilustração da representação da máscara da imagem do trabalho de Danish(2018).

Fonte: Danish (2018)

O conjunto de dados estava desequilibrado em relação ao número de pixels que representam vacas e o fundo. Para resolver esse problema o autor submeteu as imagens brutas a um pré-processamento chamado normalização, onde é alterada a faixa de valores de intensidade de pixel para valores entre 0 e 255, logo após, as imagens foram convertidas em 8 bits antes de serem armazenadas em disco novamente.

A implementação da arquitetura de seu modelo usou na camada de *backbone* o extrator de recursos FPN (LIN *et al.*, 2016) combinado com a rede neural ResNet101 (HE *et al.*, 2015) que detecta recursos em nível superior e inferior. Logo após, foi utilizada uma rede neural leve chamada RPN (REN *et al.*, 2015), que é responsável por fazer uma varredura na imagem buscando a área onde encontram-se as vacas.

A varredura RPN sobre uma região é chamada de âncora. O algoritmo escolhe onde provavelmente conterá animais e refina sua localização. Se houver várias âncoras sobrepostas, aquela com a pontuação de primeiro plano mais alta será selecionada em um processo conhecido como NMS.

As âncoras podem ter diferentes tamanhos no mapa de recursos, e isso cria um problema para o classificador. De acordo com Danish (2018), o ROI *pooling* ajuda a redimensionar o tamanho do mapa de recursos para um tamanho fixo, cortando parte da imagem. Através do mapa de características de tamanho fixo é gerada a máscara para eles. Durante o modo de inferência, a máscara prevista é redimensionada para o tamanho original da caixa delimitadora da ROI e fornece a máscara final para cada objeto.

Figura 38 – Ilustração da máscara prevista do trabalho de Danish(2018).

Fonte: Danish (2018)

Para o modelo base de treinamento alguns hiper parâmetros da rede foram configurados pelo autor, conforme demonstrado na Tabela 1

Tabela 1 – Hiperparâmetros utilizados no trabalho de Danish(2018)

Parâmetro	Valor
Architecture	ResNet101
Learning Rate	0.001
Learning rate momentum	0.9
Image min dimension	512
Image max dimension	512
Detection min confidence	0.5
Number of Epoch	2
Steps per epoch	5000
Validation steps	5
Mask shape	28x28
Num classes	2
Batch size	2

Fonte: Danish (2018)

Danish (2018), utilizou as ferramentas de aprendizado profundo Keras e Tensor Flow (ABADI *et al.*, 2016) e seu código foi escrito utilizando a linguagem Python. Já os recursos de *hardware* utilizados são demonstrados abaixo:

- Dois processadores Quad Core Intel Xeon 2.8 GHz;
- 512 MB RAM;
- Duas placas GPU Tesla K40;
- 4 TB de armazenamento.

Para avaliar o desempenho de seu modelo, Danish (2018) utilizou uma métrica específica chamada *Mean Average Precision* (mAP), que é a média da precisão média de todas as classes. Em seu trabalho apenas a classe vaca estava disponível. Já a métrica *Intersection over Union* (IoU) foi utilizada para mensurar a precisão da máscara prevista em relação a máscara anotada. O resultado de seu experimento foi comparado ao resultado encontrado utilizando os pesos pré-treinados existentes do conjunto de dados MS COCO *dataset* (LIN *et al.*, 2014) .

Além do modelo base, outros quatro modelos foram construídos alterando vários hiperparâmetros, como número de épocas, treinamento, conjunto de validação, dimensão da imagem, a fim de obter melhores resultados. O valor padrão de IoU para relatórios utilizado foi AP@0,50, AP@0,70 e AP@0,5: 0,95. A performance dos modelos propostos pode ser conferida no quadro abaixo.

Tabela 2 – Performance dos modelos propostos do trabalho de Danish(2018)

Modelo	AP@0.50	AP@0.7	AP@0.5:0.95
Modelo Base	0.502	0.442	0.325
Modelo 1	0.524	0.460	0.311
Modelo 2	0.492	0.425	0.309
Modelo 3	0.598	0.511	0.331
Modelo 4	0.632	0.522	0.347

Fonte: Danish (2018)

O modelo 4 apresentou os melhores resultados em relação aos demais e, apesar de apenas 27 imagens serem utilizadas, obteve uma melhora significativa em relação aos pesos padrões utilizados pelo MS COCO dataset.

4.2 *CATTLE SEGMENTATION AND CONTOUR EXTRACTION BASED ON MASK R-CNN FOR PRECISION LIVESTOCK FARMING (QIAO, TRUMAN E SUKKARIEH, 2019)*

De acordo com Qiao, Truman e Sukkarieh (2019) abordagens baseadas em visão computacional têm sido amplamente utilizadas para obter informações sobre saúde e bem-estar do gado, pontuação de condição corporal, peso vivo e comportamentos. Segmentar precisamente cada imagem de gado de seu fundo é um pré-requisito importante para obtenção de informações individuais do gado em tempo real.

A fim de alcançar o objetivo de segmentar com alta precisão e desempenho o contorno em ambientes de fundo complexos, uma segmentação automática de gado e método de extração de contorno baseado no *framework* Mask R-CNN (HE *et al.*, 2017) foi proposta pelos autores. A segmentação de bovinos baseada em Mask R-CNN é realizada combinando detecção, onde é localizado e classificado o animal em uma imagem e a segmentação semântica, que consiste na identificação dos pixels do gado juntos. Além disso, para comparar os seus resultados, outros dois *frameworks* foram utilizados, DeepMask (PINHEIRO; COLLOBERT; DOLLAR, 2015) e SharpMask (PINHEIRO *et al.*, 2016).

As imagens utilizadas foram obtidas através de uma câmera 3D ZED com campo de visão de 110° (Figura 39), que foi instalada em um local onde os animais transitavam um por vez. A resolução da imagem foi definida em 1920x1080 com taxa de quadros de 30fps.

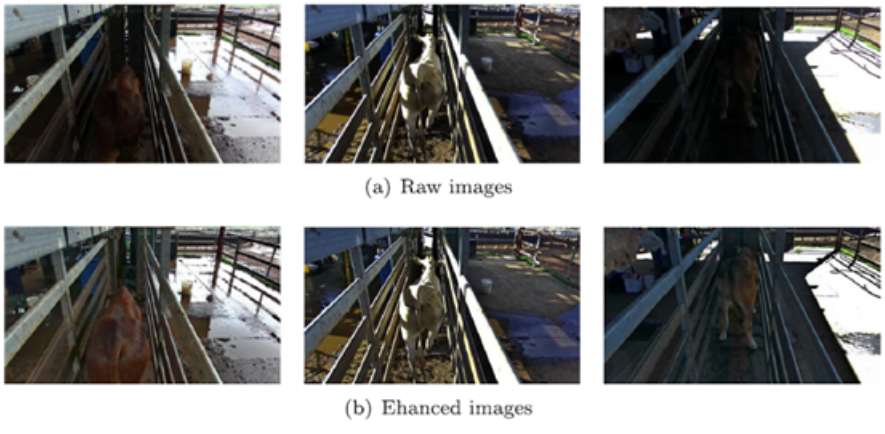
Figura 39 – Hardware utilizado por Qiao, Truman e Sukkarieh (2019)

Fonte: Qiao, Truman e Sukkarieh (2019)

Segundo Qiao, Truman e Sukkarieh (2019), as principais dificuldades no processo de aquisição das imagens são: Os animais mudam de postura com freqüência, alta similaridade entre animais da mesma espécie, iluminação mudando conforme a hora do dia, o que faz com que algoritmos de aprendizado profundo erroneamente interpretam sombras como características dos animais, além disso, um plano de fundo complexo que muitas vezes pode acabar se misturando ao contorno dos animais.

Para resolver os problemas com iluminação e evitar grandes diferenças de brilho entre os quadros de imagem, Qiao, Truman e Sukkarieh (2019) utilizaram uma função gama bidimensional baseada no algoritmo de correção adaptativa de imagem (LIU *et al.*, 2016). O algoritmo é utilizados para remover a iluminação não uniforme e influência de sombras, a fim de melhorar significantemente a qualidade da imagem. O resultado deste processo é demonstrado na Figura 40.

Figura 40 – Ilustração do processo de melhoria da iluminação de imagem utilizado por Qiao, Truman e Sukkarieh (2019)



Fonte: Qiao, Truman e Sukkarieh (2019)

As imagens utilizadas pelos autores foram redimensionadas para 960×540 e dois *datasets* foram formados, um com as imagens brutas capturadas pela câmera e outro com as mesmas imagens após passarem pelos processos de melhoria de iluminação. Um total de 1188 imagens foram utilizadas, sendo 988 para treinamento do modelo e 200 para o conjunto de testes. Todas as imagens foram anotadas utilizando a ferramenta LabelMe (RUSSELL *et al.*, 2008).

Para a construção do modelo a ferramenta de aprendizado profundo TensorFlow (ABADI *et al.*, 2016) com GPU foi utilizada. Já os recursos de *hardware* são demonstrados abaixo.

- Processador Intel Xeon E5-2630 @ 2.20 GHz×20;
- 32GB Memória RAM;
- Placa GPU GeForce GTX 1080 Ti;
- 1 TB de armazenamento.

Durante o desenvolvimento do modelo proposto por Qiao, Truman e Sukkarieh (2019), pesos pré-treinados com base no conjunto de dados COCO (LIN *et al.*, 2014) foram adotados para acelerar o processo de treinamento. A uma taxa de aprendizado de 0.0003, dois dias foram necessários para treinar o modelo no conjunto de imagens brutas e aprimoradas. Além disso, os dois *datasets* foram treinados utilizando outros dois *frameworks*, DeepMask e SharpMask. O tempo de treinamento em cada um deles ficou em 10h com uma taxa de aprendizado de 0.001.

Para avaliar desempenho da segmentação e extração do contorno do gado os autores utilizaram as métricas *Mean Pixel Accuracy* (MPA) and *Average Distance Error* (ADE). As fórmulas para cálculo das duas métricas são demonstradas abaixo.

Figura 41 – Métricas ADE E MPA

$$\text{MPA} = \frac{1}{k}\sum_{i=0}^{k}\frac{p_{ii}}{\sum_{j=0}^{k} p_{ij}} \qquad \text{ADE} = \frac{A_{union} - A_{overlap}}{T_{contour}}$$

Fonte: Qiao, Truman e Sukkarieh (2019)

Segundo os autores a segmentação de gado baseada no Mask R-CNN alcançou um MPA de 0,92 com tempo de processamento de 0,73s para cada imagem, o que é melhor do que a Deepmask com MPA de 0,53 e SharpMask com MPA de 0,82. Os resultados da segmentação nos conjuntos de dados de imagens bruta e aprimorada pode ser visto na Tabela 3, enquanto na Figura 43 são demonstradas as máscaras previstas.

Tabela 3 – Resultados da segmentação nos conjuntos de dados de imagens bruta e aprimorada no trabalho de Qiao, Truman e Sukkarieh (2019)

Método	Dataset	MPA	Tempo(s)
DeepMask	Imagens brutas	0.37	1.27
	Imagens realçadas	0.53	1.27
SharpMask	Imagens brutas	0.79	1.34
	Imagens realçadas	0.82	1.34
Mask R-CNN	Imagens brutas	0.90	0.73
	Imagens realçadas	0.92	0.73

Fonte: Qiao, Truman e Sukkarieh (2019)

Comparando a máscara rotulada manualmente usando a ferramenta LabelMe e o contorno previsto, o ADE calculado é demonstrado na Tabela 4. A abordagem baseada em Mask R-CNN atingiu um ADE de 33,56 da linha de contorno extraída, o que é significativamente melhor do que do DeepMask (53.05) e SharpMask (42,64).

Tabela 4 – Resultados do cálculo do ADE no trabalho de Qiao, Truman e Sukkarieh (2019)

Método	Min	Max	ADE	Tempo(s)
DeepMask	0.052	112.54	53.05	1.29
SharpMask	0.074	68.71	42.64	1.35
Mask R-CNN	0.035	64.17	33.56	0.77

Fonte: Qiao, Truman e Sukkarieh (2019)

De acordo com Qiao, Truman e Sukkarieh (2019), além de segmentar o corpo de animais em imagens que contenham apenas um animal, o *framework* MASK R-CNN é capaz de segmentar o contorno do gado individualmente em imagens que contenham dois ou mais animais (Figura 42).

Figura 42 – Máscaras previstas em imagens com dois animais no trabalho de Qiao, Truman e Sukkarieh (2019)

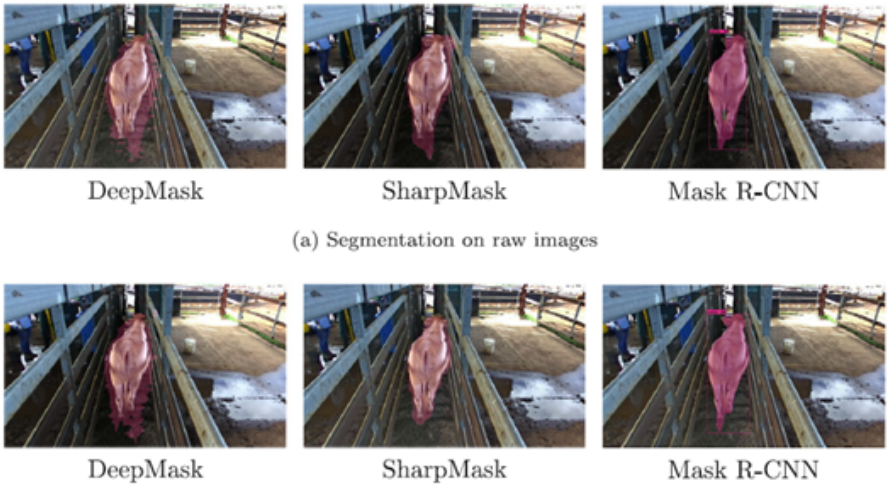
(a) Enhaced images

(b) Mask R-CNN segmentation

Fonte: Qiao, Truman e Sukkarieh (2019)

Para Qiao, Truman e Sukkarieh (2019), nas imagens brutas a área corporal segmentada do gado, principalmente nas regiões próximas à cabeça e perna, apresentaram as maiores diferenças em relação ao contorno real do animal, enquanto a situação de ultrapassagem nas regiões da perna e da cabeça é significativamente reduzida nas imagens aprimoradas. Deixaram, com isso, comprovada a eficiência dos métodos de aprimoramento utilizados para reduzir a influência das sombras e variação de iluminação.

Figura 43 – Máscaras previstas no trabalho de Qiao, Truman e Sukkarieh (2019)



Fonte: Qiao, Truman e Sukkarieh (2019)

4.3 *FLYOLOV3 DEEP LEARNING FOR KEY PARTS OF DAIRY COW BODY DETECTION (JIANG, ET AL., 2019)*

Para Jiang *et al.* (2019), as tecnologias de análise de vídeos são um meio importante para melhorar a criação inteligente de bovinos. Realizar monitoramento automático e a percepção do comportamento animal recebeu uma crescente atenção no campo da pecuária leiteira inteligente. Para identificar o comportamento dos animais, por exemplo, a detecção precisa das partes principais das vacas se torna fundamental.

A fim de resolver o problema da detecção de alta precisão de partes chave do corpo de vacas, um método que usa a extração de alta precisão de pernas, costas e cabeças de vacas em diferentes ambientes chamado de FLYOLOv3 é proposto pelos autores. FLYOLOv3 é baseado no algoritmo de aprendizagem profunda YOLOv3 (REDMON; FARHADI, 2018), porém, a fim de reduzir interferência no treinamento causado pelo ruído de partícula no mapa de características após convolução, um algoritmo de filtragem média junto com a função de ativação Leaky ReLU foi usado para integrar e personalizar a camada *FilterLayer*.

As imagens usadas no estudo foram capturadas através de vídeos gravados em uma *webcam* SONY HDR-CX290E com uma taxa de quadros de 25 fps. 800 imagens diurnas

de vacas sem comportamento anormal que continham partes do corpo independentes e completas foram selecionadas. Outras 200 imagens foram selecionadas aleatoriamente e ruídos gaussiano com uma relação sinal-ruído de 5db foram adicionados a fim de alcançar uma amplificação dos dados. Um total de 1000 imagens foram utilizadas, sendo 500 para treinamento e outras 500 para teste.

As configurações de *hardware* utilizadas no desenvolvimento do modelo são descritas abaixo.

- Processador Intel® Xeon (R) CPU E3-1620 v4 @ 3,50 GHz;
- 32GB Memória RAM;
- 2 TB de armazenamento;
- Placa GPU NVIDIA Quadro P2000.

De acordo com Jiang *et al.* (2019), o algoritmo YOLOv3 divide a imagem de entrada em grades de 7×7. Então a rede prevê quatro valores de coordenadas para cada caixa delimitadora (bx, by, bw, bh). Usando a classificação multi-rótulo, cada caixa prevê quais classes que a caixa delimitadora pode conter. A previsão de classe é realizada usando a função de perda de entropia cruzada durante o treinamento. O modelo utiliza ainda FPN (LIN *et al.*, 2016) e *clustering K-meas* para extração de recursos.

Os parâmetros do modelo baseado em YOLOv3 foram inicializados conforme os parâmetros treinados no *framework* de redes neurais *Opennet* utilizando o modelo *Darknet53*. Já a taxa de aprendizagem inicial foi 0,001 e o coeficiente de atenuação do peso foi de 0,0005. Já o modelo Faster R-CNN (REN *et al.*, 2015) foi baseado no *framework* de redes neurais Caffe (*convolutional architecture for fast feature embedding*).

Para avaliar o modelo proposto os autores utilizaram quatro medidas. Para representar a precisão média das precisões de todas as categorias foi utilizada a métrica *mean average precision* (mAP), além disso, o número médio de quadros por segundo (aNF) foi medido, a *intersection over union* (IOU) e a taxa de *recall* também foram utilizadas. A Tabela 5 demonstra o resultado da detecção dos três algoritmos.

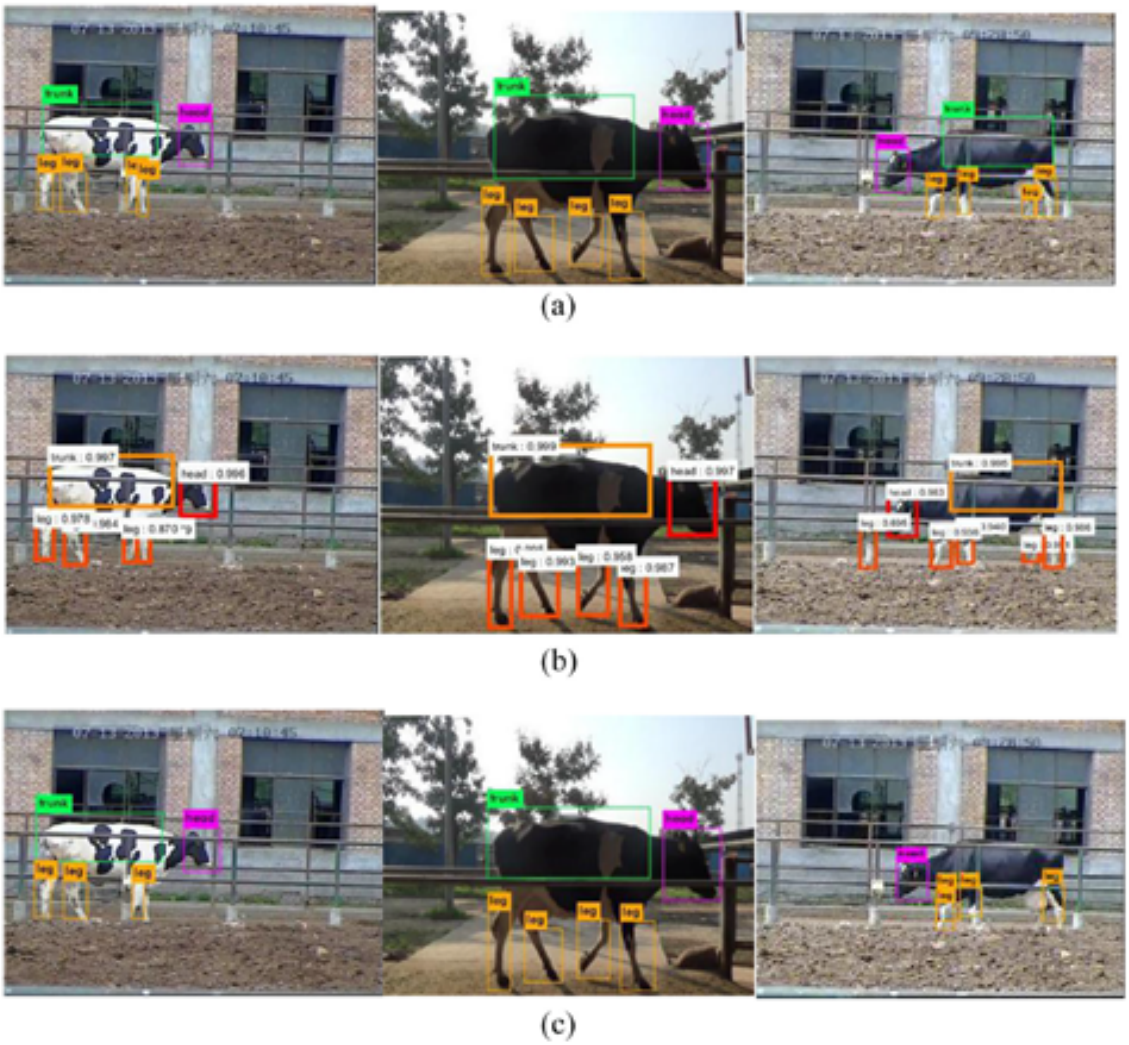
Tabela 5 – Resultado da detecção no trabalho de Jiang, *et al.*, 2019.

Algoritmo	**Precisão (%)**	**Recall (%)**	**aNF (fps)**	**mAP (%)**
FLYOLOv3	99.18	97.51	21	93.73
Faster R-CNN	97.48	95.32	8	93.47
YOLOv3	83.33	65.55	76	76.33

Fonte: Jiang *et al.* (2019)

De acordo com Jiang *et al.* (2019), a cabeça das vacas foram detectadas pelos três algoritmos, e a precisão da detecção foi relativamente alta. Nas de pernas dos animais, o algoritmo FLYOLOv3 teve a melhor detecção, já que o resultado da detecção do algoritmo Faster-RCNN tinha falsas detecções, e o corrimão no fundo foi erroneamente considerado uma perna de vaca em alguns casos.

Figura 44 – Detecções previstas no trabalho de Jiang *et al.*, 2019. (a) FLYOLOv3, (b) Faster R-CNN (c) YOLOv3.



Fonte: Jiang *et al.* (2019)

Os resultados do algoritmo YOLOv3 mostrou detecções perdidas e falsas, indicando que o YOLOv3 tem baixa precisão para pequenos objetos. Para as partes do tronco os resultados da detecção foram semelhantes às partes da cabeça, indicando que os três métodos podem obter melhores resultados para objetos grandes.

Tabela 6 – Resultado da detecção por partes do corpo no trabalho de Jiang, *et al.*, 2019.

Algoritmo	Parte do corpo	Precisão (%)
FLYOLOv3	Cabeça	90.91
	Pernas	90.83
	Tronco	99.45
Faster R-CNN	Cabeça	90.91
	Pernas	90.11
	Tronco	99.38
YOLOv3	Cabeça	80.80
	Pernas	62.26
	Tronco	85.93

Fonte: Jiang *et al.* (2019)

Neste capítulo foram apresentados três trabalhos relacionados a detecção e segmentação do corpo de bovinos em imagens digitais. O trabalho realizado por Danish (2018) utilizou o *framework* Mask R-CNN (HE *et al.*, 2017) para segmentar o corpo inteiro dos animais em um ambiente fechado. Foram utilizadas 27 imagens, das quais 70% durante o treinamento e 30% para teste. Propôs quatro modelos com diferentes parâmetros de treinamento, onde o modelo número 4 apresentou o melhor resultado. O trabalho feito por Qiao, Truman e Sukkarieh (2019), também utilizou o mesmo *framework*, mas comparou o seu resultado com outros dois *frameworks*: DeepMask (PINHEIRO; COLLOBERT; DOLLAR, 2015) e SharpMask (PINHEIRO *et al.*, 2016). Utilizaram 988 imagens para o treinamento e outras 200 para o teste. O MPA do modelo proposto foi de 0,92, o que é melhor que o Deepmask com MPA de 0,53 e SharpMask com MPA de 0,82. Jiang *et al.* (2019) propuseram um novo modelo chamado FLYOLOv3 que é baseado no algoritmo de aprendizagem profunda YOLOv3 (REDMON; FARHADI, 2018). Um total de 1000 imagens foram utilizadas, sendo 500 para treinamento e outras 500 para teste. No processo de detecção das partes do corpo dos animais a precisão em reconhecer a cabeça foi de 90,91%, as pernas 90,83% e o tronco 99,45%.

5 DESENVOLVIMENTO DO PROTÓTIPO

O protótipo consiste em um software desenvolvido na linguagem Python, capaz de segmentar a cabeça, tronco e pernas de bovinos através dos *frameworks* Mask R-CNN (HE *et al.*, 2017) e PointRend (KIRILLOV *et al.*, 2019). Sendo possível também, através deste, avaliar o resultado nos dois *frameworks* citados. Para isso, este capítulo traz detalhes da base de dados utilizada, assim como as tecnologias e o processo de treinamento dos modelos.

5.1 BASE DE DADOS UTILIZADA

Para realizar o treinamento, teste e avaliação do modelo, uma base de dados contendo diversas imagens dos animais é necessária. Infelizmente, não foi possível encontrar nenhuma base que pudesse conter imagens de forma lateral dos animais. Sendo necessário assim, a construção de um *dataset* de imagens adequadas com tamanhos e resoluções variadas.

O *dataset* criado é constituído por 250 fotografias. Destas, 170 foram cedidas por um criador de gado da cidade de Taquara/RS. 50 foram adquiridas através de pesquisas no Google Imagens, as quais foram utilizadas apenas durante o treinamento e não foram expostas neste trabalho. O restante das imagens foram fotografadas pelo autor na cidade de Ararica/RS.

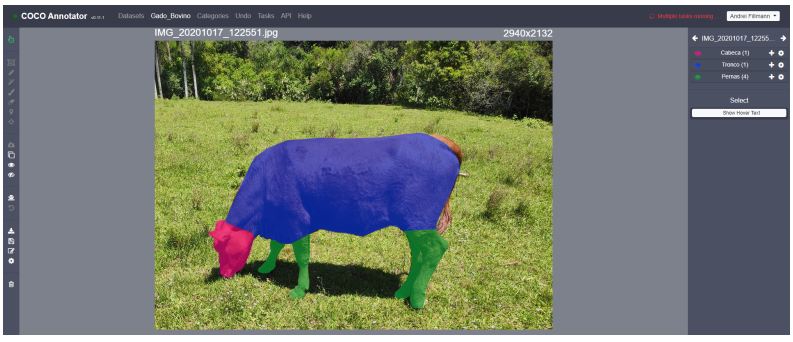
O número de imagens ainda era pequeno para o modelo, desta forma, duas técnicas de *data augmentation* foram utilizadas. Todas as imagens passaram por um processo de *flip*, invertendo tanto as imagens quanto as máscaras horizontalmente, resultando em 500 imagens. Estas ainda foram rotacionadas em 5°, totalizando 1000 imagens. A proporção 60/20/20 foi utilizada para dividir o *dataset*, sendo assim, 600 foram utilizadas durante o treino, 200 usadas para avaliar o treinamento e 200 para testar o modelo. Na Figura 45 é possível ver exemplos das imagens utilizadas.

Figura 45 – Exemplo de imagens utilizadas

Fonte: Do Autor (2021)

As imagens foram anotadas utilizando uma ferramenta de anotação gráfica chamada *COCO Annotator* (BROOKS, 2019). Esta ferramenta permite a criação de polígonos que identificam as partes do corpo dos bovinos, assim como, a exportação do resultado no formato *COCO dataset*, aceito pelos *frameworks* Mask R-CNN e PointRend. A Figura 46 demonstra a ferramenta e um exemplo de imagem anotada.

Figura 46 – Ferramenta COCO Annotator



Fonte: Do autor (2021)

5.2 TECNOLOGIAS

Esta seção apresenta algumas das tecnologias utilizadas durante o processo de construção do protótipo. Dentre elas, será enfatizada a plataforma Detectron2, utilizada para facilitar a implementação dos *frameworks* Mask R-CNN (HE *et al.*, 2017) e Poin-

tRend (KIRILLOV *et al.*, 2019), a linguagem de programação Python, PyTorch, OpenCV e o Google Colab, que permite o treinamento de modelos em GPU na nuvem.

5.2.1 Detectron2

O Detectron2 (WU *et al.*, 2019) é uma plataforma de código aberto lançada pelo *Facebook AI Research* para desenvolver projetos de visão computacional, é flexível e extensível, permitindo, desta forma, a personalização em qualquer parte dos modelos disponíveis. Além disso, através do *backbone* desenvolvido em PyTorch, é capaz de fornecer treinamento rápido em GPU. Ainda segundo o autor citado, Detectron2 inclui implementações de alta qualidade de algoritmos de detecção e segmentação de objetos de última geração, como os citados abaixo:

- Faster R-CNN;
- Mask R-CNN;
- RetinaNet;
- DensePose;
- Cascade R-CNN;
- Panoptic FPN;
- TensorMask;
- PointRend.

5.2.2 Python

Python é uma linguagem de programação de computadores de propósito geral criada em 1991 por Guido van (BANIN, 2018). Possui uma grande comunidade mundial de colaboradores e usuários. É uma linguagem interpretada e fracamente tipada, que tem como enfoque a simplicidade na escrita do código-fonte. De acordo com Banin (2018), a linguagem é por um lado simples e intuitiva, atendendo bem ao propósito de ser uma linguagem inicial utilizada por estudantes de programação. E ao mesmo tempo poderosa e robusta, pois conta com uma grande e variada gama de bibliotecas aplicáveis a várias áreas da tecnologia da informação.

5.2.3 PyTorch

PyTorch é uma biblioteca de código aberto do Facebook que facilita a escrita de código de aprendizado profundo em Python. Permite assim, que o desenvolvedor se conecte ao vasto ecossistema de bibliotecas e *softwares* disponíveis na linguagem. Possui módulos

que ajudam a manipular texto (torchtext), imagens (torchvision) e áudio (torchaudio). Além de ser amplamente utilizada no meio científico, diversas empresas constroem suas aplicações de aprendizado profundo em cima do PyTorch, entre elas é possível citar o Twitter, Salesforce, Uber, Tesla e o próprio Facebook (CATANZARO, 2018).

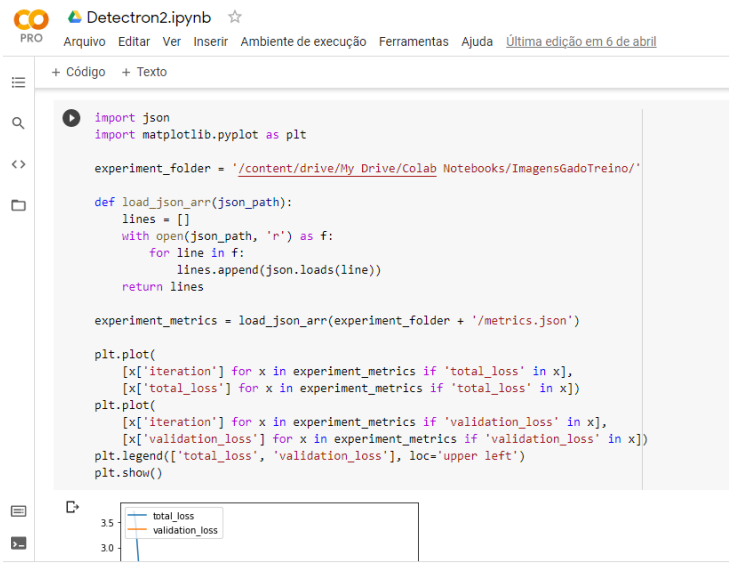
5.2.4 OpenCV

OpenCV é uma biblioteca de visão computacional de código aberto que é escrita em C e C ++ e que pode ser rodada em Linux, Windows e Mac OS X. Há um desenvolvimento ativo em interfaces para Python, Ruby, Matlab, entre outras linguagens. Foi projetado para eficiência computacional e com um forte foco em tempo real. De acordo com Bradski e Kaehler (2008), o principal objetivo do OpenCV é fornecer uma infraestrutura de visão computacional simples de usar, que ajuda as pessoas a criar rapidamente aplicativos de visão bastante sofisticados.

5.2.5 Google Colab

O *Google Colaboratory* (Colab) é um ambiente *Jupyter Notebook* personalizado, sem necessidade de instalação e gratuito, que permite a execução de códigos Python em GPU na nuvem por até 12h de funcionamento contínuo. O ambiente inclui versões pré-instaladas do TensorFlow e PyTorch, para utilização junto com a plataforma de computação paralela CUDA nas placas GPU da Nvidia (GOOGLE, 2018). Na Figura 47 é demonstrado um código Python escrito no Colab.

Figura 47 – Google Colab



Fonte: Do autor (2021)

5.3 TREINAMENTO DO MODELO

Uma das etapas mais importantes no desenvolvimento do protótipo é o treinamento do modelo. Como citado anteriormente, neste trabalho dois *frameworks* foram utilizados, portanto, foi necessário realizar o treinamento tanto para o MASK R-CNN quanto para o PointRend. Além disso, foi avaliado o desempenho do treinamento para o *dataset* original e também para o conjunto de imagens após o processo de *data augmentation*. Na Tabela 7 são apresentadas as distribuições das partes do corpo dos bovinos contidos nos conjuntos de imagens utilizadas para o treinamento e também para a avaliação do treinamento.

Tabela 7 – Distribuição das partes do corpo dos animais nas imagens

Dataset	**Cabeça**		**Tronco**		**Pernas**	
-	*Train*	*Val.*	*Train*	*Val.*	*Train*	*Val.*
Original	157	54	152	52	553	183
Data Augmentation	647	210	619	204	2215	735

Fonte: Do Autor (2021)

O conjunto de dados de treino é utilizado para ajustar os pesos durante as diversas iterações do treinamento. Já o conjunto de validação, de acordo com Brownlee (2017b), é uma amostra de dados usada para fornecer uma avaliação imparcial de ajuste do modelo. A partir do resultado obtido em cima da base de dados de validação, é possível verificar e modificar os hiperparâmetros do modelo, a fim de se evitar, por exemplo, o *overfitting* durante o treinamento.

Diversos hiperparâmetros estão disponíveis na hora de treinar os modelos, entre eles é possível citar *learning rate*, que é responsável pelo ritmo em que os pesos são atualizados, o *batch size*, que indica quantas imagens a rede receberá como entrada de uma única vez, o *max iter*, que define o número de vezes em que a rede processa as imagens de entrada a fim de aprender os pesos das camadas, e o *eval period*, que representa à quantas iterações o modelo será avaliado na base de dados de avaliação. Os hiperparâmetros utilizados no desenvolvimento deste trabalho foram os padrões disponibilizados pela plataforma Detectron2 e são demonstrados na Tabela 8.

Tabela 8 – Hiperparâmetros utilizados nos modelos

Hiperparâmetro	**Valor**
learning rate	0.001
batch size	64
max iter	10000-25000
eval period	500

Fonte: Do Autor (2021)

Os dois modelos propostos utilizam o conceito de aprendizagem por transferência que permite um progresso rápido e melhor desempenho ao modelar redes profundas. De acordo com Brownlee (2017a), a aprendizagem por transferência é uma abordagem popular em aprendizagem profunda, onde modelos pré-treinados são usados como ponto de partida em tarefas de visão computacional. Esses modelos pré-treinados são unidos aos modelos que se propõem a resolver problemas específicos, onde a parte pré-treinada age como extrator de características da imagem, detectando padrões de baixo nível como curvas, bordas e pontos. Para a realização deste trabalho foram utilizados os modelos pré-treinados "pointrend_rcnn_R_101_FPN_3x_coco" e "mask_rcnn_R_101_FPN_3x" disponibilizados pela plataforma Detectron2. A utilização destes dois modelos pré-treinados se deve especificamente ao fato de, segundos os autores citados, estes terem apresentado os melhores resultados para segmentar os obejtos presentes no COCO *dataset*.

Antes de iniciar o treinamento foram realizadas algumas configurações e instalações de dependências necessárias (Figura 48). Primeiro foi dada permissão para que o Colab tenha acesso ao repositório do Google Drive, com esta ação é possível acessar as imagens e os arquivos JSON das imagens. Além disso, permite salvar *logs* e o resultado final do modelo na nuvem. Logo após, foram instalados os pacotes pytorch 1.6.0+cu101 e torchvision 0.7.0+cu101, compatíveis com o CUDA e o Detectron2. Por último, foi instalada a plataforma Detectron2 e clonado o repositório que armazena os modelos pré-treinados.

Figura 48 – Configuração do Colab

```
[ ] from google.colab import drive
    drive.mount("/drive")

    #instalacao da versao especifica do pytorch e do detectron2
    !pip install torch==1.6.0+cu101 torchvision==0.7.0+cu101 -f https://download.pytorch.org/whl/torch_stable.html
    !pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.6/index.html
    # respositorio com modeos pre-treinados
    !git clone https://github.com/facebookresearch/detectron2.git detectron2_repo
```

Fonte: Do autor (2021)

Para iniciar o processo de treinamento, foram registrados os *datasets* de treino, validação e teste na plataforma, passando como parâmetro o caminho para o arquivo JSON de anotação e o das imagens. A seguir, foram determinados o modelo pré-treinado e os *datasets* de treino e validação. O próximo passo foi configurar os hiperparâmetros utilizados, para então iniciar o treinamento. Por último, o modelo foi avaliado utilizando as imagens de teste. Na Figura 49 é possível ver o código Python utilizado para o treinamento do *framework* PointRend.

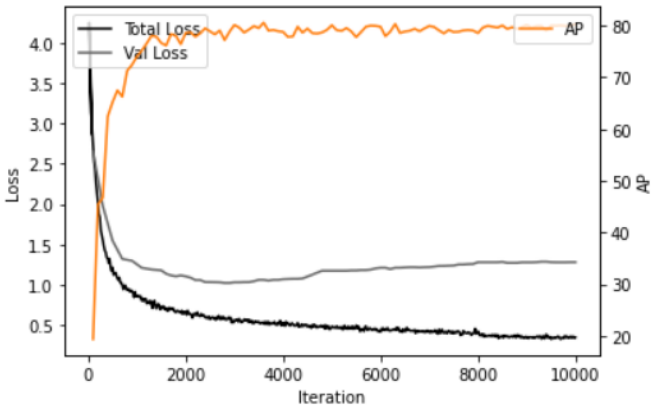
Figura 49 – Código Python responsável pelo treinamento

```
from detectron2.data.datasets import register_coco_instances
from detectron2.engine import DefaultTrainer
from detectron2.config import get_cfg
import os
from detectron2.projects import point_rend
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader

# registra os datasets utilizados
register_coco_instances("Gado_train", {}, "/content/drive/My Drive/Colab Notebooks/ImagensGado/train.json", "/content/drive/My Drive/Colab Notebooks/ImagensGado/")
register_coco_instances("Gado_test", {}, "/content/drive/My Drive/Colab Notebooks/ImagensGado/test.json", "/content/drive/My Drive/Colab Notebooks/ImagensGado/")
register_coco_instances("Gado_val", {}, "/content/drive/My Drive/Colab Notebooks/ImagensGado/val.json", "/content/drive/My Drive/Colab Notebooks/ImagensGado/")

cfg = get_cfg()
cfg.OUTPUT_DIR = '/content/drive/My Drive/Colab Notebooks/Resultados/POINT/'

# carrega as configuracoes do modelo pre-treinado
cfg.merge_from_file("detectron2_repo/projects/PointRend/configs/InstanceSegmentation/pointrend_rcnn_R_101_FPN_3x_coco.yaml")
cfg.DATASETS.TRAIN = ("Gado_train",)
cfg.DATASETS.TEST = ("Gado_val", )

# carrega o modelo pre-treinado
cfg.MODEL.WEIGHTS = "detectron2://PointRend/InstanceSegmentation/pointrend_rcnn_R_101_FPN_3x_coco/28119983/model_final_3f4d2a.pkl"

# hiperparametros
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001
cfg.SOLVER.MAX_ITER = 25000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 3
cfg.MODEL.POINT_HEAD.NUM_CLASSES = 3
cfg.TEST.EVAL_PERIOD = 500

# inicia o treinamento
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = MyTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()

# avalias os resultados
evaluator = COCOEvaluator("Gado_test", cfg, False)
val_loader = build_detection_test_loader(cfg, "Gado_test")
inference_on_dataset(trainer.model, val_loader, evaluator)
```

Fonte: Do autor (2021)

O treinamento foi realizado de duas formas, uma com as imagens originais e outra com a base de dados aumentada. Utilizando a base de dados original, com 150 imagens e 10000 iterações, foi identificado, que perto da iteração de número 5000, a rede parou de convergir, ocasionando um *overfitting* no treinamento. Na Figura 50 é possível notar que por volta desta iteração, a curva que indica a perda na base de validação para de decair e começa a crescer, evidenciando o *overfitting*.

Figura 50 – Treinamento com *overfitting*

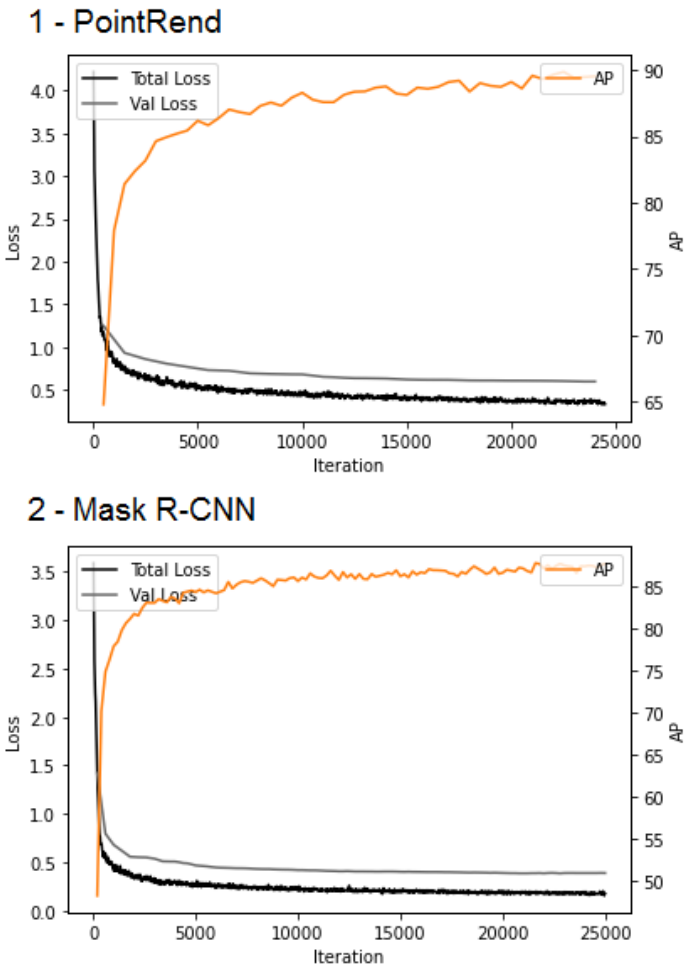


Fonte: Do autor (2021)

Por outro lado, utilizando os dados aumentados, foi possível realizar o treinamento até 25000 iterações sem que houvesse a presença de *overfitting*. Tanto com o Mask R-CNN quanto com o PointRend a curva de perda foi sempre decrescente (Figura 51). É possível notar também que o Mask R-CNN atinge AP de 85 primeiro que o Point, mas após essa marca não existe uma melhora significativa da segmentação. Ao contrário do

PointRend, que apesar de levar mais iterações para atingir a marca de AP 85, possui uma melhora contínua da segmentação até atingir a iteração de número 25000. O tempo total de treinamento foi de 6 horas 59 minutos e 46 segundos para o PoinRend e 7 horas 56 minutos e 13 segundos para o Mask R-CNN. No capítulo a seguir serão discutidos os resultados encontrados durante a aplicação do modelo na base de teste.

Figura 51 – Resultado do treinamento



Fonte: Do autor (2021)

Neste capítulo foi explicado o desenvolvimento do protótipo proposto para esse trabalho. Este incluiu a elaboração de um *dataset* de imagens para ser utilizado durante o treinamento, validação e teste dos modelos. Esse conjunto é composto por 250 imagens, que posteriormente foi aumentado utilizando as técnicas de inversão e rotação, dando origem a um total de 1000 imagens. As imagens foram anotadas utilizando a ferramenta COCO *Annotator*. Além dessa ferramenta, outras tecnologias foram usadas neste processo, dentre elas, a plataforma Detectron2, a linguagem de programação Python, o PyTorch, OpenCV e o Google Colab. O conjunto de dados de treino é utilizado para ajustar os pesos durante o treinamento. Já o conjunto de validação, é uma amostra de dados usada

para avaliar o desempenho do treinamento e realizar ajustes nos parâmetros do modelo. Diversos hiperparâmetros estão disponíveis na hora de treinar os modelos, para este trabalho o *learning rate* usado foi 0.001, o *batch size* foi igual a 64, o *max iter* foi de 10000 e 25000, e o *eval period* foi de 500. Além disso, os dois modelos treinados permitem utilizar o conceito de aprendizagem por transferência através de modelos pré-treinados. Antes de iniciar o treinamento de fato, foram realizadas algumas configurações e instalações de dependências necessárias, entre elas, a permissão para o Colab acessar os dados do Google Drive, e instalação dos pacotes pytorch 1.6.0+cu101 e torchvision 0.7.0+cu101. Utilizando o *dataset* original, o treinamento evidencia o *overfitting* a partir da iteração 5000. Por outro lado, utilizando os dados aumentados, foi possível realizar o treinamento até 25000 iterações sem que houvesse o *overfitting*. O tempo total de treinamento foi de 6 horas 59 minutos e 46 segundos para o PoinRend e 7 horas 56 minutos e 13 segundos para o Mask R-CNN. No capítulo a seguir serão discutidos os resultados encontrados durante a aplicação do modelo na base de teste.

6 RESULTADOS E DISCUSSÃO

Neste capítulo serão apresentados os resultados encontrados no *dataset* de teste utilizando os modelos propostos. Para mensurar a qualidade da segmentação foram utilizadas as métricas de precisão média (AP) e a média de precisão média (mAP), que de acordo com Castro e Pereira (2020), são indicadores bem consolidados para os problemas de detecção e segmentação de objetos. O mAP nada mais é do que a média do AP de cada uma das classes reconhecidas pelo modelo. Sobre o AP, Castro e Pereira (2020) citam que a métrica se trata de uma ponderação entre a capacidade de predizer corretamente os casos positivos corretos entre todos os casos positivos reais (*recall*), a porcentagem de predições positivas corretas (precisão) e a área de intersecção sobre união (IoU) entre a máscara predita do objeto pela máscara real.

A IoU pode variar de 0 a 1, com 0 representando sem sobreposição e 1 significando sobreposição perfeita entre a máscara prevista e a real. Além disso, é preciso definir um limite que é usado para distinguir uma segmentação positiva de uma negativa. Para essa avaliação foram definidos os limites de 0.5 e 0.75, representando 50% e 75% respectivamente, e que dão origem às nomenclaturas AP@50 e AP@75. Também foi possível identificar a precisão dos dois modelos em reconhecer objetos por seus tamanhos utilizando as métricas APs (pequenas), APm (médias) e APl (grandes). Na Tabela 9 são demonstrados os valores encontrados na realização dos testes para cada uma das métricas citadas.

Tabela 9 – Resultados dos modelos

Modelo	**mAP**	**AP@50**	**AP@75**	**APs**	**APm**	**APl**
Mask R-CNN	88.529	98.331	97.739	36.341	84.202	91.696
PointRend	90.576	98.321	97.860	35.367	86.591	94.808

Fonte: Do Autor (2021)

Olhando para os resultados é possível identificar que a precisão média de todas as classes, o PoinRend tem um aproveitamento de 2,047% a mais que o Mask R-CNN. Em relação ao AP@50, ambos possuem resultados bem próximos. Já para a AP@75, o PointRend tem um desempenho superior. Para segmentação de objetos de diferentes tamanhos, o Mask R-CNN teve um melhor resultado para objetos pequenos e o PointRend apresenta vantagem em segmentar objetos médios e grandes. O tempo médio de predição foi de 0,21 segundos por imagem para o Mask R-CNN e 0,26 segundos para o PointRend.

Os resultados ainda foram avaliados de forma individual para cada uma das classes, conforme demonstra a Tabela 10. A cabeça dos animais são objetos de tamanho médio e

tiveram um bom resultado para a segmentação. Já o tronco apresentou a melhor precisão, este resultado se deve principalmente ao tamanho e pela menor variância do objeto, sendo esse muito parecido em todos os animais. As pernas apresentam uma grande variedade devido ao movimento e posição do animal, além disso, muitas imagens apresentavam oclusão parcial das pernas, o que fez aumentar a dificuldade em segmentar corretamente esta parte do corpo.

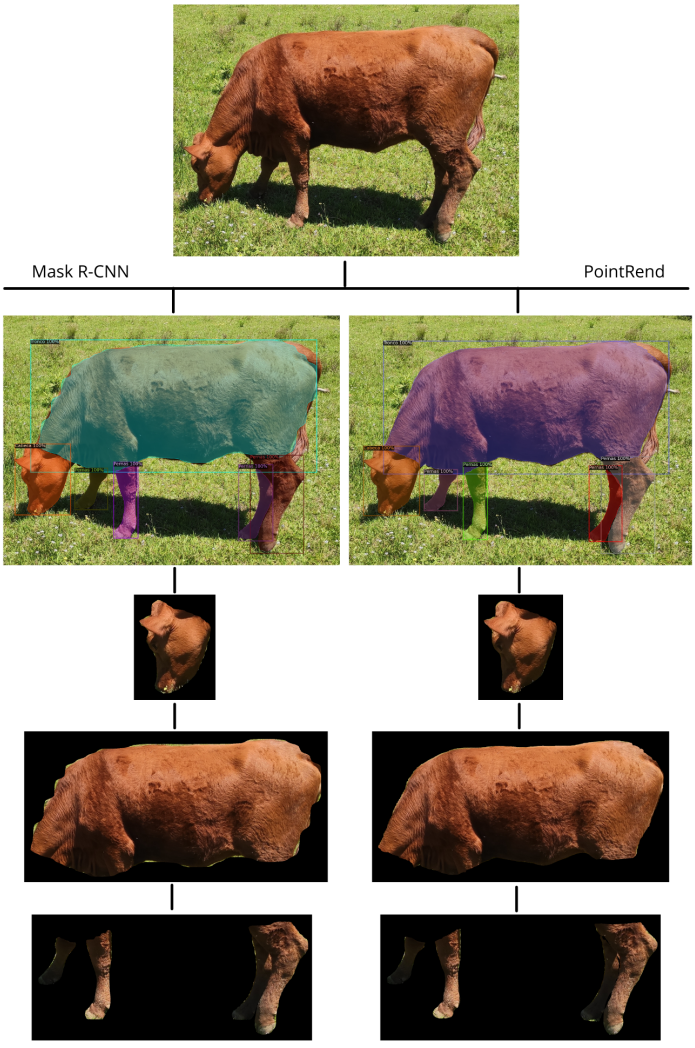
Tabela 10 – Resultados dos modelos por classes

Modelo	Cabeça	Tronco	Pernas
Mask R-CNN	87.824	99.012	78.751
PointRend	91.432	99.338	80.958

Fonte: Do Autor (2021)

Na Figura 52 é demonstrado um exemplo de segmentação feita pelos dois modelos.

Figura 52 – Imagem Segmentada - Mask R-CNN X Point Rend



Fonte: Do autor (2021)

Estes recebem uma imagem como entrada, em seguida, preveem as máscaras e rótulos dos objetos como saída, para então os objetos serem extraídos da imagem original. Analisando a cabeça extraída, é possível verificar que ambos os *frameworks* possuem resultados bem próximos. Se tratando do tronco do animal, a capacidade de segmentar objetos utilizando técnicas de renderização do PointRend fica bem explícita, já que o serrilhado encontrado no Mask R-CNN não é mais visível. A segmentação das pernas não apresenta também grandes discrepâncias entre os modelos.

Desta forma, neste capítulo conclui-se que, as métricas mAP e AP são adequadas para mensurar a qualidade da segmentação de objetos em imagens digitais. Além disso, foi utilizado o valor da IoU para determinar uma predição positiva de uma negativa. Este pode variar de 0 a 1, com 0 representando sem sobreposição e 1 significando sobreposição total entre a máscara prevista e a real. Para este trabalho foram utilizados os valores de 0.50 e 0.75 como limite para uma predição positiva, dando origem às nomenclaturas AP@50 e AP@75. Em relação ao Mask R-CNN, a média de precisão média (mAP) foi de 88.529. Enquanto o AP@50 e AP@75 foi de 98.331 e 97.739 respectivamente. Já o tempo de predição médio deste modelo foi de 0.21 segundos por imagem. O PointRend teve como mAP o valor de 90.576, o AP@50 foi igual a 98.321 e o AP@75 teve como resultado 97.739. O tempo médio de predição para cada imagem foi de 0.26 segundos. Em relação a avaliação individual de cada classe, o PointRend teve um melhor desempenho para prever as máscaras da cabeça, tronco e pernas dos animais. Em ambos os modelos, a cabeça dos animais, assim como o tronco tiveram um bom resultado para a segmentação. O resultado do tronco se deve principalmente ao tamanho e pela menor variação do objeto. Já as pernas apresentam uma grande variedade devido ao movimento e posição do animal, além disso, muitas imagens apresentavam oclusão parcial das pernas, levando a ser parte do corpo com pior desempenho ao ser segmentado.

7 CONCLUSÃO

A pecuária é um segmento muito importante para a economia brasileira, mas ainda carece de eficiência em seus processos. A pecuária de precisão busca, através de altas tecnologias, melhorar os procedimentos do dia a dia de uma fazenda. A Inteligência Artificial é uma das áreas que busca desenvolver tecnologias a serem utilizadas na pecuária de precisão, assim como, segmentar partes do corpo de bovinos utilizando Redes Neurais. Este trabalho, é um exemplo de novos processos que podem contribuir no aumento do desempenho dos criadores de gado.

Durante esta pesquisa, foram realizadas revisões teóricas sobre o Processamento Digital de Imagens (PDI) e as Redes Neurais Artificiais (RNA). O PDI inclui diversos procedimentos que foram importantes para entender como as máquinas processam imagens, além de gerar conhecimento sobre o tratamento destas. Dentre os tratamentos possíveis, incluem-se o melhoramento na iluminação e contraste, assim como a redução do ruído causado na aquisição das imagens, entre outros. Outras etapas também foram abordadas, como a segmentação, o pós-processamento, a extração de atributos e a classificação e reconhecimento.

Através da pesquisa realizada sobre RNA, foi possível compreender a forma de se adquirir conhecimento, bem como, a forma de aprendizagem das Redes Neurais Convolucionais. Este é um tipo de rede de aprendizado profundo muito utilizada em Visão Computacional e que é considerada estado da arte para detecção, segmentação, classificação e reconhecimento de objetos.

Ao longo do treinamento dos modelos ficou evidente a importância da utilização de modelos pré-treinados. Treinar os modelos do zero levaria muito tempo, mesmo utilizando GPU. Apesar de usar os modelos pré-treinados e uma placa CPU, o processo de treinamento seria infinitamente longo. Já adquirir um equipamento com GPU se tornaria inviável. Neste caso, a utilização do Google Colab, que permitiu a utilização de GPU na nuvem de forma gratuita, assim como a plataforma Detectron2, que disponibiliza modelos pré-treinados, se mostraram fundamentais na realização deste trabalho. Ainda sobre o treinamento, uma quantidade considerável de imagens foi necessária, pois utilizando a base de imagens original, por volta da iteração 5000 já foi possível identificar a presença de *overfitting*. Este problema foi contornado através de técnicas de *data augmentation*, multiplicando o número de imagens disponíveis.

Se tratando dos resultados encontrados, o Mask R-CNN teve a média de precisão média (mAP) de 88.529 e o PointRend de 90.576. Individualmente, para cada uma das classes a precisão do Mask R-CNN foi de 87.824 para a cabeça, 99.012 no tronco e 78.751

para as pernas. Enquanto o PointRend, teve como resultado 91.432 para a cabeça, 99.338 no tronco e 80.958 para as pernas.

A Inteligência Artificial, junto com as Redes Neurais Artificiais evoluíram muito nos últimos anos. No campo de estudo da Visão Computacional não foi diferente, o *framework* Mask R-CNN, proposto por He *et al.* (2017), é considerado o estado da arte para tarefas de segmentação. Mas como demonstrado na realização deste trabalho, o PointRend teve um desempenho superior na tarefa de segmentar partes do corpo de gado bovino.

Desta forma, conclui-se que tanto o modelo treinado como o *framework* Mask R-CNN, quando o treinado com o PointRend, são adequados para segmentar partes do corpo de bovinos e as medidas extraídas das partes segmentadas do corpo dos animais podem ser utilizadas como entrada em modelos que se propõem estimar o peso e qualidade corporal de bovinos.

Além de aprimorar os conhecimentos nos ramos da pecuária de precisão e Inteligência Artificial, este trabalho deixa como contribuição científica a formação de uma base de imagens rotuladas de partes do corpo de bovinos. Para trabalhos futuros propõem-se a diferenciação dos animais por raça, já que estes possuem características específicas que variam, além disso, o *dataset* criado pode ser utilizado para avaliar outros métodos de segmentação.

REFERÊNCIAS

ABADI, M. *et al.* Tensorflow a system for large-scale machine learning. *CoRR*, abs/1605.08695, 2016. Disponível em: <http://arxiv.org/abs/1605.08695>. Citado 2 vezes nas páginas 52 e 55.

ALVAREZ, J. *et al.* Body condition estimation on cows from depth images using convolutional neural networks. Buenos Aires, Argentina, 2018. Citado na página 14.

BANIN, S. L. *Python 3: CONCEITOS E APLICAÇÕES: UMA ABORDAGEM DIDATICA*. [S.l.]: ERICA, 2018. ISBN 9788536527819. Citado na página 63.

BODLA, N. *et al.* Improving object detection with one line of code. *CoRR*, abs/1704.04503, 2017. Disponível em: <http://arxiv.org/abs/1704.04503>. Citado na página 40.

BORTH, M. R. *et al.* A visão computacional no agronegócio: Aplicações e direcionamentos. *Segundo Seminário Internacional de Integração e Desenvolvimento Regional*, Ponta Porã, MS, 2014. Citado na página 13.

BRADSKI, G. R.; KAEHLER, A. *Learning OpenCV - computer vision with the OpenCV library: software that sees.* [S.l.]: O'Reilly, 2008. ISBN 97-0596516130. Citado na página 64.

BROOKS, J. *COCO Annotator*. 2019. Disponível em: <https://github.com/jsbroks-/coco-annotator/>. Acesso em: 11 abr. 2021. Citado na página 62.

BROWNLEE, J. *A Gentle Introduction to Transfer Learning for Deep Learning*. 2017. Disponível em: <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>. Acesso em: 28 abr. 2021. Citado na página 66.

BROWNLEE, J. *What is the Difference Between Test and Validation Datasets?* 2017. Disponível em: <https://machinelearningmastery.com/difference-test-validation-datasets/>. Acesso em: 28 abr. 2021. Citado na página 65.

CASTRO, L. B. d.; PEREIRA, F. G. Inspeção de componentes de vagões ferroviários baseado em visão computacional e aprendizagem profunda. Espírito Santo, ES, 2020. Citado na página 70.

CATANZARO, B. *PyTorch Developer Conference 2018: Enterprise, Education, and Future of AI Panel*. 2018. Disponível em: <https://developers.facebook.com/videos-/2018/pytorch-developer-conference-2018-enterprise-education-future-of-ai-panel/>. Acesso em: 11 abr. 2021. Citado na página 64.

CHOLLET, F. *Deep Learning with Python*. 1. ed. [S.l.]: Manning Publications Co., 2017. ISBN 1617294438. Citado na página 36.

COHEN, P. R.; FEIGENBAUM, E. A. *The handbook of artificial intelligence*. Callifornia: Willian Kaufman, 1984. Citado na página 14.

DANISH, M. *Beef Cattle Instance Segmentation Using Mask R-Convolutional Neural Network*. Dissertação (Mestrado) — DIT, Dublin, Irlanda, 2018. Citado 6 vezes nas páginas 49, 50, 51, 52, 53 e 60.

ESQUEF, I. A.; ALBUQUERQUE, M. P. d.; ALBUQUERQUE, M. P. d. *Processamento Digital de Imagens*. 2003. Disponível em: <http://www.dsc.ufcg.edu.br/~hmg-/disciplinas/graduacao/vc-2016.2/Rita-Tutorial-PDI.pdf>. Acesso em: 26 out. 2020. Citado 5 vezes nas páginas 16, 17, 22, 23 e 27.

EUCLIDES, V. Alternativas para intensificação da produção de carne bovina em pastagem. *Embrapa*, Campo Grande, MS, 2000. Citado na página 13.

GIRSHICK, R. B. Fast r-cnn. *CoRR*, abs/1504.08083, 2015. Disponível em: <http://arxiv.org/abs/1504.08083>. Citado 3 vezes nas páginas 38, 39 e 48.

GONZALEZ, R. C.; WOODS, R. E. *Processamento Digital de Imagens*. 3. ed. São Paulo: Pearson Universidade, 2010. ISBN 978-8576054016. Citado 7 vezes nas páginas 13, 16, 17, 20, 21, 23 e 24.

GOOGLE. *Colaboratory*. 2018. Disponível em: <https://research.google.com-/colaboratory/faq.html>. Acesso em: 11 abr. 2021. Citado na página 64.

HAVENS, W. S.; MACKWORTH, A. K. *Computer graphics*. London: Prentice-hall, 1986. Citado na página 14.

HAYKIN, S. *Redes neurais : princípios e prática*. Porto Alegre: Bookman, 2011. Citado 2 vezes nas páginas 29 e 30.

HE, K. *et al.* Mask r-cnn. *CoRR*, abs/1703.06870, 2017. Disponível em: <http://arxiv-.org/abs/1703.06870>. Citado 11 vezes nas páginas 39, 45, 46, 47, 48, 49, 53, 60, 61, 62 e 74.

HE, K. *et al.* Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. Disponível em: <http://arxiv.org/abs/1603.05027>. Citado na página 44.

HE, K. *et al.* Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. Disponível em: <http://arxiv.org/abs/1512.03385>. Citado 2 vezes nas páginas 44 e 51.

IBGE. *Indicadores IBGE: Estatística da Produção Pecuária*. 2020. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/periodicos/2380/epp_2020_3tri.pdf>. Acesso em: 01 jun. 2021. Citado na página 13.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. Disponível em: <http://arxiv.org/abs/1502.03167>. Citado na página 37.

JIANG, B. *et al.* Flyolov3 deep learning for key parts of dairy cow body detection. *Computers and Electronics in Agriculture*, v.166, 104982, 2019. Citado 4 vezes nas páginas 57, 58, 59 e 60.

KHAN, S. *et al. A Guide to Convolutional Neural Networks for Computer Vision*. Australia: Morgan & Claypool Publishers, 2018. ISBN 9781681730226. Citado 13 vezes nas páginas 29, 30, 31, 32, 33, 34, 35, 36, 38, 41, 42, 44 e 45.

KIRILLOV, A. *et al.* Pointrend: Image segmentation as rendering. In: . [s.n.], 2019. Disponível em: <http://arxiv.org/abs/1912.08193>. Citado 4 vezes nas páginas 47, 48, 61 e 63.

LECUN, Y. *et al.* Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*. [S.l.: s.n.], 1998. p. 2278–2324. Citado na página 31.

LIN, T.-y. *et al.* Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. Disponível em: <http://arxiv.org/abs/1612.03144>. Citado 5 vezes nas páginas 41, 42, 48, 51 e 58.

LIN, T.-y. *et al.* Microsoft coco: Common objects in context. *CoRR*, abs/1405.0312, 2014. Disponível em: <http://arxiv.org/abs/1405.0312>. Citado 2 vezes nas páginas 52 e 55.

LIU, Z. *et al.* Adaptive adjustment algorithm for non-uniform illumination images based on 2d gamma function. *Transactions of Beijing Institute of Technology*, v.36, n.2, p.191-196, 2016. Citado na página 54.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. Disponível em: <http://arxiv.org/abs/1411-.4038>. Citado 3 vezes nas páginas 42, 43 e 48.

MARQUES FILHO, O.; VIEIRA NETO, H. *Processamento Digital de Imagens*. Rio de Janeiro: Brasport, 1999. ISBN 8574520098. Citado 3 vezes nas páginas 17, 18 e 20.

NIELSEN, M. *Neural Networks and Deep Learning*. 2015. Disponível em: <http://www-.neuralnetworksanddeeplearning.com/>. Acesso em: 31 out. 2020. Citado 3 vezes nas páginas 31, 34 e 36.

OZKAYA, S. The prediction of live weight from body measurements on femaleholstein calves by digital image analysis. *Journal of Agricultural Science*, Cambridge University, v. 151, p. 570–576, 2013. Citado na página 14.

OZKAYA, S.; BOZKURT, Y. The relationship of parameters of body measures and body weight by using digital image analysis in pre-slaughter cattle. *Arch. Tierz*, Dummerstorf, Alemanha, 2008. Citado na página 14.

OZKAYA, S. *et al.* Estimation of bodyweight from body measurements and determination of body measurements on limousin cattle using digital image analysis. *Animal Production Science*, v. 56, p. 2060–2063, 2016. Citado na página 14.

PEDRINI, H.; SCHWARTZ, W. R. *Análise de imagens digitais: princípios, algoritmos e aplicações*. [S.l.]: Cengage, 2008. ISBN 8522105952. Citado 9 vezes nas páginas 19, 20, 22, 23, 24, 25, 26, 27 e 28.

PINHEIRO, P. H. O.; COLLOBERT, R.; DOLLAR, P. Learning to segment object candidates. *CoRR*, abs/1506.06204, 2015. Disponível em: <http://arxiv.org/abs/1506-.06204>. Citado 2 vezes nas páginas 53 e 60.

PINHEIRO, P. H. O. *et al.* Learning to refine object segments. *CoRR*, abs/1603.08695, 2016. Disponível em: <http://arxiv.org/abs/1603.08695>. Citado 2 vezes nas páginas 53 e 60.

PONTI, M. A.; COSTA, G. B. P. d. Como funciona o deep learning. *Tópicos em Gerenciamento de Dados e Informações, SBC*, 2017. Citado 2 vezes nas páginas 32 e 37.

QIAO, Y.; TRUMAN, M.; SUKKARIEH, S. Cattle segmentation and contour extraction based on mask r-cnn for precision livestock farming. *Computers and Electronics in Agriculture*, v.165, 104958, 2019. Citado 7 vezes nas páginas 13, 53, 54, 55, 56, 57 e 60.

QUEIROZ, J. E. R. d.; GOMES, H. M. *Introdução ao Processamento Digital de Imagens*. 2016. Disponível em: <http://www.dsc.ufcg.edu.br/~hmg/disciplinas/graduacao-/vc-2016.2/Rita-Tutorial-PDI.pdf>. Acesso em: 26 out. 2020. Citado na página 16.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. Disponível em: <http://arxiv.org/abs/1804.02767>. Citado 2 vezes nas páginas 57 e 60.

REIS, G. *et al.* Predição do peso vivo a partir de medidas corporais em animais mestiços holandês/gir. *Ciência Rural*, v.38, n.3, p. 778–783, 2008. Citado na página 13.

REN, S. *et al.* Faster r-cnn: Towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. Disponível em: <http://arxiv.org/abs/1506-.01497>. Citado 6 vezes nas páginas 39, 40, 45, 46, 51 e 58.

ROSA, B. *et al.* Correlações entre medidas corporais e características das carcaças de tourinhos nelore terminados em confinamento. *Boletim de Indústria Animal*, v. 71, n. 4, p. 371–380, 2014. Citado na página 13.

ROSEBROCK, A. *Deep Learning for Computer Vision with Python 1. Starter Bundle*. 1. ed. [S.l.]: Py Image Search, 2017. Citado 3 vezes nas páginas 30, 33 e 35.

ROSEBROCK, A. *Deep Learning for Computer Vision with Python 3. ImageNet Bundle*. 1. ed. [S.l.]: Py Image Search, 2017. Citado 3 vezes nas páginas 44, 45 e 46.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958. Citado na página 30.

RUSSELL, B. *et al.* Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, v. 77, p. 157–173, 2008. Citado na página 55.

USDA. *Livestock and Poultry: World Markets and Trade*. 2021. Disponível em: <https://apps.fas.usda.gov/psdonline/circulars/livestock_poultry.pdf>. Acesso em: 01 jun. 2021. Citado na página 13.

VIOLA, F. *Filtragem no domínio da freqüencia*. 2007. Disponível em: <http://www.ic-.uff.br/~aconci/filtragemdominiofrequencia.pdf>. Acesso em: 26 out. 2020. Citado na página 21.

WU, Y. *et al. Detectron2*. 2019. Disponível em: <https://github.com/facebookresearch-/detectron2>. Acesso em: 11 abr. 2021. Citado na página 63.