

Universidade de Brasília - UnB  
Faculdade UnB Gama - FGA  
Engenharia Eletrônica

# **Aplicação de Redes Neurais Convolucionais e Técnica de Janelamento para Contagem de Gado em Imagens Aéreas**

Autor: Marcelo dos Santos Junior  
Orientador: Prof. Dr. Gerardo Antonio Idrobo Pizo

Brasília, DF  
2023



Marcelo dos Santos Junior

## **Aplicação de Redes Neurais Convolucionais e Técnica de Janelamento para Contagem de Gado em Imagens Aéreas**

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Gerardo Antonio Idrobo Pizo

Brasília, DF

2023

---

Marcelo dos Santos Junior

Aplicação de Redes Neurais Convolucionais e Técnica de Janelamento para Contagem de Gado em Imagens Aéreas/ Marcelo dos Santos Junior. – Brasília, DF, 2023-

117 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Gerardo Antonio Idrobo Pizo

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB  
Faculdade UnB Gama - FGA , 2023.

1. Palavra-chave01. 2. Palavra-chave02. I. Prof. Dr. Gerardo Antonio Idrobo Pizo. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Aplicação de Redes Neurais Convolucionais e Técnica de Janelamento para Contagem de Gado em Imagens Aéreas

CDU 02:141:005.6

---

Marcelo dos Santos Junior

# **Aplicação de Redes Neurais Convolucionais e Técnica de Janelamento para Contagem de Gado em Imagens Aéreas**

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Trabalho aprovado. Brasília, DF, 26 de julho de 2023:

---

**Prof. Dr. Gerardo Antonio Idrobo  
Pizo**  
Orientador

---

**Prof. Dr. Jones Yudi Mori Alves da  
Silva**  
Convidado 1

---

**Prof. Dr. Marcus Vinicius Chaffim  
Costa**  
Convidado 2

Brasília, DF  
2023

# Resumo

A ampliação das técnicas de visão computacional vem proporcionando avanços em diversos setores. No agronegócio, as redes neurais convolucionais, assim como a massiva quantidade de dados adquiridos com o avanço da internet das coisas, propulsionam o Agro 4.0. Uma tarefa desafiadora para a pecuária inteligente do futuro é a contagem precisa e confiável do rebanho por meio de imagens. Este trabalho apresenta uma abordagem promissora para a contagem de gado em imagens aéreas, utilizando a arquitetura de redes neurais convolucionais *YOLOv8 Nano* 640x640 pixel e a técnica de janelamento. O objetivo do estudo foi desenvolver um modelo capaz de reconhecer e contar gados em imagens de grandes dimensões em ambientes descontrolados de pastagens de produção extensiva. Para isso, foi utilizado o conjunto de dados "*Cattle detection and counting in UAV images based on convolutional neural networks*" ([SHAO REI KAWAKAMI, 2020](#)). Inicialmente, foram criadas duas versões de modelo de contagem a partir de diferentes dimensões de janelas: uma com janelas de 500x600 pixels e outra com janelas de 200x200 pixels. Através da análise das métricas de validação, foi constatado que o modelo com janelas de 500x600 pixels apresentou resultados mais precisos e acurados, devido à proximidade com a dimensão da rede neural convolucionar utilizada no modelo de detecção e também pela menor probabilidade de contagem duplicada. Além disso, esse modelo final passou por técnicas de otimização, resultando em um terceiro modelo mais robusto e preciso. O modelo final obteve uma acurácia média de 92.7% e contou corretamente 91.4% das vacas presentes em todo o conjunto de imagens de validação. Os resultados demonstram a viabilidade da abordagem proposta para a contagem de gado em ambientes descontrolados. No entanto, sugere-se a repetição do experimento com conjuntos de imagens mais diversificados e a aplicação do modelo em um mosaico de imagens construído a partir do mapeamento usando drones. Essa abordagem pode trazer contribuições significativas para aplicações práticas em monitoramento de rebanhos e pecuária de precisão.

**Palavras-chaves:** Contagem de gado, detecção de objetos, aprendizado profundo, YOLOv8 Nano, gerenciamento de gado, visão computacional, monitoramento de rebanhos, detecção e contagem em imagens de UAV.

# Abstract

The expansion of computer vision techniques has been providing advances in various sectors. In agribusiness, convolutional neural networks, along with the massive amount of data acquired through the Internet of Things, drive Agro 4.0. A challenging task for the intelligent livestock of the future is the precise and reliable counting of the herd through images. This work presents a promising approach for cattle counting in aerial images, utilizing the YOLOv8 Nano 640x640 pixel convolutional neural network architecture and the windowing technique. The objective of the study was to develop a model capable of recognizing and counting cattle in large-scale images in uncontrolled environments of extensive grazing pastures. For this purpose, the dataset "Cattle detection and counting in UAV images based on convolutional neural networks" ([SHAO REI KAWAKAMI, 2020](#)) was used. Initially, two counting model versions were created based on different window dimensions: one with windows of 500x600 pixels and another with windows of 200x200 pixels. Through the analysis of validation metrics, it was found that the model with 500x600 pixel windows presented more precise and accurate results, due to its proximity to the dimension of the convolutional neural network used in the detection model and also due to the lower probability of duplicate counting. Additionally, this final model underwent optimization techniques, resulting in a third, more robust and accurate model. The final model achieved an average accuracy of 92.7% and correctly counted 91.4% of the cows in the entire validation image set. The results demonstrate the feasibility of the proposed approach for cattle counting in uncontrolled environments. However, it is suggested to repeat the experiment with more diversified image datasets and apply the model in a mosaic of images constructed from mapping using drones. This approach can bring significant contributions to practical applications in herd monitoring and precision livestock farming.

**Key-words:** Cattle counting, object detection, deep learning, YOLOv8 Nano, livestock management, computer vision, herd monitoring, UAV image detection and counting.

# Listas de ilustrações

Figura 1 – Arquitetura de uma rede neural convolucional . . . . .	22
Figura 2 – Operação de <i>padding</i> . . . . .	23
Figura 3 – Operação de <i>pooling</i> . . . . .	24
Figura 4 – Linha do tempo das versões da Yolo . . . . .	25
Figura 5 – Processo de detecção da arquitetura Yoloo . . . . .	26
Figura 6 – Comparação de performance entre as versões Yolo sobre o conjunto de dados COCO . . . . .	27
Figura 7 – Tabela de resultados do YoloV8 avaliada no conjunto de dados <i>COCO</i> e <i>ImageNet</i> . . . . .	27
Figura 8 – Arquitetura da Yolo v8 . . . . .	28
Figura 9 – Interface da ferramenta de rotulagem de dados para imagens <i>Label Studio</i> . . . . .	29
Figura 10 – Exemplo do formato de anotação <i>Pascal VOC</i> . . . . .	30
Figura 11 – Exemplo do formato de anotação <i>COCO</i> . . . . .	30
Figura 12 – Matriz de confusão . . . . .	32
Figura 13 – Imagem aérea original de dimensões 4000x3000 pixels de vacas em uma criação extensiva . . . . .	36
Figura 14 – Imagem aérea redimensionada para 640x640 pixels de vacas em uma criação extensiva . . . . .	37
Figura 15 – Técnica de janelamento aplicado à uma imagem aérea com dimensões de 4000x3000 pixels de vacas em uma criação extensiva . . . . .	37
Figura 16 – Ilustração do uso inapropriado da técnica de janelamento com janelas de 600x600 pixels aplicada em uma imagem original de 4000x3000 pixels . . . . .	38
Figura 17 – Arquitetura do modelo de contagem por janelamento. . . . .	42
Figura 18 – Fluxograma da coleta e preparação dos dados. . . . .	44
Figura 19 – Exemplo de imagem presente no <i>dataset Cattle detection and counting in UAV images based on convolutional neural networks</i> . . . . .	45
Figura 20 – Exemplo de arquivo de anotação no formato <i>Pascal VOC</i> . . . . .	46
Figura 21 – Exemplo de arquivo de anotação no formato <i>Yolo</i> . . . . .	46
Figura 22 – Fluxograma da aplicação do janelamento para a construção do conjunto de dados 1 e 2. . . . .	47
Figura 23 – Fluxograma do pré-processamento e divisão em treino e teste para os conjuntos de dados 1 e 2. . . . .	50
Figura 24 – Imagem de uma vaca dividida em duas janelas de 200x200 pixels . . . . .	51
Figura 25 – Fluxograma do pré-processamento e divisão em treino e teste para os conjuntos de dados 1 e 2. . . . .	53
Figura 26 – Fluxograma do processo de otimização do conjunto de dados . . . . .	58

Figura 27 – Imagens da classe <i>NoCow</i> que podem confundir o modelo . . . . .	59
Figura 28 – Exemplo de imagens da classe <i>Cow</i> pós <i>data augmentation</i> . . . . .	61
Figura 29 – Fluxograma de treinamento e validação do modelo de detecção e modelo de contagem v3 . . . . .	62
Figura 30 – Número de instâncias de cada classe no conjunto de dados utilizados no treinamento do modelo de detecção v1 . . . . .	63
Figura 31 – Métricas de erros no decorrer de 100 épocas do treinamento do modelo de detecção v1 . . . . .	64
Figura 32 – Métricas de precisão e <i>recall</i> à cada época durante o treinamento do modelo de detecção v1 . . . . .	65
Figura 33 – Curva precisão- <i>recall</i> para o modelo de detecção v1 . . . . .	65
Figura 34 – Métricas de <i>mean Average Precision</i> para cada época de treinamento do modelo de detecção v1 . . . . .	66
Figura 35 – Comparação entre as anotações reais e as previsões do modelo de detecção para um grupo de 14 imagens de teste . . . . .	67
Figura 36 – Matriz de confusão obtida no processo de validação do modelo de contagem v1 . . . . .	68
Figura 37 – Exemplo de uma contagem errada do modelo de contagem v1 em uma imagem do conjunto de validação . . . . .	69
Figura 38 – Exemplo de uma contagem realizada pelo modelo de contagem v1 em uma imagem do conjunto de validação . . . . .	69
Figura 39 – Número de instâncias de cada classe no conjunto de dados utilizados no treinamento do modelo de detecção v2 . . . . .	71
Figura 40 – Gráfico da curva de métricas de erro no decorrer de cada uma das 100 épocas . . . . .	72
Figura 41 – Métricas de precisão e <i>recall</i> à cada época durante o treinamento do modelo de detecção v2 . . . . .	72
Figura 42 – Curva precisão- <i>recall</i> para o modelo de detecção v2 . . . . .	73
Figura 43 – Métricas de <i>mean Average Precision</i> para cada época de treinamento do modelo de detecção v2 . . . . .	74
Figura 44 – Comparação entre as anotações reais e as previsões do modelo de detecção v2 para um grupo de 14 imagens de teste . . . . .	74
Figura 45 – Matriz de confusão obtida no processo de validação do modelo de contagem v2 . . . . .	76
Figura 46 – Exemplo de uma contagem errada do modelo de contagem v2 em uma imagem do conjunto de validação . . . . .	77
Figura 47 – Exemplo de uma contagem realizada pelo modelo de contagem v2 em uma imagem do conjunto de validação . . . . .	77

Figura 48 – Número de instâncias de cada classe no conjunto de dados utilizados no treinamento do modelo de detecção v3 . . . . .	79
Figura 49 – Métricas de erros obtidas durante as 150 épocas de treinamento do modelo de detecção v3 . . . . .	80
Figura 50 – Métricas de precisão, <i>recall</i> e mAP à cada época durante o treinamento do modelo de detecção v3 . . . . .	81
Figura 51 – Curva precisão- <i>recall</i> para o modelo de detecção v3 . . . . .	82
Figura 52 – Comparação entre as anotações reais e as previsões do modelo de detecção v3 para um grupo de 14 imagens de teste . . . . .	83
Figura 53 – Matriz de confusão obtida no processo de validação do modelo de contagem v3 . . . . .	84
Figura 54 – Exemplo de uma contagem errada do modelo de contagem v3 em uma imagem do conjunto de validação . . . . .	85
Figura 55 – Exemplo de uma contagem realizada pelo modelo de contagem v3 em uma imagem do conjunto de validação . . . . .	86
Figura 56 – Arquitetura de um Perceptron. . . . .	100
Figura 57 – Curva característica da função sinal. . . . .	101
Figura 58 – Arquitetura de um multilayer perceptron. . . . .	102
Figura 59 – Curva característica da função sigmóide. . . . .	104
Figura 60 – Curva característica da derivada da função sigmóide. . . . .	105
Figura 61 – Curva característica da função tangente hiperbólica. . . . .	106
Figura 62 – Curva característica da derivada da função tangente hiperbólica. . . . .	106
Figura 63 – Curva característica da função ReLU. . . . .	107
Figura 64 – Curva característica da derivada da função ReLU. . . . .	108
Figura 65 – Curva característica da função Leaky ReLU e PReLU. . . . .	109
Figura 66 – Curva característica da derivada da função Leaky ReLU e PReLU. . . . .	109
Figura 67 – Arquitetura de rede neural feedforward. . . . .	112
Figura 68 – Backpropagation. . . . .	114
Figura 69 – Resultados totais da validação do modelo de contagem v1 . . . . .	115
Figura 70 – Resultados totais da validação do modelo de contagem v2 . . . . .	116
Figura 71 – Resultados totais da validação do modelo de contagem v3 . . . . .	117

# **Lista de abreviaturas e siglas**

CNN	Convolutional Neural Network
CPU	Central Process Unit
GPU	Graphics Processing Unit
MAE	Mean Absolute Error
MNIST	Modified National Institute of Standards and Technology database
MSE	Mean Squared Error
RCNN	Region Based Convolutional Neural Networks
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RMSE	Root Mean Square Error
SVM	Support vector machine
YOLO	You Only Look Once

# Lista de símbolos

$b$	: Bias
$b'$	: Bias atualizado
$E$	: Função de custo
$h$	: Resultado do produto entre inputs e pesos
$Lr$	: Leaky ReLu
$Lr'$	: Derivada Leaky ReLu
$n$	: Número de camadas ocultas
$N$	: Regiões de interesse
$\nabla E$	: Gradiente da função de custo
$\beta$	: Learning rate
$\epsilon$	: Cross Entropy
$\hat{y}$	: Output da rede neural
$P_r$	: PReLU
$P_r'$	: Derivada PReLU
$R$	: ReLU
$R'$	: Derivada ReLU
$S$	: Softmax
$S'$	: Derivada softmax
$tanh$	: Tangente hiperbólica
$tanh'$	: Derivada tangente hiperbólica
$w$	: Pesos tensoriais
$w'$	: Peso atualizado
$x$	: Vetor de entrada

$y$  : label  
 $\lambda$  : Função de ativação  
 $\sigma$  : Sigmoide  
 $\sigma'$  : Derivada sigmoide

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>15</b>
1.1	Contextualização	15
1.2	Justificativa	17
1.3	Objetivo Geral	18
1.3.1	Objetivo Específico	18
1.4	Organização do Documento	19
<b>2</b>	<b>REFERENCIAL TEÓRICO</b>	<b>20</b>
2.1	Considerações Iniciais	20
2.2	Contexto Histórico	20
2.3	Rede neural convolucional (CNN)	22
2.3.1	<i>Padding</i>	23
2.3.2	<i>Pooling</i>	24
2.4	<b>Yolo V8</b>	<b>25</b>
2.5	<b>Anotações</b>	<b>28</b>
2.5.1	Formato Pascal VOC	29
2.5.2	Formato COCO	30
2.5.3	Formato YOLO	31
2.6	<b>Métricas de avaliação dos modelos</b>	<b>31</b>
2.6.1	A matriz de confusão	31
2.6.2	A precisão	33
2.6.3	O <i>recall</i>	33
2.6.4	O <i>F1 Score</i>	33
2.6.5	A acurácia	33
2.6.6	A diferença na contagem	34
2.6.7	Tempo de contagem (TC)	34
2.7	<b>Data Augmentation</b>	<b>34</b>
2.8	<b>Janelamento</b>	<b>36</b>
2.9	<b>Considerações Finais</b>	<b>38</b>
<b>3</b>	<b>TRABALHOS RELACIONADOS</b>	<b>40</b>
3.1	Técnicas clássicas de processamento de imagem	40
3.2	Técnicas avançadas de aprendizado de máquina	40
<b>4</b>	<b>METODOLOGIA</b>	<b>42</b>
4.1	Parte I: A escolha das janelas	43

4.1.1	Fase 1: Coleta e preparação dos dados . . . . .	43
4.1.1.1	Datasets . . . . .	44
4.1.1.2	Preparação das anotações dos conjuntos de dados . . . . .	45
4.1.1.3	Seleção do conjunto de dados de validação . . . . .	46
4.1.2	Fase 2: Aplicação do janelamento e a construção do conjunto de dados 1 e 2	47
4.1.2.1	Aplicação do janelamento . . . . .	48
4.1.2.2	Conjunto de dados 1 . . . . .	49
4.1.2.3	Conjunto de dados 2 . . . . .	49
4.1.3	Fase 3: Pré-processamentos e divisão das imagens em conjuntos de treino e teste . . . . .	49
4.1.3.1	Pré-processamentos . . . . .	51
4.1.3.2	Construção dos conjuntos de dados de treino e teste . . . . .	51
4.1.4	Fase 4: Construção dos modelos de detecção e contagem . . . . .	52
4.1.4.1	O Modelo de Detecção: . . . . .	53
4.1.4.1.1	O processo de treinamento do modelo de detecção . . . . .	53
4.1.4.1.2	Processo de validação do modelo de detecção . . . . .	54
4.1.4.2	Modelo de Contagem . . . . .	55
4.1.4.2.1	Calibração do hiperparâmetro do modelo de contagem . . . . .	55
4.1.4.2.2	Validação do modelo de contagem . . . . .	56
<b>4.2</b>	<b>Parte II - Otimização do modelo . . . . .</b>	<b>57</b>
4.2.1	Criação do conjunto de dados 3 . . . . .	58
<b>5</b>	<b>RESULTADOS E DISCUSSÕES . . . . .</b>	<b>63</b>
5.0.1	Conjunto de dados 1 . . . . .	63
5.0.1.1	Resultados do treinamento do modelo de detecção v1 . . . . .	63
5.0.1.2	Resultados do modelo de contagem v1 . . . . .	67
5.0.2	Conjunto de dados 2 . . . . .	71
5.0.2.1	Resultados do treinamento do modelo de detecção v2 . . . . .	71
5.0.2.2	Resultados do modelo de contagem v2 . . . . .	75
5.0.3	Conjunto de dados 3 . . . . .	79
5.0.3.1	Resultados do treinamento do modelo de detecção v3 . . . . .	79
5.0.3.2	Resultados do modelo de contagem v3 . . . . .	83
5.0.4	Comparação entre os modelos . . . . .	86
5.0.4.1	Comparação entre os modelos de detecção . . . . .	86
5.0.4.2	Comparação entre os modelos de contagem . . . . .	87
<b>6</b>	<b>CONCLUSÃO . . . . .</b>	<b>90</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>92</b>

<b>APÊNDICE A – CONCEITOS FUNDAMENTAIS DE <i>DEEP LEARNING</i></b> . . . . .	<b>100</b>
<b>A.1</b> <b>Rede neural de uma camada</b> . . . . .	<b>100</b>
<b>A.2</b> <b>Rede neural de múltiplas camadas</b> . . . . .	<b>101</b>
<b>A.3</b> <b>Funções de ativação</b> . . . . .	<b>103</b>
A.3.1   Softmax . . . . .	103
A.3.2   Sigmóide . . . . .	103
A.3.3   Tangente Hiperbólica . . . . .	105
A.3.4   ReLU . . . . .	107
A.3.5   Leaky ReLU e PReLU . . . . .	108
<b>A.4</b> <b>Função de perda</b> . . . . .	<b>109</b>
A.4.1   Funções de perdas para problemas de regressão . . . . .	110
A.4.2   Funções de perdas para problemas de classificação . . . . .	110
<b>A.5</b> <b>Otimização</b> . . . . .	<b>112</b>
A.5.1   Feedforward . . . . .	112
A.5.2   Gradiante Descendente . . . . .	113
A.5.3   Backpropagation . . . . .	114
<b>APÊNDICE B – RESULTADOS COMPLETOS DA VALIDAÇÃO DOS MODELOS DE CONTAGEM</b> . . . . .	<b>115</b>

# 1 Introdução

## 1.1 Contextualização

De acordo com o relatório das Nações Unidas publicado em 2019, estima-se que a população mundial chegará a 9,8 bilhões de pessoas em 2050 ([ONU, 2019](#)). Desse modo, um grande desafio de hoje, intensificado pela área de terra limitada, é a necessidade de produzir mais culturas para alimentar a população cada vez maior, especialmente para os países em desenvolvimento como o Brasil. Ainda, processos ineficientes e o uso excessivo de produtos químicos nas plantações, a fim de aumentar a fertilidade do solo e reduzir ervas daninhas e pragas, tem afetado negativamente o meio ambiente e a saúde humana, mostrando-se inviáveis para a agricultura do futuro. A forte necessidade de produzir mais culturas a partir de menos terra de maneira mais ecológica levou a vários desafios no campo da agricultura ([PANDEY et al., 2022](#)).

Desse modo, os avanços tecnológicos da informação e comunicação (TIC), que há décadas vêm contribuindo de forma impactante em diversas áreas do conhecimento, passam a ter um caráter estratégico e político no mundo globalizado atual, permitindo o armazenamento e processamento de grandes volumes de dados, automatização de processos e o intercâmbio de informações e de conhecimento auxiliando a tomada de decisão ([MASSRUHÁ et al., 2014](#)).

A agricultura do futuro tem como desafios principais: aumentar a produtividade e a qualidade dos alimentos, reduzir os custos operacionais e otimizar o uso de insumos. Contudo, os avanços de tecnologias como a inteligência artificial, em especial a visão computacional, nos permitem automatizar e acelerar as operações de campo dando origem a um novo ramo da Indústria 4.0, chamado Agricultura 4.0. Esse tipo de agricultura inteligente integra dados e informações para monitorar as atividades de campo por meio da aplicação de sensoriamento remoto e proximal ([ANTONUCCI et al., 2019](#)).

O uso das TIC e das novas tecnologias digitais é um caminho sem volta na era da Agro 4.0. Isso porque proporcionam inovação dentro e fora da cadeia produtiva. As TIC estão presentes na pré-produção, no melhoramento genético e na bioinformática; durante a produção, na agricultura de precisão e diversos equipamentos e no pós-produção, com melhorias na logística e transporte. Todas estas tecnologias e inovações estarão cada vez mais conectadas, auxiliando na tomada de decisão e gestão rural ([MASSRUHÁ; LEITE, 2017](#)).

Nesse cenário, a inteligência artificial (IA) surgiu como uma tecnologia promissora na agricultura digital. Diversos trabalhos desenvolvidos nos últimos anos têm trazido

resultados positivos em diversas frentes da agricultura mundial. Avanços em problemas como na detecção de doenças, classificação de culturas, identificação de plantas e pragas, contagens, tiveram a contribuição de importantes pesquisadores.

No seu trabalho de 2015, Yusuke Kawasaki contribuiu propondo uma rede neural convolucional especializada na detecção de doenças na folha do pepino obtendo uma acurácia 94,90% na detecção de Vírus do mosaico amarelo (KAWASAKI et al., 2015). Em 2016, Srdjan Sladojevic, propôs um modelo de detecção baseado em CaffeNet capaz de classificar 13 variedades de doenças em folhas obtendo uma acurácia de 96,30% (SLADOJEVIC et al., 2016). Em 2020, o estado da arte foi alcançado por Junde Chen ao propor uma rede neural convolucional profunda especializada na detecção de doenças em folhas obtendo uma acurácia média de 92% em imagens de culturas de arroz. (CHEN et al., 2020)

Em relação aos problemas de classificação, os estudos de Sharada P. Mohanty realizados em 2016 contribuíram com a identificação de 26 doenças e 14 espécies de culturas usando o framework CaffeNet obtendo um F1 score de 0.9935 (MOHANTY; HUGHES; SALATHÉ, 2016). Em 2017, foi a vez dos estudos (LU et al., 2017) sobre detecção de doenças no trigo, (AMARA; BOUAZIZ; ALGERGAWY, 2017) na detecção de doenças em folhas de bananeiras, (BRAHIMI; KAMEL; MOUSSAOUI, 2017) na classificação de 9 doenças em tomateiros, (KUSSUL et al., 2017) classificação de culturas como trigo, soja, batata doce, girassol e milho por meio de imagens de satélite junto a arquitetura de um multilayer perceptron.

Outros estudos desenvolvidos foram em relação a estimar o dimensionamento de áreas de cultivo. Nessa frente, destacam-se os estudos de (KUWATA; SHIBASAKI, 2015) que utilizou CaffeNet para estimar área de cultivo de plantações de milho e (KAMILARIS; PRENAFETA-BOLDU, 2018) que propôs uma rede neural convolucional para estimar áreas de cultivo de cana de açúcar em Costa Rica.

Já os problemas de contagem, em 2017, Steven W. Chen propôs contar laranjas e maçã utilizando deep learning (CHEN et al., 2017). Os trabalhos de Nicolai Häni de 2018 contribuíram com uma otimização na performance de algoritmos de contagem por meio de Faster R-CNN (HÄNI; ROY; ISLER, 2020) e os estudos de Daniel I. Onwude demonstraram uma forma simples, rápida e não destrutiva para distinguir e separar morangos danificados após processo de congelamento. (ONWUDE et al., 2018)

O uso da inteligência artificial não se limita a agricultura, na pecuária, as redes neurais convolucionais são utilizadas na classificação de diferentes raças de gado como no trabalho (SANTONI et al., 2015), Detecção precoce e alerta de problemas de curvas de produção de ovos de galinha (MORALES et al., 2016), Estimativa do peso de gados (ALONSO; VILLA; BAHAMONDE, 2015) entre outros.

## 1.2 Justificativa

As aplicações de Inteligência Artificial podem criar uma grande revolução no setor da pecuária, por meio de possibilidades de automação e transformação de dados em informação, em tempo real. Promovendo, assim, uma agricultura conectada, intensiva em conhecimento, com altos níveis de produtividade e de sustentabilidade, com redução de custos e melhoria nas condições de trabalho no campo ([AGRISHOW, 2016](#)).

Pensando nisso, ([PANDEY et al., 2022](#)) realizou uma revisão de literatura abrangendo as principais técnicas e aplicações de inteligência artificial em diversos setores da agricultura e da pecuária. Dentre elas, diversos trabalhos que visam aprimorar as técnicas para monitorar e gerenciar rebanhos em tempo real.

Na criação de animais, a atividade de monitorar a produção é essencial na gestão das fazendas pecuárias. Entretanto, essa atividade requer uma quantidade de recursos, desde financeiros até humanos. Em grandes propriedades que adotam a pecuária extensiva, muito comuns no Brasil, esse problema é intensificado ([BARBEDO et al., 2020](#)).

Monitorar e gerenciar os rebanhos em tempo real tem como benefício a possibilidade de uma gestão do uso de recursos naturais de maneira otimizada, aumentando a eficiência, os ganhos na produtividade e o bem-estar do animal. Ainda, auxiliam na tomada de decisão, por parte do produtor rural, sobre assuntos estratégicos, tornando sua produção mais competitiva.

Uma das tarefas do monitoramento e gerenciamento dos rebanhos é a contagem periódica dos animais. Geralmente essa tarefa ocorre reunindo os animais no curral com o auxílio de diversos funcionários montados a cavalo. Durante esse tipo de manejo os animais estão sujeitos a situações estressantes causando impactos diretos no bem-estar e na produtividade do rebanho. Por despender grandes quantidades de recursos, geralmente essa atividade é realizada de forma secundária, juntamente a alguma outra atividade, como por exemplo a vacinação impossibilitando, assim, a geração de dados em tempo real ([BARBEDO et al., 2020](#)).

Entretanto, os avanços no estudo de aplicação de técnicas de redes neurais e visão computacional na contagem de animais trouxeram resultados animadores. Em 2014, as redes neurais convolucionais propostas por Chamoso obtiveram uma acurácia de 96.2% ([SANTOS PABLO; RAVEANE WILLIAM, 2014](#)). No ano de 2020, a contagem de rebanhos tiveram a contribuição dos trabalhos ([BARBEDO et al., 2020](#)) e ([SHAO et al., 2020](#)). Esse último, baseado em um modelo de arquitetura YOLOv2 ([REDMON; FARHADI, 2016](#)), proporcionou avanços na contagem de animais em áreas extensas ([SHAO et al., 2020](#)). Outro trabalho relevantes são ([SOARES et al., 2021](#)); ([HAN; TAO; MARTIN, 2019](#)); ([WEBER et al., 2020](#)).

Com o desenvolvimento das redes neurais convolucionais profundas, como R-CNN

(GIRSHICK et al., 2013), Fast R-CNN (GIRSHICK, 2015), Faster R-CNN (REN et al., 2015) e Mask R-CNN (HE et al., 2017) intensificaram ainda mais os avanços nas técnicas de contagem de rebanhos. No seu trabalho de 2017 Andrew utilizou Faster R-CNN para identificar 30 vacas leiteiras em ambiente confinado obtendo 99.3% de acurácia (ANDREW; GREATWOOD; BURGHARDT, 2017). O estado da arte deu-se em 2020 com a aplicação da arquitetura Mask R-CNN obtendo 94% em uma produção extensiva e 92% de acurácia em uma produção confinada (XU et al., 2020a).

A contagem de gado por meio de visão computacional é uma tarefa desafiadora. Cenas mais complexas com baixo contraste entre diferentes animais, presença de sombras, obstáculos como a vegetação, intensidade da luz, contagem duplicada de animais são problemas que devem ser levados em consideração (BARBEDO et al., 2020).

Diante disso, a fim de contribuir com os avanços nas metodologia para a agrimensura pecuária, esse projeto tem como objetivo construir um modelo de *deep learning*, associado a técnica de janelamento, capaz de reconhecer e contar gados em imagens aéreas de um ambiente não controlado em pastagens de produção extensiva.

## 1.3 Objetivo Geral

O objetivo deste trabalho consistiu em aplicar a arquitetura de rede neural convolucional especializada no reconhecimento de objetos junto à técnica de janelamento, a fim de construir um modelo capaz de reconhecer e contar gados em imagens aéreas com dimensões, relativamente, grandes em diferentes situações de ambiente descontrolado de pastagens de produção extensiva.

### 1.3.1 Objetivo Específico

Para alcançar o objetivo geral proposto, o trabalho se desdobrará nas seguintes metas específicas:

- Realizar uma revisão bibliográfica detalhada sobre as Redes Neurais Convolucionais, técnicas de janelamento e suas aplicações no contexto de visão computacional e contagem de objetos em imagens aéreas.
- Coletar e preparar um conjunto de dados representativo, composto por imagens aéreas que contenham rebanhos bovinos em diferentes cenários, abrangendo variações de iluminação, ângulos de captura e densidades de gado.
- Implementar e treinar uma arquitetura de Rede Neural Convolucional adequada para a tarefa de contagem de gado, explorando diferentes configurações e otimizações para obter um modelo de alta precisão.

- Avaliar quantitativamente o desempenho do sistema desenvolvido, medindo a acurácia da contagem em comparação com a contagem manual, e analisar outras métricas relevantes para verificar a eficiência e a escalabilidade da solução.
- Verificar a relação do tamanho do janelamento e o impacto na duplicitade na contagem.
- Discutir as contribuições e limitações do sistema desenvolvido, fornecendo *insights* para possíveis melhorias e futuras pesquisas na área de contagem automatizada de gado utilizando técnicas de visão computacional.

Ao cumprir com esses objetivos, este trabalho busca contribuir para o avanço da tecnologia aplicada à pecuária e, também outros setores, e oferecer uma ferramenta promissora que possa ser adotada na indústria agropecuária para otimizar o monitoramento e a contagem do rebanho bovino, tornando esse processo mais preciso, eficiente e confiável.

## 1.4 Organização do Documento

Este documento é dividido em seis capítulos onde no primeiro capítulo é constituído por uma contextualização, uma justificativa para o desenvolvimento do trabalho e a definição dos objetivos gerais e específicos do projeto.

O segundo capítulo refere-se ao referencial teórico. Ele é constituído de uma contextualização histórica sobre *Deep Learning* e em seguida, aborda os avanços teóricos relevantes para o desenvolvimento deste projeto: desde os fundamentos do Perceptron, passando pelo estado da arte das redes neurais convolucionais com a arquitetura do *YoloV8*, até as técnicas de *data augmentation* utilizadas para a otimização de modelos de inteligencia artificial.

O terceiro capítulo traz alguns trabalhos relacionados.

O quarto capítulo refere-se à metodologia utilizada para desenvolver a proposta de solução.

O quinto capítulo discute os resultados obtidos após a construção do modelo de detecção e contagem.

O sexto capítulo refere-se a uma conclusão sobre o trabalho realizado, evidenciando as contribuições e a importância do estudo. Por fim, apresenta sugestões para trabalhos futuros.

## 2 Referencial Teórico

### 2.1 Considerações Iniciais

Esta seção tem a finalidade de reunir todo o conhecimento teórico necessário para o desenvolvimento do projeto. Desse modo, inicia-se com uma contextualização histórica sobre fatos relevantes que contribuíram com a evolução dos primeiros algoritmos de inteligência artificial até os atuais e sofisticados modelos convolucionais, geralmente empregados em projetos de visão computacional como este trabalho. Ainda, serão descritas algumas arquitetura de redes neurais, juntamente com conceitos fundamentais para o entendimento de modelos de *deep learning*, assim como seus processos de treinamento, otimização e avaliação de performance. Inicialmente, é descrita a arquitetura de rede mais simples, de uma camada, seguindo pelo *Multilayer Perceptron* e redes neurais convolucionais, finalizando com a família de redes neurais Yolo, incluindo a arquitetura YoloV8. Por fim, neste capítulo também é descrito, de forma resumida, a técnica de janelamento como pré-processamento de imagens com objetos relativamente pequenos.

### 2.2 Contexto Histórico

Desde o primeiro modelo matemático de um neurônio proposto por Warren McCulloch e Walter Pitts em 1943, denominado de *Threshold Logic Unit* ([MCCULLOCH; PITTS, 1943](#)), houveram diversos avanços significativos nos estudos de inteligência artificial, resultando no estado da arte: as redes neurais convolucionais. O conceito de modelo matemático de um neurônio consiste em representar um neurônio biológico com suas estruturas. Ou seja, pelos axônios, dendritos e corpo celular.

Inspirado pelas ideias de ([MCCULLOCH; PITTS, 1943](#)), Frank Rosenblatt propôs o *perceptron*, um modelo matemático de uma rede neural artificial de camada única, no seu artigo de 1958 ([ROSENBLATT, 1958](#)). Esse modelo era capaz de resolver problemas simples, linearmente separáveis. Entretanto, em 1969 Minsky e Papert provaram que embora o perceptron fosse capaz de aprender qualquer coisa que fosse capaz de representar, ele poderia representar muito pouco ([MINSKY; PAPERT, 1969](#)) e isso levou à um momento da história conhecido como o primeiro inverno da inteligência artificial. Mais detalhes sobre o perceptron serão abordados no tópico [A](#).

Rosenblatt, em 1962, ainda foi responsável por ser pioneiro nos estudos do modelo *Multilayer Perceptron* ([ROSENBLATT, 1962](#)). Esse modelo consiste em uma rede neural composta por mais de uma camada *perceptron*. Os estudos desse modelo de múltiplas

camadas tiveram, também, uma grandiosa contribuição de Alexey Ivakhnenko nos seus diversos artigos, em especial o de 1968 ([IVAKHnenko, 1968](#)) onde apresentou um algoritmo de aprendizado para o *Multilayer Perceptron* e serviu como base para o surgimento das redes neurais profundas.

O conceito de *backpropagation* foi introduzido por Seppo Linnainmaa em sua tese de mestrado ([SEppo, 1970](#)) e aprofundada mais tarde, em 1976, no seu artigo ([LINNAINMAA, 1976](#)). *backpropagation* consiste em minimizar a função de perda otimizando os coeficientes de pesos da rede neural. Isso permitiu que as redes neurais fossem capazes de aprender com exemplos, resultando em redes neurais adaptáveis, permitindo melhor desempenho em exemplos futuros.

A limitação das redes neurais de uma camada, resolver problemas simples linearmente separáveis, foi superada com a arquitetura do *Multilayer Perceptron* adicionado à técnica de *backpropagation*. Isso foi demonstrado por David Rumelhart no seu artigo de 1986 ([RUMELHART; HINTON; WILLIAMS, 1986](#)). Nesse artigo, mesmo se tratando de um problema não linearmente separável, Rumelhart conseguiu modelar a tabela verdade de uma porta lógica XOR, o que era impossível realizar com a arquitetura de um *perceptron*. Isso propulsionou o desenvolvimento do *Deep Learning*.

O termo *Deep Learning* refere-se ao aprendizado de máquina usando várias camadas de elementos de computação simples e ajustáveis ([HAYKIN, 2003](#)). Embora capazes de representar problemas linearmente e não linearmente separáveis, as redes neurais de múltiplas camadas tinham como desafio o tempo necessário para o seu treinamento. Em seu trabalho de 1989, Yann LeCun levou aproximadamente 3 dias para treinar uma rede neural profunda com o propósito de reconhecer códigos de CEP manuscritos em cartas aplicando o algoritmo de backpropagation padrão ([LECUN et al., 1998](#)). Porém, em 1991, Sepp Hochreiter contribuiu com a redução do tempo necessário para o treinamento por meio da dissipação do gradiente ([HOCHREITER, 1991](#)).

Desde então, surgiram diversas arquiteturas de Deep Learning que trouxeram resultados animadores para os problemas de reconhecimento de fala e também para o reconhecimento de imagens. Para o reconhecimento de fala, destaca-se a arquitetura CTC explorada por Alex Graves, Santiago Fernandez, Faustino Gomez e Jürgen Schmidhuber no seu trabalho de 2006 ([GRAVES et al., 2006](#)). Já para o reconhecimento de imagem, as redes neurais, que são o foco deste trabalho, se destacam.

Um dos primeiros a trabalhar com as redes neurais convolucionais foi Kunihiko Fukushima em seu trabalho de 1980 ([FUKUSHIMA, 1980](#)). Ainda, Yann LeCun trouxe grandes contribuições no reconhecimento de padrões em ([LECUN et al., 1998](#)), ([LECUN et al., 1990](#)) e ([CUN et al., 1990](#)). No seu trabalho de 1998, Yann LeCun utilizou o conjunto de dados MNIST. Esse conjunto de dados é famoso entre os iniciantes nos estudos de redes neurais da atualidade e consiste em imagens manuscritas de números naturais em tons de

cinza (LECUN et al., 1998).

No ano de 2012, outro marco na história das redes neurais aconteceu. Um sistema de aprendizado profundo criado pelo grupo Geoffrey Hinton na universidade de Toronto apresentou uma melhora significativa na competição ImageNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Essa competição consiste em reconhecer imagens em um banco com mil categorias e milhões de imagens.

Os resultados alcançados pelos pesquisadores de Toronto em 2012 impulsionaram o surgimento de outras redes neurais, tais como: a ZFNet em 2013 (ZEILER; FERGUS, 2014), a GoogLeNet (SZEGEDY et al., 2015) e a VGG NET (SIMONYAN; ZISSERMAN, 2015) em 2014, a ResNet (HE et al., 2016), rede neural considerada o estado da arte, entre outras.

## 2.3 Rede neural convolucional (CNN)

As redes neurais convolucionais, do inglês *convolutional neural network (CNN)*, são um tipo de arquitetura que utilizam operação de convolução em pelo menos uma de suas camadas a fim de extrair os diferentes recursos de uma imagem e montar um feature map (GOODFELLOW; BENGIO; COURVILLE, 2016). Esse tipo de arquitetura tem como principal aplicação na área de visão computacional processando imagens e vídeos.

A figura 1 representa a arquitetura de uma rede neural convolucional modelada para classificar imagens.

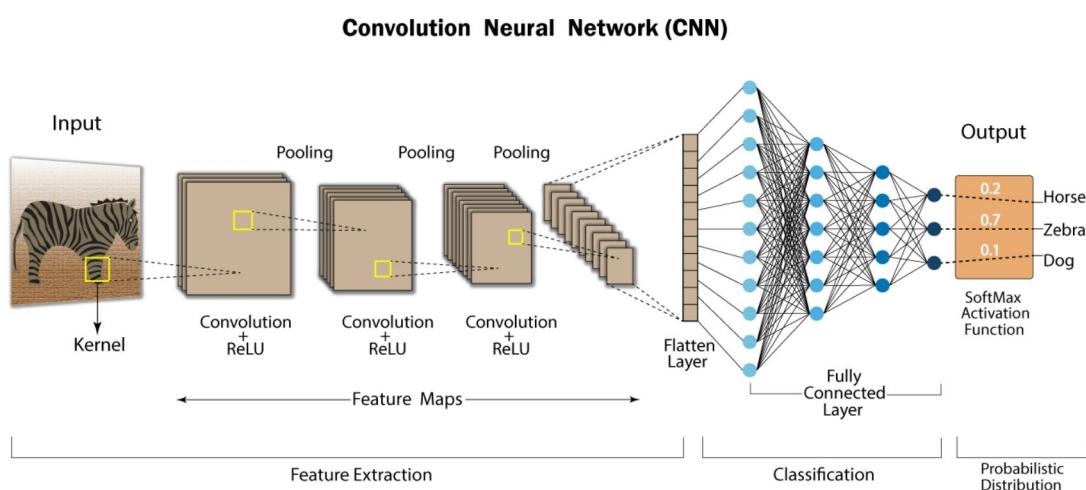


Figura 1 – Arquitetura de uma rede neural convolucional.

Fonte: (AGHDAM; HERAVI, 2018)

A arquitetura de uma rede neural convolucional, normalmente, consiste de uma camada de entrada, camadas ocultas e camada de saída. Nessas camadas ocultas é onde ocorrem as operações de convolução e das funções de ativação que normalmente é do tipo

ReLU, descrita na seção A.3.4, permitindo a esparsidade da rede, gerando um aumento de velocidade ([YAMASHITA et al., 2018](#)).

Em problemas de visão computacional, as entradas da CNN são tensores tridimensionais que carregam os dados na forma de largura, altura e profundidade. No caso de imagens, a dimensão de profundidade é determinada pelos canais de cores presentes representados pelo padrão RGB.

Já a camada convolucional tem a finalidade de gerar um feature map extraindo os diferentes recursos da entrada. A convolução reduz o número de parâmetros livres, permitindo que a rede seja mais profunda e capaz de modelar padrões mais complexos ([AGHDAM; HERAVI, 2018](#)).

A operação de convolução muda a dimensão da imagem. Isso pode ser controlado adicionando uma camada de *pooling* ou de *padding* de acordo com o objetivo do desenvolvedor.

### 2.3.1 Padding

A camada de *padding* tem como objetivo preservar a dimensionalidade e as informações contidas nas bordas da imagem após passar pela camada convolucional. Para isso é adicionado vários pixels de magnitude zero, antes da operação de convolução, aumentando a dimensão da imagem original ([AGGARWAL, 2018](#)). A figura 2 ilustra um exemplo de operação de *padding*.

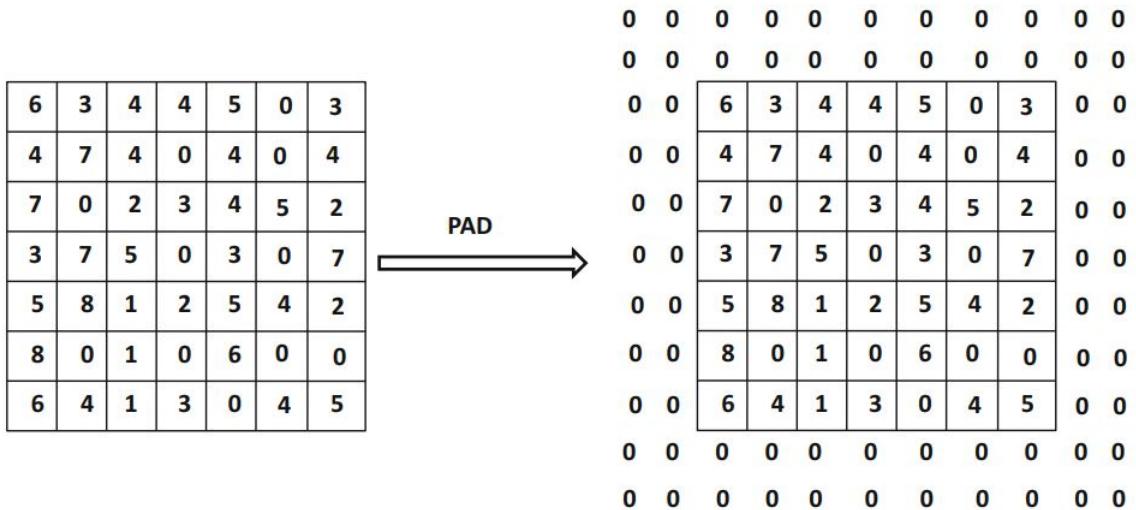


Figura 2 – Operação de *padding*.

Fonte: ([AGGARWAL, 2018](#))

### 2.3.2 Pooling

A camada de *pooling* é adicionada com o objetivo de simplificar e reduzir a dimensionalidade da imagem. Essa operação permite que informações irrelevantes sejam descartadas, reduzindo o custo computacional, e as informações relevantes para o modelo sejam evidenciadas (YAMASHITA et al., 2018). Existem dois tipos de operação de *pooling* comumente utilizados em CNNs para a construção da *feature map*: *pooling* máximo e *pooling* médio. O *pooling* máximo utiliza o valor máximo de cada cluster local de neurônios no mapa de recursos. Já o *pooling* médio utiliza o valor médio de cada cluster (CIRESAN et al., 2011).

A figura 3, ilustra uma operação do tipo *max pooling* para a redução de dimensionalidade de cluster local de tamanho 3x3 com *stride* igual a 1 e outro exemplo com *stride* igual a 2. O *stride* é o parâmetro que determina o passo que a convolução irá percorrer na imagem original possibilitando sobreposição ou não. Entretanto, é desejável ter pelo menos alguma sobreposição entre as unidades espaciais nas quais o agrupamento é realizado, porque torna a abordagem menos provável de *overfit* (CIRESAN et al., 2011).

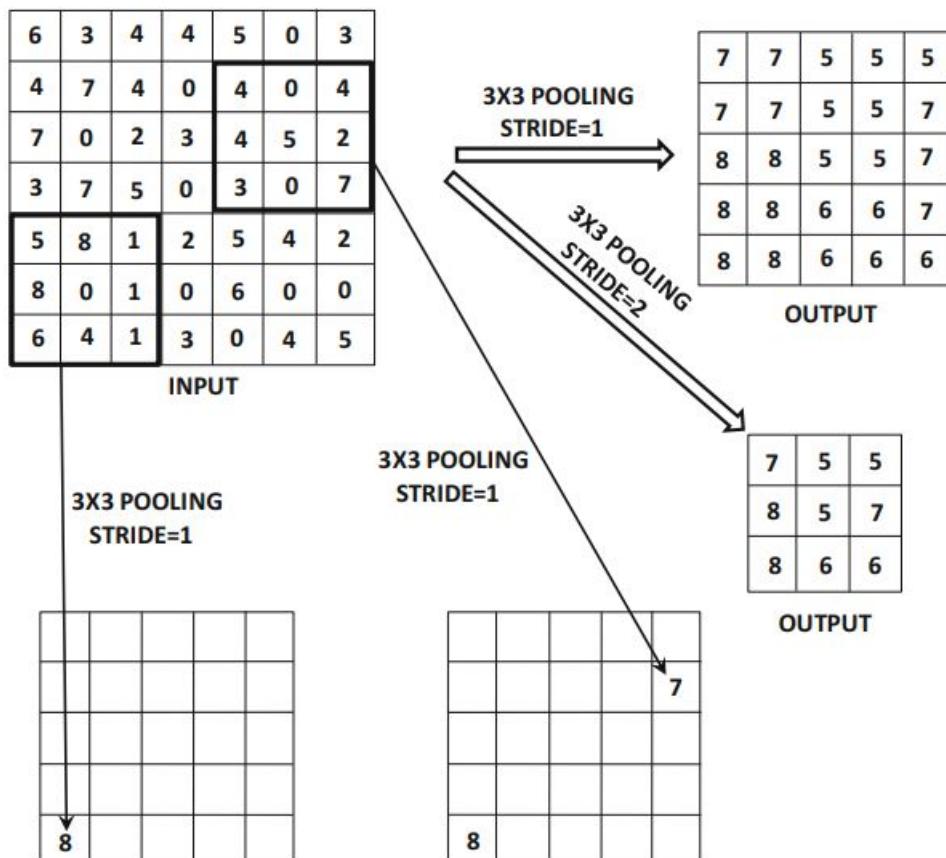


Figura 3 – Operação de *pooling*.

Fonte: (AGGARWAL, 2018)

## 2.4 Yolo V8

Uma arquitetura de rede neural convolucional utilizada com muita frequência em trabalhos de visão computacional e detecção de objetos em imagens é a arquitetura *YOLO* (*You Only Look Once*). Essa arquitetura foi apresentada pela primeira vez em 2015 (REDMON et al., 2015) e desde então vem sofrendo atualizações. A figura 4 ilustra a linha do tempo das versões da Yolo.

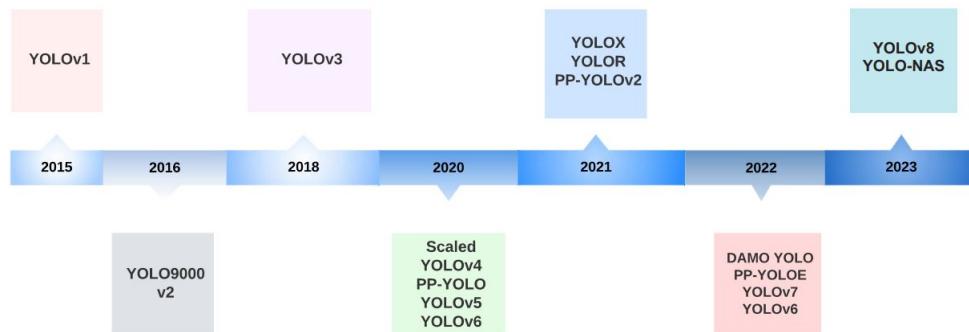


Figura 4 – Linha do tempo das versões da Yolo

Fonte: (TERVEN; CORDOVA-ESPARZA, 2023)

Desenvolvida para ser eficiente, rápida e precisa na identificação de múltiplos objetos, a arquitetura *YOLO* tornou-se especialmente adequada para aplicações em tempo real. A agilidade da arquitetura *YOLO* está no fato de realizar a detecção em apenas uma passagem pela rede neural convolucional. A figura 5 ilustra o processo de detecção dessa arquitetura.

O YOLO divide a imagem em uma *grid* de células e, em cada célula, prevê as caixas delimitadoras dos objetos, juntamente com a confiança de que um objeto está presente naquela célula e as probabilidades de classe dos objetos. Para isso, o modelo utiliza uma única rede neural para realizar todas essas previsões simultaneamente, o que acelera o processo.(REDMON et al., 2015)

Em seguida, ocorre uma filtragem pra desconsiderar as detecções com baixa confiança por meio de um parâmetro de limiar de confiança. Já para eliminar as detecções redundantes o modelo aplica um algoritmo de Supressão Não-Máxima, também chamada de *Non-Maximum Suppression (NMS)*.

Esse algoritmo começa a percorrer as caixas em ordem decrescente de pontuação, verificando se há sobreposição significativa com outras caixas que já foram selecionadas. Se a sobreposição for maior que um determinado limiar pré-definido, a caixa é descartada, pois provavelmente representa a mesma detecção que uma caixa anterior com pontuação semelhante e maior confiança. Dessa forma, o NMS ajuda a produzindo uma saída mais limpa e precisa para a detecção de objetos em imagens.

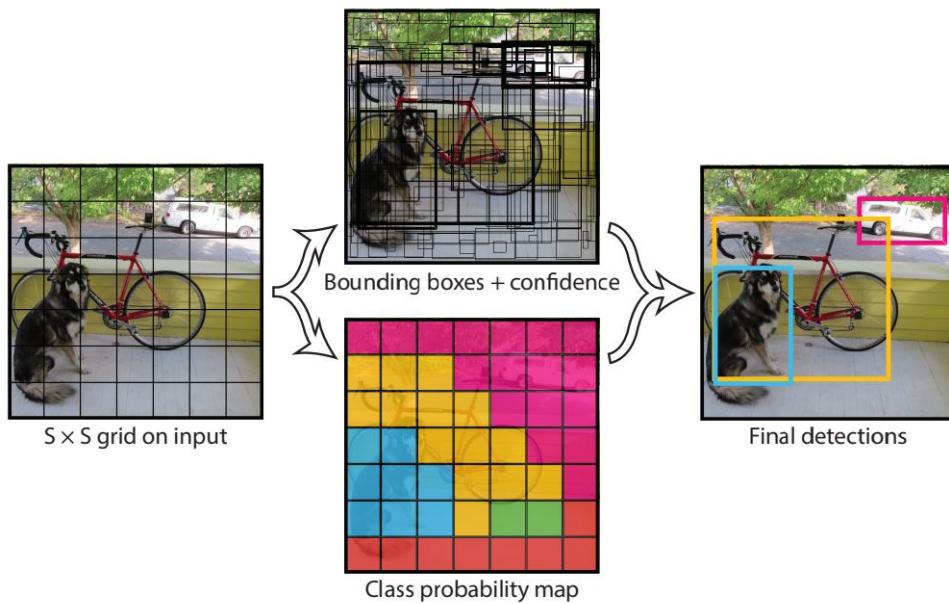


Figura 5 – Processo de detecção da arquitetura Yoloo

Fonte: ([REDMON et al., 2015](#))

Ainda, a arquitetura YOLO é capaz de lidar com várias classes de objetos simultaneamente e é flexível quanto ao número de caixas delimitadoras previstas por célula. Isso significa que é capaz de detectar objetos de tamanhos diferentes e com sobreposições entre si.

Baseado em versões anteriores, principalmente na versão 5, surge recentemente a YOLOv8 ([JOCHER; CHAURASIA; QIU, 2023](#)). Essa arquitetura criada pela *Ultralytics*, introduz novas funcionalidades e melhorias para aumentar ainda mais o desempenho e a flexibilidade. A figura 6 apresenta uma comparação entre as diferentes versões da YOLO. Nesse figura, é apresentado a *mean Average Precision* calculada em uma faixa de confiança de 50% a 95% para as detecções no conjunto de imagens COCO([LIN et al., 2015](#)). Esse figura evidencia a superioridade dessa nova arquitetura em relação as demais.

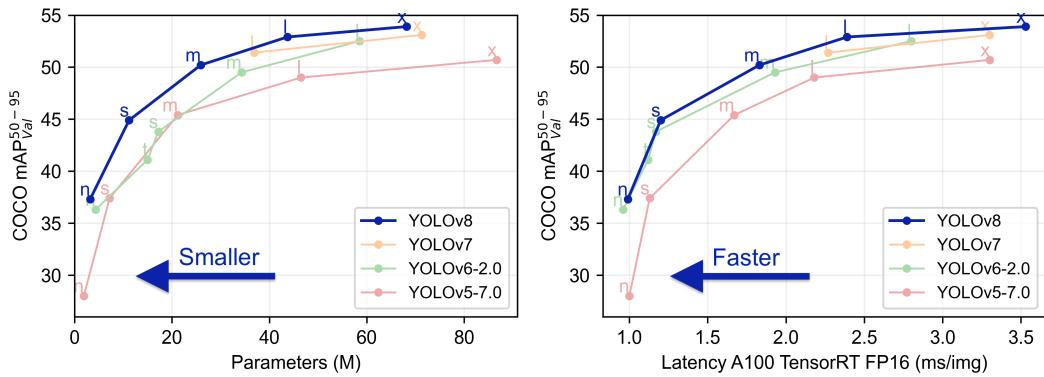


Figura 6 – Comparaçõa de performance entre as versões Yolo sobre o conjunto de dados COCO.

Fonte: ([JOCHER; CHAURASIA; QIU, 2023](#))

Já a figura 7 apresenta a tabela de resultados para diferentes versões do modelo YoloV8 treinado e avaliado com os conjunto de dados *COCO* ([LIN et al., 2015](#)) e *ImageNet* ([RUSSAKOVSKY et al., 2015](#)). Por fim, a figura 8 apresenta a arquitetura da Yolo v8.

Model	size (pixels)	mAP <sub>50-95</sub>	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Figura 7 – Tabela de resultados do YoloV8 avaliada no conjunto de dados *COCO* e *ImageNet*

Fonte: ([JOCHER; CHAURASIA; QIU, 2023](#))

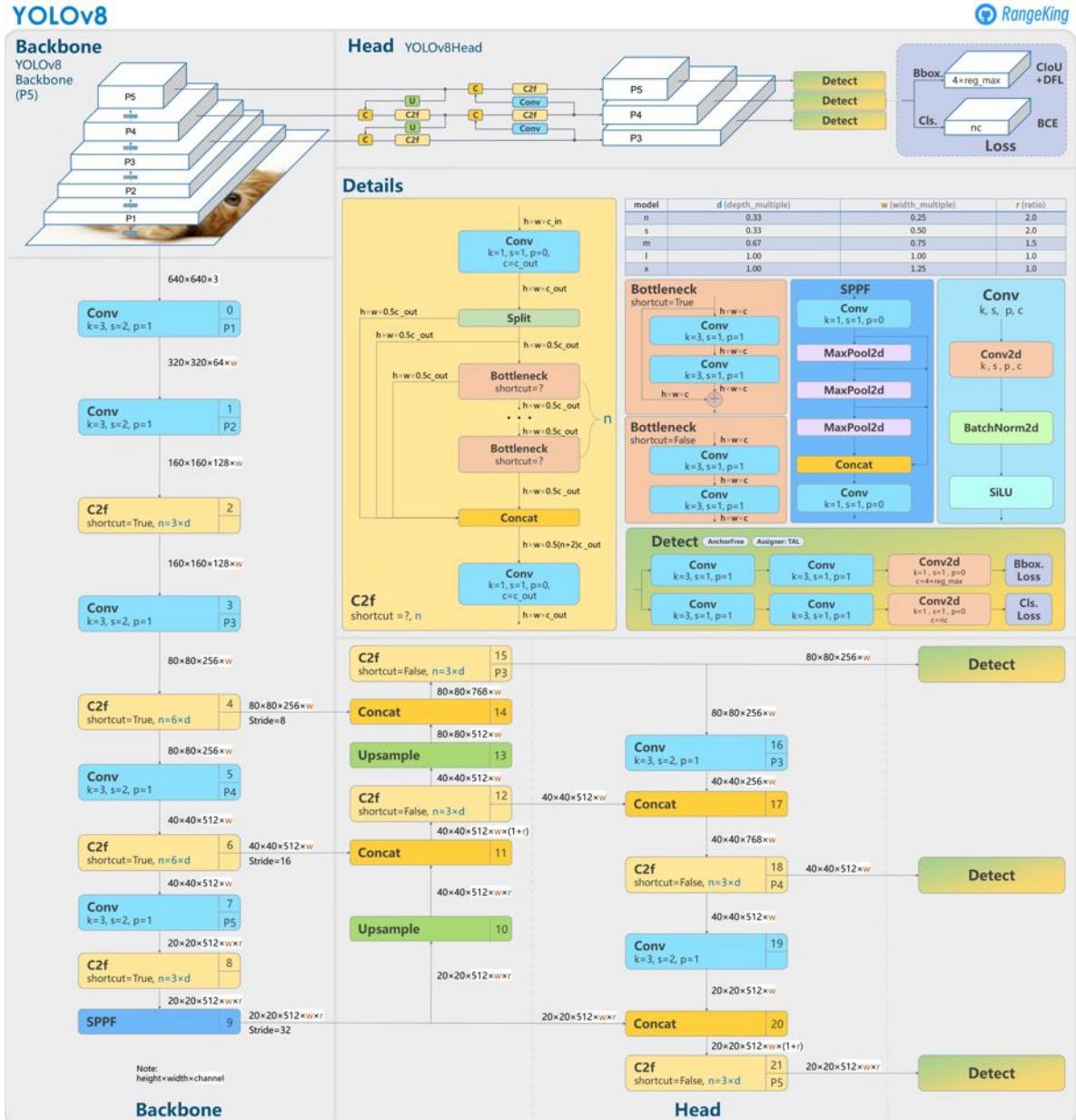


Figura 8 – Arquitetura da Yolo v8

Fonte: (REDMON et al., 2015)

## 2.5 Anotações

As anotações são cruciais para o desenvolvimento e treinamento de modelos supervisionados de detecção de objetos. Isso porque fornecem os rótulos necessários para que o modelo aprenda a identificar e localizar os objetos de interesse em novas imagens ou vídeos.

Essas anotações, geralmente, são arquivos que contêm informações sobre a localização e a classe dos objetos em uma imagem. Essas anotações, posteriormente, são utilizadas como "gabaritos" para treinar e avaliar algoritmos de detecção de objetos, como redes neurais convolucionais descritas na seção 2.2.

Geralmente, essas anotações são construídas manualmente com auxílio de softwares de rotulagem de dados para imagens. Esse processo requer que o agente humano marque a localização dos objetos de interesse em cada imagem. Um exemplo de ferramenta de rotulagem de dados para imagens é o *Label Studio* ([TZUTALIN, 2018](#)), uma ferramenta de código aberto desenvolvida por *Tzutalin*. A figura 9 ilustra a interface dessa ferramenta.

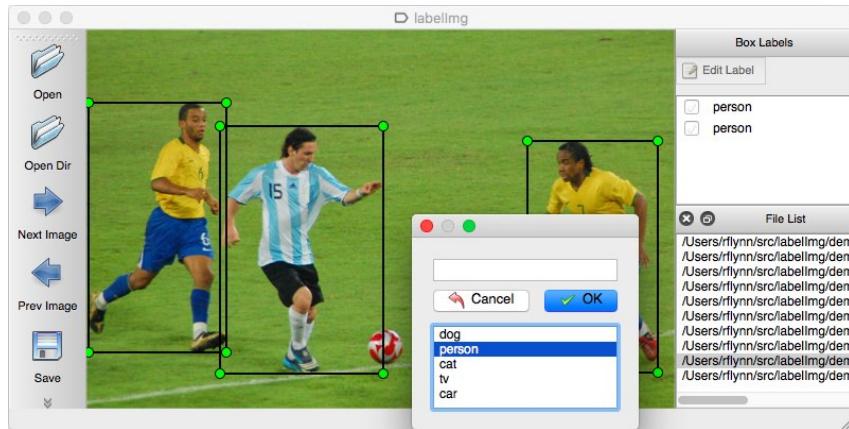


Figura 9 – Interface da ferramenta de rotulagem de dados para imagens *Label Studio*

Fonte: ([TZUTALIN, 2018](#))

Cada anotação contém as coordenadas que delimitam a caixa delimitadora do objeto, também chamada de *Bounding box*, que geralmente é retangular. Ainda, possui a classe à qual o objeto pertence.

Por fim, existem vários formatos de anotações de caixas delimitadoras. Cada formato usa sua representação específica de coordenadas de caixas delimitadoras e podem ser convertidos entre si. Os mais usuais incluem o formato Pascal VOC (Visual Object Classes), o formato COCO (Common Objects in Context) e o formato YOLO (You Only Look Once).

### 2.5.1 Formato Pascal VOC

Pascal VOC é um formato usado pelo conjunto de dados *Pascal VOC* ([EVERINGHAM et al., 2010](#)). Nesse formato, as coordenadas de uma caixa delimitadora são codificadas com quatro valores em pixels:  $x_{\text{min}}$ ,  $y_{\text{min}}$ ,  $x_{\text{max}}$ ,  $y_{\text{max}}$ . A figura 10 ilustra essa formatação.

O  $x_{\text{min}}$  e o  $y_{\text{min}}$  são as coordenadas do canto superior esquerdo da caixa delimitadora. Já o  $x_{\text{max}}$  e  $y_{\text{max}}$  são as coordenadas do canto inferior direito da caixa delimitadora.

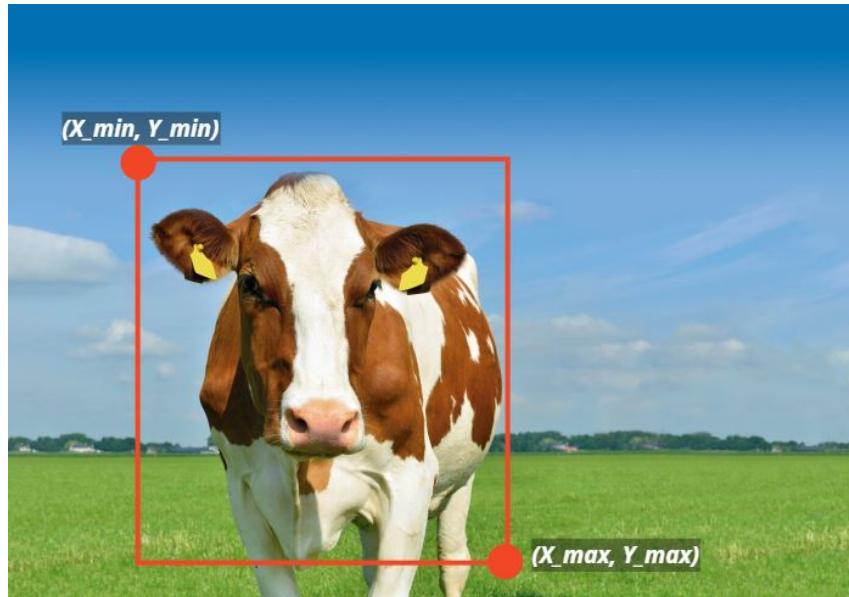


Figura 10 – Exemplo do formato de anotação *Pascal VOC*

Fonte: Autoral

### 2.5.2 Formato COCO

O formato COCO é utilizado pelo conjunto de dados *COCO common objects in context* (LIN et al., 2015). Nesse formato, a caixa delimitadora é definida por quatro valores em pixels: x\_min, y\_min, width, height. Como podemos ver na figura 11, esses valores são as coordenadas do canto superior esquerdo junto com a largura e a altura da caixa delimitadora, respectivamente.

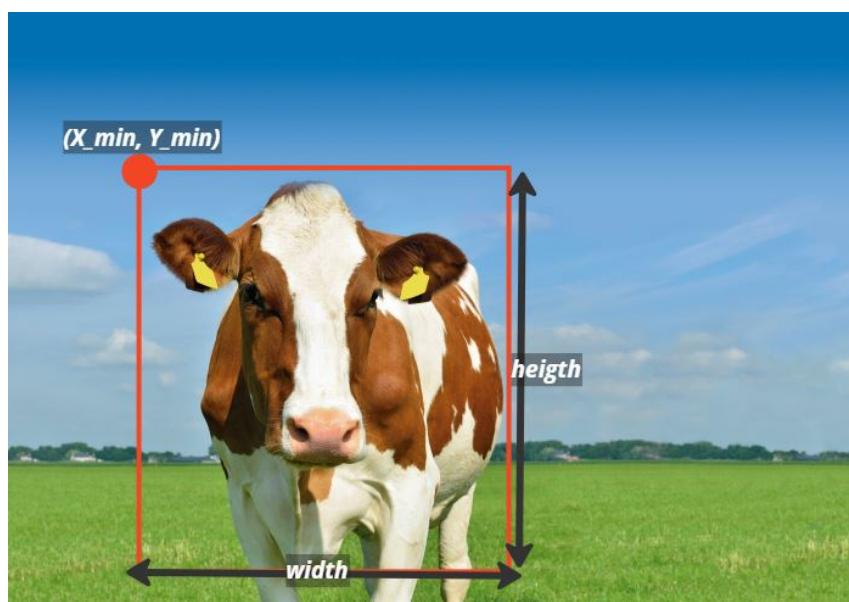


Figura 11 – Exemplo do formato de anotação *COCO*

Fonte: Autoral

### 2.5.3 Formato YOLO

O formato de anotação YOLO é constituído de quatro valores normalizados:  $x\_center$ ,  $y\_center$ ,  $width$ ,  $height$ . Onde  $x\_center$  e  $y\_center$  são as coordenadas normalizadas do centro da caixa delimitadora. Já  $width$  e  $height$  correspondem, respectivamente, à largura e à altura, normalizados, da caixa delimitadora.

## 2.6 Métricas de avaliação dos modelos

O objetivo principal para a maioria dos trabalhos de contagem por meio de detecção de objetos existentes é obter uma melhor acurácia. Ainda, por se tratar de um problema de classificação supervisionado, onde desejamos atribuir um rótulo aos dados de entrada e, também, conhecemos o rótulo real, as métricas mais apropriadas para avaliar a performance do modelo serão a acurácia, precisão, *recall* e *F1 Score* ([AGHDAM; HERAVI, 2018](#)). E por se tratar, também, de um problema de contagem, é apropriado verificar a diferença de contagem entre o valor real e o valor predito. O tempo gasto para se efetuar a contagem (TC) é analisado para se avaliar a velocidade do modelo.

### 2.6.1 A matriz de confusão

A contagem de vacas por meio de detecção consiste, primeiramente, em o agente inteligente especializado decidir se cada um dos objetos na imagem são ou não são vacas. Ou seja, trata-se de um problema de classificação binária.

A matriz de confusão é uma ferramenta que possibilita a visualização, de maneira facilitada, da performance do modelo de classificação binária. Mas também, pode ser estendida para problemas de classificação multiclasse ([TING, 2010](#)). Essa ferramenta organiza as previsões feitas por um modelo em relação aos valores reais das amostras. A figura 12 traz um exemplo de matriz de confusão para duas classes: Positivo e Negativo.

		Preditos	
		Positivo	Negativo
Real	Positivo	TP	FN
	Negativo	FP	TN

Figura 12 – Matriz de confusão

Fonte: Autoral

Como podemos ver na figura 12, a matriz de confusão é uma tabela que apresenta quatro elementos principais:

- **Verdadeiro Positivo (TP):** representa o número de observações corretamente classificadas como positivas pelo agente inteligente. No contexto deste projeto, verdadeiro positivo são as observações que o modelo detectou e rotulou como vacas que, realmente, correspondem ao animal presente nas imagens. Detecções duplicadas, mesmo realmente se tratando de vacas, não são consideradas como verdadeiro Positivo neste projeto.

- **Falso Positivo (FP):** representa o número de observações erroneamente classificadas como positivas. No contexto deste projeto, o falso positivo são as observações que o modelo detectou e rotulou vacas. Entretanto, trata-se de objetos de outra natureza, tais como: pedras, sombras, vegetação, entre outros. Ainda, neste projeto, as vacas contadas duplicadamente são consideradas falso positivo.

- **Verdadeiro Negativo (TN):** representa o número de observações corretamente classificadas como negativas. No contexto deste projeto, o verdadeiro Negativo são as observações que o modelo detectou e rotulou não vacas.

- **Falso Negativo (FN):** representa o número de observações erroneamente classificadas como negativas. No contexto deste projeto, o falso Negativo são as vacas que o modelo não detectou e rotulou, consequentemente, não contabilizou.

A partir da matriz de confusão, e esses quatro valores apresentados a cima, é possível calcular outras métricas relevantes para a avaliação da performance de um problema de classificação.

### 2.6.2 A precisão

A métrica de precisão, do inglês *precision*, calcula a fração de positivos previstos e é dado pela equação 2.1, onde  $TP$  é o número de previsões positivas corretamente e  $FP$  é o número de previsões positivas erradas. Quanto mais próximo de um, melhor é a performance do modelo. (GU SHUANG BAI, 2022)

$$precision = \frac{TP}{TP + FP} \quad (2.1)$$

No contexto deste projeto, a precisão é a métrica que identifica nas observações contabilizadas como vacas, quantas realmente são vacas.

### 2.6.3 O *recall*

A métrica *recall* calcula a fração de positivos reais e é dado pela equação 2.2, onde  $TP$  é o número de previsões positivas corretamente e  $FN$  é o número de previsões negativas erradas. Quanto mais próximo de um, melhor é a performance do modelo.(GU SHUANG BAI, 2022)

$$recall = \frac{TP}{TP + FN} \quad (2.2)$$

Ao contrário da precisão, o *recall* dá maior ênfase para os erros por falso negativo. Desse modo, no contexto deste projeto, um *recall* baixo significa que o modelo deixou de detectar e contar uma grande quantidade de vacas.

### 2.6.4 O *F1 Score*

Com as métricas de precisão e *recall* conhecidas é possível relacioná-las por meio da média harmônica e obter o *F1 Score*. Desse modo, podemos avaliar um modelo de classificação com base em uma única quantidade. O *F1 Score* é dado pela equação 2.3, onde onde  $TP$  é o número de previsões positivas corretamente,  $FP$  é o número de previsões positivas erradas e  $FN$  é o número de previsões negativas erradas. Assim como a precisão e o *recall*, quanto mais próximo de um, melhor é a performance do modelo.(GU SHUANG BAI, 2022)

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2TP}{2TP + FP + FN} \quad (2.3)$$

### 2.6.5 A acurácia

A acurácia calcula a fração de amostras que são classificadas corretamente e é dado pela equação 2.4, onde  $TP$  é o número de previsões positivas corretamente,  $TN$

é o número de previsões negativas corretamente,  $FP$  é o número de previsões positivas erradas e  $FN$  é o número de previsões negativas erradas.

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.4)$$

Assim como a precisão e o *recall*, quanto mais próximo de 1 for a acurácia, melhor é a performance do modelo de classificação. Entretanto, a acurácia alta, por si só, não é suficiente para avaliar alguns problemas de classificação. Isso porque, em casos de um desbalanceamento entre as classes no conjunto de dados de validação, pode ocorrer de o modelo classificar todas as amostras com o rótulo da classe de maior quantidade e nenhuma com o rótulo da classe de menor quantidade e, mesmo assim, o modelo apresente uma acurácia alta. Por isso, a importância de analisar a performance do modelo de classificação com a acurácia, juntamente, com o *F1 Score*(TING, 2010).

### 2.6.6 A diferença na contagem

Outra métrica de avaliação importante para o modelo de contagem desenvolvida neste projeto é a diferença na contagem (DiC). Essa métrica tem o objetivo de contar a diferença entre o número previsto e o número real. o módulo da diferença na contagem é o valor absoluto da média de DiC média em todas as imagens e é dado pela seguinte equação 2.5 onde  $Count_i$  é o número de instâncias previstas,  $count_i^*$  é o número de instâncias verdadeiras e  $N$  é o número de imagens (GU SHUANG BAI, 2022).

$$|\text{DiC}| = \frac{1}{N} \sum_i^N |count_i - count_i^*| \quad (2.5)$$

### 2.6.7 Tempo de contagem (TC)

O tempo de contagem, como nome já diz, é uma métrica, em segundos, para avaliar o tempo que o modelo de contagem gasta para realizar as detecções e apresentar as contagens.

A agilidade de detecção de uma rede neural é importante para garantir a eficiência, a responsividade e a usabilidade de sistemas de inteligência artificial em uma variedade de cenários práticos.

## 2.7 Data Augmentation

*Data augmentation* é uma técnica que gera variações artificiais dos dados de treinamento, ajudando a aumentar o volume e a diversidade dos dados disponíveis para o treinamento do modelo. Isso pode melhorar o desempenho e a capacidade de generalização

do modelo, tornando-o mais robusto e capaz de lidar com diferentes variações dos dados de entrada. ([LECUN et al., 1995](#))

Em processamento de imagens, alguns exemplos comuns de transformações aplicadas no *data augmentation* são:

- **Rotação:** girar a imagem em um determinado ângulo.
- **Espelhamento horizontal ou vertical:** refletir a imagem horizontalmente ou verticalmente.
- **Translação:** deslocar a imagem em uma determinada direção.
- **Zoom:** ampliar ou reduzir a imagem.
- **Corte aleatório:** recortar uma parte aleatória da imagem.
- **Alterações de cor:** ajustar brilho, contraste, saturação, entre outros.

Essas transformações criam novas instâncias de dados que são semelhantes às originais, mas têm algumas variações. Isso permite que o modelo de inteligência artificial aprenda a reconhecer diferentes variações dos objetos presentes nas imagens, tornando-o mais robusto e capaz de lidar com imagens de entrada que possam ter variações semelhantes.

Outra aplicação do *data augmentation* é quando existe um desequilíbrio entre as classes do conjunto de treinamento. Ao aplicar transformações aos dados de classes minoritárias, é possível gerar mais exemplos dessas classes e equilibrar a distribuição dos dados([PEREZ; WANG, 2017](#)).

## 2.8 Janelamento

Como já descrito no tópico 2.2, as primeiras camadas de uma rede neural são responsáveis por um redimensionamento dos dados de entrada. Ao se trabalhar com a detecção de objetos pequenos em imagens relativamente grandes, um problema recorrente é a perda de características do objeto de interesse logo nas primeiras camadas do um modelo de rede neural. A figura 13 e a figura 14 ilustram esse fenômeno.

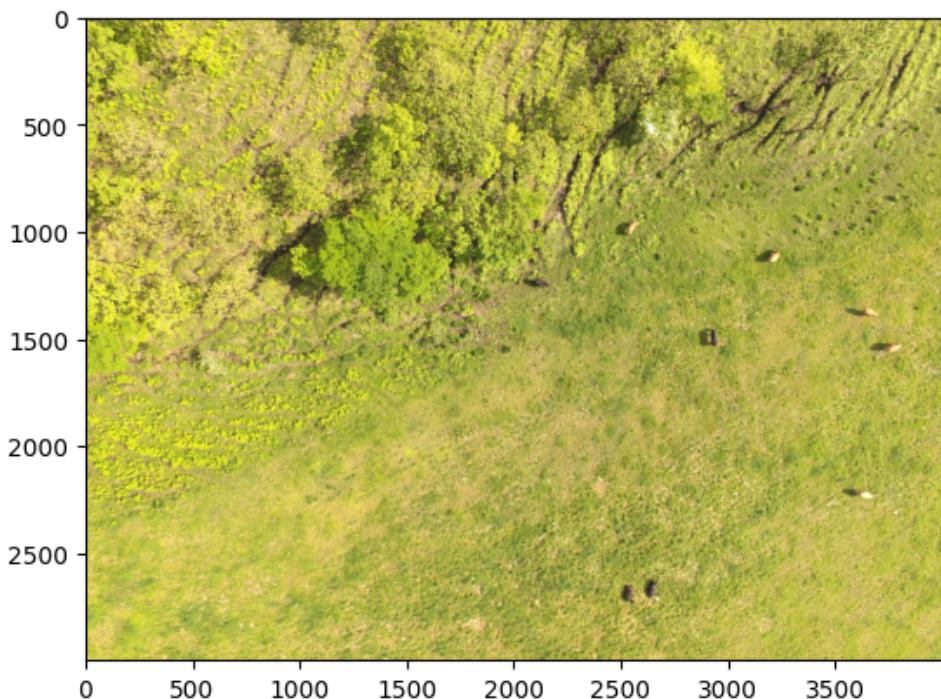


Figura 13 – Imagem aérea original de dimensões 4000x3000 pixels de vacas em uma criação extensiva

Fonte: adaptado ([SHAO REI KAWAKAMI, 2020](#))

Analizando a figura 13, percebemos que, embora muito pequenas, há a presença de nove vacas na imagem. Essas vacas ocupam apenas alguns poucos pixels em comparação com a imagem total. Ao realizar um redimensionamento para 640x640 pixels, resultou-se na figura 14. Nessa figura, diminuiu-se a diferenciação entre as vacas e qualquer outro elemento presente no ambiente, tais como: pedras, vegetação e deformação no solo.

Essa mesma dificuldade que enfrentamos para perceber as vacas na figura 14, um modelo de rede neural pode enfrentar se realizarmos transformações inadequadas. Isso porque, o objeto de interesse perdeu suas características fundamentais. Ou seja, perdeu informação.

No seu trabalho de 2022 ([KOYUN et al., 2022](#)) propõe algumas estratégias para lidar com problemas de detecção de objetos muito pequenos em relação a imagem. Uma dessas abordagem é a técnicas de janelamento.

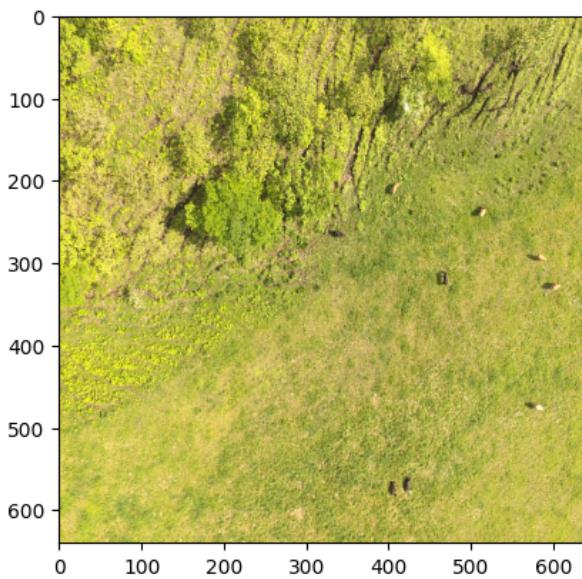


Figura 14 – Imagem aérea redimensionada para 640x640 pixels de vacas em uma criação extensiva

Fonte: adaptado ([SHAO REI KAWAKAMI, 2020](#))

O janelamento consiste em repartir a imagem em imagens de dimensões menores. Ao fazer esse pré-processamento no conjunto de dados, há um aumento do tamanho dos objetos de interesse em relação ao espaço dimensional da imagem. Dessa forma, as perdas de informações são reduzidas nas primeiras camadas de rede neural. A figura 15 ilustra esse pré-processamento.

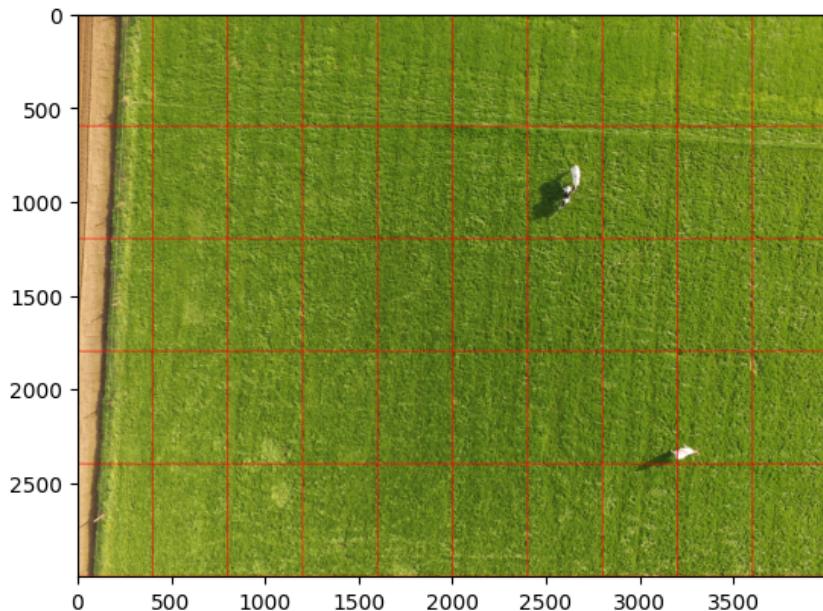


Figura 15 – Técnica de janelamento aplicado à uma imagem aérea com dimensões de 4000x3000 pixels de vacas em uma criação extensiva

Fonte: adaptado ([FRANKE; MUCHER, 2021](#))

Ainda, o dividir a imagem original em janelas menores deve-se atentar às dimensões das janelas. Isso porque a largura e altura das janelas devem ser múltiplas da largura e da altura da imagem original, respectivamente. Caso contrário, as janelas criadas não contemplariam toda a área da imagem original. A figura 16 ilustra uma escolha incorreta do tamanho das janelas. Nessa figura, foram criadas janelas de dimensão 600x600 pixels, deixando uma parte da imagem original fora da cobertura.

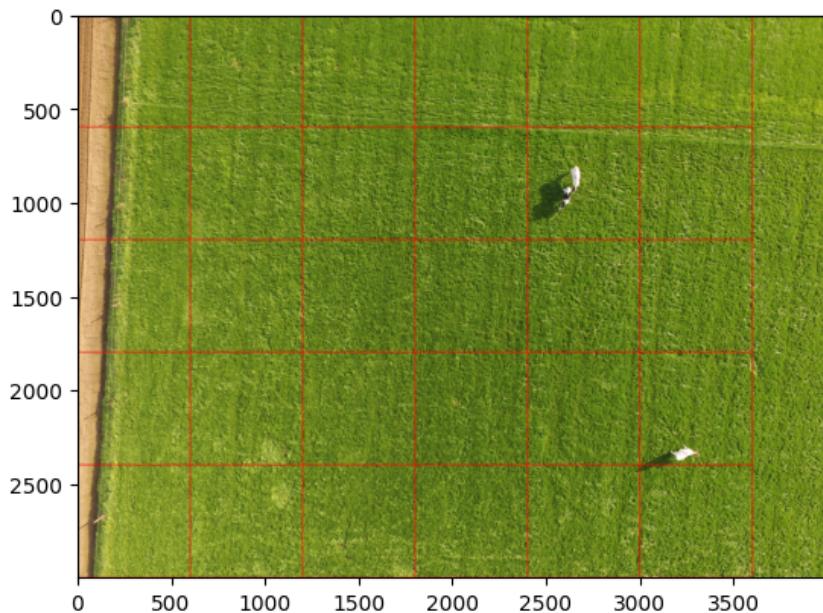


Figura 16 – Ilustração do uso inapropriado da técnica de janelamento com janelas de 600x600 pixels aplicada em uma imagem original de 4000x3000 pixels

Fonte: adaptado (FRANKE; MUCHER, 2021)

Por fim, o janelamento produz um aumento considerável no número de imagens. E como é possível notar na figura 15, a maior parte dessas imagens produzidas pelo janelamento não possuem os objetos de interesse. Consequentemente, isso resulta em um desbalanceamento no conjunto de dados. Por outro lado, existem várias técnicas para minimizar esse desbalanceamento, uma delas é o *data augmentation* descrito no tópico 2.7

## 2.9 Considerações Finais

Nessa seção foram apresentados os conhecimentos fundamentais necessários para o desenvolvimento de um projeto de visão computacional. Os conceitos foram apresentados de maneira gradual, começando pelo contexto histórico da inteligência artificial, chegando na arquitetura YoloV8, arquitetura com resultados promissores para aplicações em problemas de detecção e contagem e finalizando com a técnica de janelamento de imagens. Na próxima seção será apresentado como a arquitetura YoloV8, juntamente a técnica de

pré-processamento, possibilitará a construção do modelo de contagem de gado por meio de visão computacional neste projeto.

### 3 Trabalhos Relacionados

#### 3.1 Técnicas clássicas de processamento de imagem

No artigo ([SANTO; FILHO, 2020](#)), o autor apresenta uma solução para identificação e contagem de bovinos brancos em imagens aéreas obtidas por meio de drone. Para construir a solução são utilizados técnicas clássicas de processamento de imagens, tais como: suavização gaussiana, conversão de cores, segmentação, erosão e dilatação, e detecção de contornos. Ao final do estudo, o autor obteve uma **precisão de 89,1%** e **contou corretamente 82,6% das vacas** existente no conjunto de validação.

No artigo ([RIBEIRO; GUEDES; BARBIERI, 2019](#)), o outro se propôs desenvolver solução para contar rebanhos de gados em criação extensiva e de confinamento por meio de aplicação de filtros de processamento de imagem e a utilização de algoritmos de visão computacional. Nesse estudo, o autor obteve uma **precisão média de 73.6%**.

#### 3.2 Técnicas avançadas de aprendizado de máquina

No seu ([BARBODO et al., 2019](#)), foi realizado um estudo sobre a detecção de gado em imagens de UAV usando *Deep Learning*. Nesse trabalho as imagens aéreas de rebanhos de gado das raças Canchim e Nelore, animais de coloração clara. Ainda, foram geradas, cuidadosamente, imagens de  $224 \times 224$  pixels a partir das imagens originais de  $4864 \times 3648$  pixels, comportando totalmente o animal no interior da imagem. Com as imagens preparadas foram testados 900 modelos de detecção de diferentes a partir de 15 arquiteturas de CNN. Dentre essas arquiteturas, destacam-se as famílias VGG, MobileNet, NasNet Large e ResNet. O experimento mostrou que a maioria dos modelos foram capaz de atingir uma **precisões acima de 95%** quando os blocos de imagens de treinamento e teste foram cuidadosamente gerados para fornecer a melhor caracterização de cada classe. E apontou que na prática, usar modelos treinados dessa forma é desafiador pois os blocos de imagem não podem ser gerados adequadamente para abranger perfeitamente os animais a serem detectados. Ainda, o autor sugere como trabalhos futuros que uma possível solução para esse problema é varrer toda a imagem usando uma janela deslizante de forma que cada animal apareça, pelo menos parcialmente, em vários blocos de imagem a serem analisados pelo modelo. Essa ideia se aproxima bastante do estudo realizado neste projeto com a aplicação da técnica de janelamento.

No artigo ([SHAO REI KAWAKAMI, 2020](#)), assim como este trabalho, propôs um sistema de detecção e contagem de gado baseado em Redes Neurais Convolucionais.

nais (CNNs) utilizando imagens aéreas capturadas por um Veículo Aéreo Não Tripulado (UAV). Nesse trabalho, AUTOR realizou diversos testes redimensionando as imagens até obter uma imagem ideal determinada pelo tamanho do objeto e a taxa de *downsampling* da rede neural, tanto no treinamento quanto no teste. A rede neural utilizada nesse projeto foi a YOLOv2  $544 \times 544$ . Ao realizar esse experimento o Autor concluiu que o melhor redimensionamento foi de **768x768px** no qual obteve uma **precisão 0.957, um recall de 0.946 e uma medida F1 Score de 0.952.**

Outro trabalho que não pode deixar de ser citado é o artigo ([XU et al., 2020b](#)). Nesse trabalho o autor utilizou técnica de segmentação de instâncias para contabilizar gados em criação extensiva e também em confinamento. Para isso foram testadas as arquiteturas Faster R-CNN, Mask R-CNN, Yolo v3 e SSD. Ainda, as arquiteturas de rede neural foram treinadas para detectar o animal por completo e também para detectar apenas a cabeça do animal, a fim de comparar os resultados e concluir qual a técnica resulta em uma contagem acurada. Ao final do experimento, o autor obteve uma **acurácia de 90% para rebanhos de gado em uma criação extensiva e 94% em confinamento.**

## 4 Metodologia

A metodologia utilizada neste projeto para a construção de um sistema de contagem de vacas por meio de visão computacional está ilustrada na figura 17. O sistema recebe, como entrada, uma imagem aérea de rebanhos de gados. Essas imagens devem possuir grandes dimensões e ângulo de captura vertical, a uma altura constante e conhecida. Neste projeto usou-se imagens de 4000x3000 pixels. Em seguida, essa imagem é subdividida em janelas de dimensões inferiores, afim de preservar as características dos objetos de interesse, produzindo diversas imagens menores que juntas cobrem toda a área da imagem original. O Módulo de detecção é constituído por uma rede neural convolucional treinada para detectar gados. Esse módulo, então, recebe cada uma das janelas e realiza a detecção dos objetos de interesse presentes em cada imagem, retornando dois artefatos: o valor acumulado do número vacas detectadas nas janelas e uma tabela com as coordenadas dessas detecções. Por fim, os valores produzidos pelo módulo de detecção são inseridos na imagem original, retornando para o usuário a quantidade total de gados e suas respectivas localizações.

### Modelo de Contagem

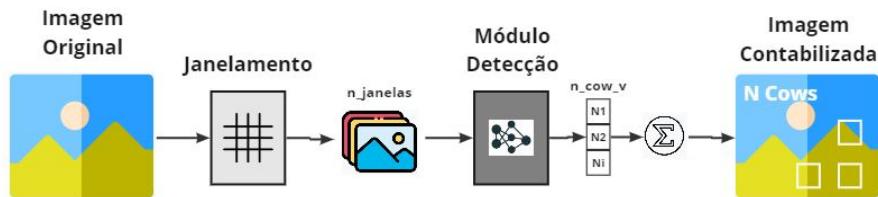


Figura 17 – Arquitetura do modelo de contagem por janelamento.

Fonte: Autoral

Para se chegar no modelo de contagem ilustrado na figura 17 o fluxo de trabalho desse projeto foi dividido em duas partes. A primeira parte refere-se à construção e comparação de dois modelos de contagem construídos a partir de dois tamanhos diferentes de janelas. A segunda parte consiste na aplicação de técnicas de otimização no modelo de contagem que obteve a melhor performance na primeira parte, afim de se obter um modelo de contagem final.

## 4.1 Parte I: A escolha das janelas

### 4.1.1 Fase 1: Coleta e preparação dos dados

A construção de qualquer modelo de *deep learning* começa pela coleta dos dados. Sem a disponibilidade dos dados adequados, não é possível desenvolver nenhuma arquitetura de *deep learning*. Isso porque os dados são utilizados para treinar, testar e validar o modelo.

Desse modo, para desenvolver a modelagem de maneira satisfatória é necessário coletar um volume razoável de imagens de bovinos vistos de um ângulo superior a uma altura constante e conhecida. Além do ângulo específico, é necessário que as imagens possuam uma alta resolução e, para ter uma performance satisfatória em um ambiente descontrolado, uma certa interferência de ruídos. Ruídos neste contexto pode ser entendido como tudo aquilo que venha atrapalhar ou confundir o agente inteligente, como: luminosidade, sombras e obstáculos.

Com as imagens coletadas, é necessário identificar as coordenadas dos objetos de interesses em cada imagem. Essas coordenadas são insumos para a construção das caixas delimitadoras, também chamadas de *bounding box*. Em seguida, cria-se os arquivos de anotação para cada uma das imagens. Esses arquivos são cruciais para o treinamento supervisionado e sua função é de armazenar as informações de localização e classe de cada objeto presente na imagem.

Por fim, com as imagens e suas respectivas anotações prontas, é necessário selecionar uma parte das imagens originais para compor o conjunto de dados de validação. Esses dados devem ser isolados de todo o processo de construção do modelo de contagem, afim de evitar qualquer imputação de viés.

A figura 18 resume, por meio de um diagrama de blocos, o fluxo de trabalho realizado neste projeto para a coleta e preparação da imagens.

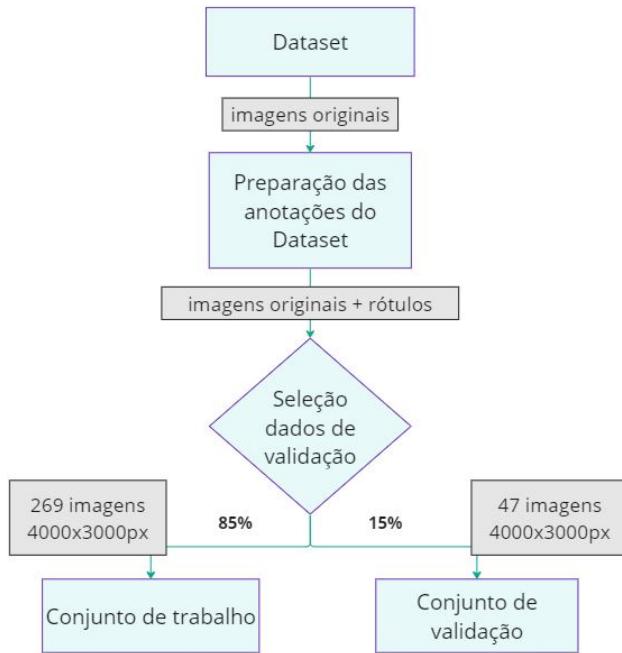


Figura 18 – Fluxograma da coleta e preparação dos dados.

Fonte: Autoral

#### 4.1.1.1 Datasets

Para este projeto, as imagens utilizadas para o desenvolvimento do modelo são provenientes do *Cattle detection and counting in UAV images based on convolutional neural networks* ([SHAO REI KAWAKAMI, 2020](#)). Ele é formado por 316 imagens de alta resolução, com dimensão de 4000x3000 pixels, capturadas por meio de um drone modelo *DJI Phantom 4* mantido a uma altura de 50 metros resultando em uma área de cobertura de 60 x 80 metros ([SHAO REI KAWAKAMI, 2020](#)). As imagens mostram, em uma visão aérea, rebanhos de gado de diversas colorações em um ambiente descontrolado, com a presença de árvores, sombras, pedras, deformidade no solo e outros objetos. Por fim, o *dataset* não possui os arquivos de anotações das imagens. Desse modo, houve a necessidade de criar as anotações manualmente. A figura 19 traz um exemplo de imagem presente nesse *dataset*. Já a tabela 1 resume as principais informações desse conjunto de imagens.

Tabela 1 – Características do *dataset* original

Número de imagens	316
Dimensão das imagens	4000x3000px
Anotação	não
Drone	DJI Phantom 4
Altura de voo	50 m
Coloração do gado	Diversa
Presença de ruídos	Sim



Figura 19 – Exemplo de imagem presente no *dataset Cattle detection and counting in UAV images based on convolutional neural networks*.

Fonte: ([SHAO REI KAWAKAMI, 2020](#))

#### 4.1.1.2 Preparação das anotações dos conjuntos de dados

Como dito no tópico 2.5, os arquivos de anotação são cruciais para o treinamento supervisionado. Então, antes de iniciarmos o treinamento do modelo de detecção é necessário que todas as anotações de cada imagem estejam devidamente criadas.

Para a criação das anotações das imagens do *dataset* foi utilizado uma ferramenta de rotulagem de dados para imagens de código aberto chamada *Label Studio*, desenvolvida por *Tzutalin* e descrita no tópico 2.5.

Ao final do processo de rotulagem, foram produzidos 316 arquivos *.XML* com as anotações para as imagens do *dataset Cattle detection and counting in UAV images based on convolutional neural networks* ([SHAO REI KAWAKAMI, 2020](#)) seguindo o formato *Pascal VOC*, descrito no tópico 2.5.1. A figura 20 apresenta um dos arquivos de anotação resultante desse processo.

Em seguida, como utilizamos o modelo de arquitetura de rede neural convolucional *YoloV8 Nano* para este projeto, precisou-se converter o formato das anotações de *Pascal VOC* para *Yolo*. A figura 21 traz um exemplo do arquivo de anotação de acordo com esse novo formato, onde o primeiro elemento refere-se a classe do objeto e os outros quatro

```

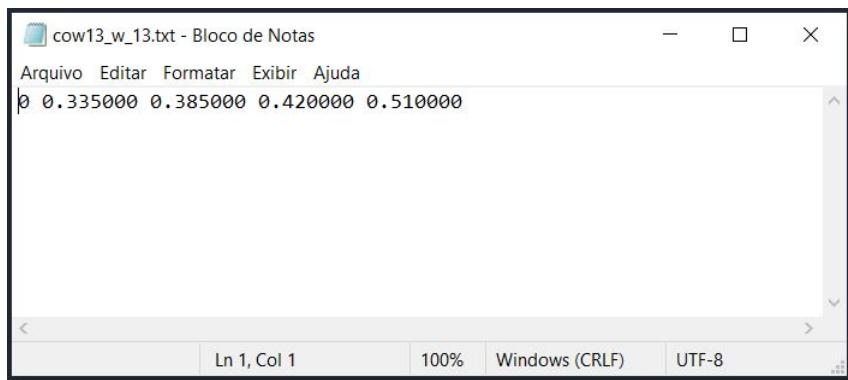
cow1.xml*  X
└─<annotation>
    └─<folder>cow</folder>
    └─<filename>cow1.JPG</filename>
    └─<path>C:\ObjectDetection\cow\cow1.JPG</path>
    └─<source>
        <database>Unknown</database>
    </source>
    └─<size>
        <width>4000</width>
        <height>3000</height>
        <depth>3</depth>
    </size>
    └─<segmented>0</segmented>
    └─<object>
        <name>cow</name>
        <pose>Unspecified</pose>
        <truncated>1</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>151</xmin>
            <ymin>1</ymin>
            <xmax>260</xmax>
            <ymax>93</ymax>
        </bndbox>
    </object>
</annotation>

```

Figura 20 – Exemplo de arquivo de anotação no formato *Pascal VOC*

Fonte: Autoral

elementos são, respectivamente, as coordenada do centro da caixa delimitadora, a largura e a altura da caixa delimitadora, todos normalizados.

Figura 21 – Exemplo de arquivo de anotação no formato *Yolo*

Fonte: Autoral

#### 4.1.1.3 Seleção do conjunto de dados de validação

Como visto na figura 18, uma parte dos dados são utilizados para validação. Esses dados são pré selecionados, exclusivamente, para validar a performance do modelo de

inteligencia artificial com dados inéditos. As métricas provenientes dessa validação se aproximam, com fidelidade, com os resultados obtidos do modelo em um uso real. Para isso, essas imagens de validação não podem ser incluídas durante a fase de treinamento do modelo, afim de evitar o surgimento de algum tipo viés do agente inteligente.

Portanto, foram pré selecionados 47 imagens do *dataset* original, juntamente com suas respectivas anotações, para compor o conjunto dados de validação. Essa imagens selecionadas representam, aproximadamente, 15% do *dataset* original e foram selecionadas aleatoriamente, também com o intuito de evitar a inclusão de qualquer viés humano e representar a diversidade do conjunto de imagens original. Com o conjunto de validação formado, o restante das imagens compõem o conjunto de trabalho. Esse conjunto é formado por 269 imagens que foram utilizado para realizar o treinamento do modelo de detecção.

#### 4.1.2 Fase 2: Aplicação do janelamento e a construção do conjunto de dados 1 e 2

Com o conjunto de imagens de trabalho formado, aplica-se o janelamento afim de se obter dois conjuntos de dados com diferentes dimensões de janelas. A figura 22 resume, por meio de diagrama de blocos, o fluxo de trabalho realizado na aplicação da técnica janelamento.

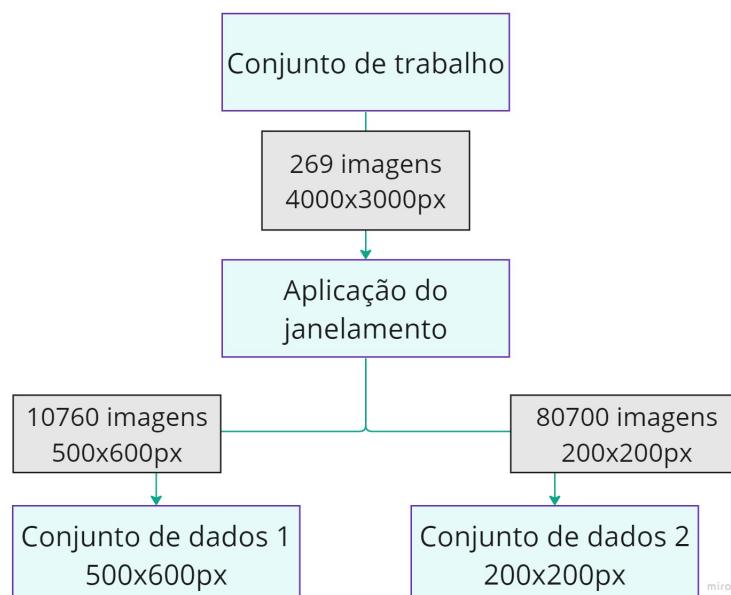


Figura 22 – Fluxograma da aplicação do janelamento para a construção do conjunto de dados 1 e 2.

Fonte: Autoral

#### 4.1.2.1 Aplicação do janelamento

Como descrito no tópico 2.7, detectar e contar objetos pequenos em imagens, relativamente, grandes é uma tarefa desafiadora. Isso porque há uma perda de informação e, consequentemente, características do objeto de interesse causado pelo redimensionamento da imagem logo nas primeiras camadas de uma rede neural.

Para minimizar esse fenômeno, usou-se neste projeto a técnica de janelamento das imagens, produzindo múltiplas imagens menores que juntas compõem a imagem original.

A escolha do tamanho ideal da janela pode ser um desafio, pois depende do tamanho do objeto de interesse e da resolução da imagem. Janelas muito pequenas podem levar a perda de informações relevantes, aumentam a probabilidade do objeto de interesse ficar localizado entre múltiplas janelas, fazendo com que o modelo detecte e conte o objeto mais de uma vez. Ainda, quanto menor o tamanho das janelas, maior é o número de janelas necessárias para cobrir a área da imagem original. Consequentemente, isso reduz a velocidade do processo de contagem. Isso porque, o processamento repetido de várias janelas através da rede neural pode aumentar significativamente o tempo de inferência e tornar a abordagem mais lenta (LEE JINSU E BANG, 2017).

Já janelas muito grandes podem resultar em um alto custo computacional e na detecção incorreta de objetos menores. Isso porque, quanto maior a janela, menos características do objeto de interesse são preservadas nas camadas de *downsample* de uma rede neural, dificultando a aprendizagem do modelo de detecção.

Ainda, outro fator relevante para a escolha das janelas é certificar que a largura e a altura das janelas são divisores da largura e a altura da imagem original, respectivamente. Como descrito no tópico 2.7, essa condição é necessária para garantir a cobertura total da imagem original. Desse modo, a largura das janelas utilizadas nesse projeto devem ser divisores de 4000 e a altura das janelas devem ser divisores de 3000. A tabela 2 traz todos os valores divisores de 4000. Já a tabela 3 traz todos os valores divisores de 3000.

Tabela 2 – Valores divisores do número 4000

1	2	4	5	8	10	16	20
25	32	40	50	80	100	125	160
200	250	400	500	800	1000	2000	4000

Tabela 3 – Valores divisores do número 3000

1	2	3	4	5	6	8	10
12	15	20	24	25	30	40	50
60	75	100	120	125	150	200	250
300	375	500	600	750	1000	1500	3000

Como vimos, a escolha do tamanho das janelas possui vantagens e desvantagens.

Desse modo, para garantir a melhor performance do modelo de contagem final foram testados duas possibilidades de janelas. Pra isso, foram construídos dois conjunto de dados a partir do conjunto de trabalho, o conjunto de imagens restantes após a seleção do conjunto de validação descrita no tópico 4.1.1.2.

#### 4.1.2.2 Conjunto de dados 1

O conjunto de dados 1 tem como objetivo testar a performance do modelo de contagem utilizando janelas com dimensões próximas aos da primeira camada da arquitetura da rede neural. A arquitetura de rede neural convolucional utilizada neste projeto é a YoloV8 nano, cuja a dimensão é 640x640 pixels. Outras características dessa arquitetura está descrito no tópico 4.1.4.1.1.

Considerando a dimensão da YoloV8 nano utilizada neste projeto e consultando as tabelas 2 e 3, conclui-se que 500x600 pixels são uma escolha interessantes para a dimensão das janelas para esse primeiro conjunto de dados.

O janelamento de 500x600 pixels aplicados no conjunto de 269 imagens de 4000x3000 pixels resultou na criação de 10760 imagens. Dessas imagens criadas, 9498 imagens não possuem a presença do objeto de interesse. Isso representa mais de 88% das imagens criadas. Ainda, das 1262 imagens onde há a presença de vacas apenas 829 possuem ao menos um animal completamente inserido.

#### 4.1.2.3 Conjunto de dados 2

O conjunto de dados 2 tem como objetivo testar a performance do modelo de contagem utilizando janelas relativamente pequenas em comparação com o conjunto de dados 1. Dessa forma, o conjunto de dados 2 foi construído aplicando janelas de 200x200 pixels em cada imagem que restaram após o processo descrito no tópico 4.1.1.2.

O janelamento de 200x200 pixels aplicados no conjunto de 269 imagens de 4000x3000 pixels resultou na criação de 80700 imagens. Dessas imagens criadas, 78014 imagens não possuem a presença do objeto de interesse, representando mais de 96% das imagens criadas. Ainda, das 2686 imagens onde há alguma presença de vaca, apenas 643 possuem ao menos um vaca totalmente inserida.

### 4.1.3 Fase 3: Pré-processamentos e divisão das imagens em conjuntos de treino e teste

Como visto nos tópico 4.1.2.1, a construção do conjunto de dados 1 e do conjunto de dados 2 por meio do janelamento, resultou em um grande desbalanceamento dos dados. Onde a maior parte das imagens criadas não possuem a presença do objeto de interesse. Contendo, apenas, vegetação, solo e outros objetos, exceto vacas. Dessa forma,

antes de iniciar o treinamento do modelo de detecção, foi necessário efetuar alguns pré-processamentos nos conjuntos de imagens. A figura 25 resume, por meio de diagrama de blocos, o fluxo de trabalho realizado na preparação dos dados de treino e teste para os conjuntos de dados 1 e 2 descrito a seguir.

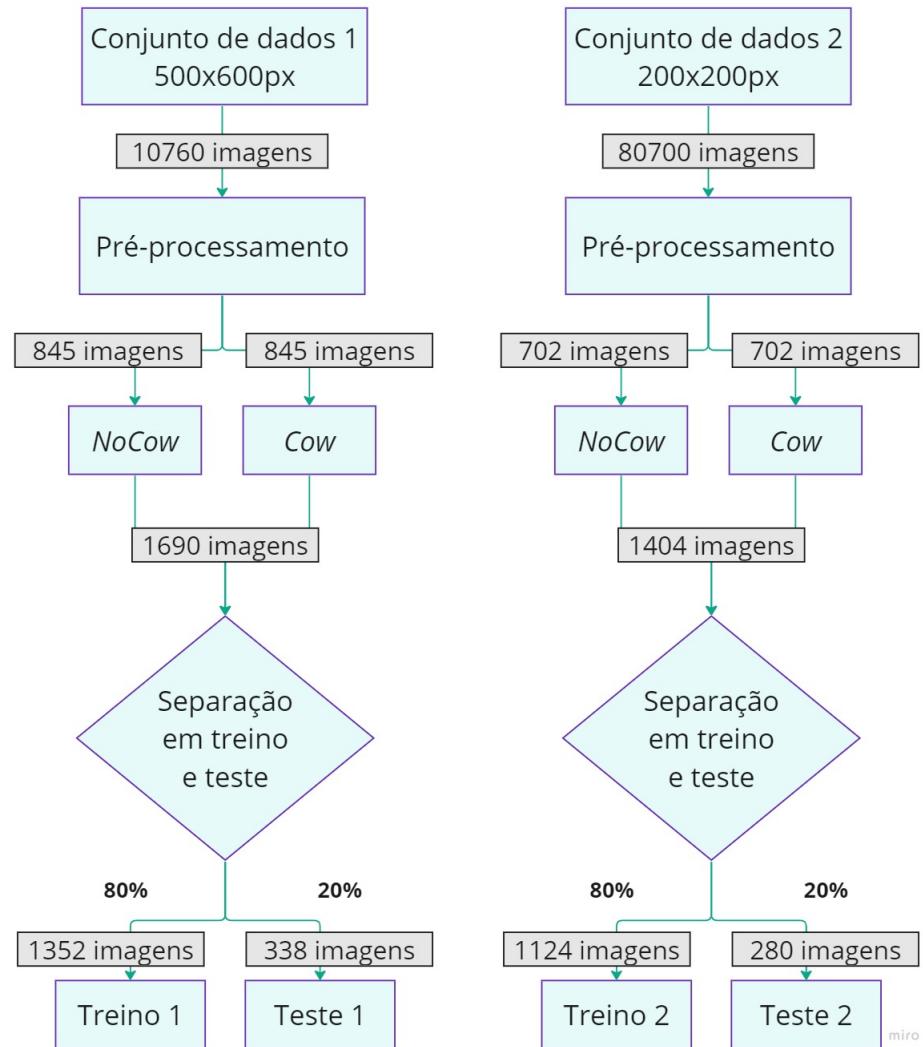


Figura 23 – Fluxograma do pré-processamento e divisão em treino e teste para os conjuntos de dados 1 e 2.

Fonte: Autoral

#### 4.1.3.1 Pré-processamentos

Primeiramente, para cada conjunto de dados, criou-se duas classes de imagens: "*NoCow*", onde não há presença de vacas, e "*Cow*", onde há qualquer intersecção entre a janela e algum *bounding box* do objeto de interesse. Cada classe de imagens exigiu um processamento específico.

Com o objetivo de selecionar apenas as imagens onde as características do objeto de interesse foram, consideravelmente, preservadas. O pré-processamento para as imagens *Cow*, iniciou-se selecionando apenas as imagens onde ao menos uma vaca possui a nova área do *bounding box* superior a 95% da área total do *bounding box* na imagem original antes do janelamento. A figura 24 ilustra essa situação. Nessa figura, a janela 1 seria descartada, sendo utilizado para o treinamento apenas a janela 2, que possui a maior parte das características da vaca.



Figura 24 – Imagem de uma vaca dividida em duas janelas de 200x200 pixels

Fonte: adaptado ([SHAO REI KAWAKAMI, 2020](#))

Com essas imagens selecionadas, em seguida, houve uma filtragem no conteúdo das anotações dessas imagens. Essa filtragem consistiu em excluir, completamente, as informações das vacas com a nova área do *bounding box* inferior a 70% da área total do *bounding box* na imagem original antes do janelamento. Isso tem o objetivo de evidenciar os objetos que tiveram a maior parte das suas características preservadas para o treinamento do modelo de detecção e ajudar o modelo a contabilizar a vaca apenas uma vez caso ela esteja dividida em múltiplas janelas.

O pré processamento das imagens *Cow* descrito foi realizado nos conjuntos de dados 1 e conjunto de dados 2. A tabela 4 mostra a quantidade de imagens *Cow* no início e no final do pré processamento para os dois conjuntos de dados.

#### 4.1.3.2 Construção dos conjuntos de dados de treino e teste

Ainda na fase de preparação dos dados, dividimos as imagens em dois conjuntos: treino e teste. Para isso, primeiramente, dividiu-se o conjunto de imagens da classe "*Cow*",

Tabela 4 – Quantidade de imagens *Cow* antes e depois do pré processamento em cada conjunto de dados

Dataset	Inicio	Final
Conjunto de dados 1 (500x600)	1262	845
Conjunto de dados 2 (200x200)	2686	702

resultante do processo anterior, de maneira aleatória, respeitando a seguinte proporção: 20% das imagens foram selecionadas para compor o conjunto de dados de teste e 80% das imagens foram selecionadas para compor o conjunto de dados de treino.

Em seguida, afim de evitar um grande desbalanceamento dos conjuntos de dados entre as classes "*Cow*" e "*NoCow*", foram selecionadas, também de maneira aleatória, o mesmo número de imagens de janelas da classe "*NoCow*" para compor o conjunto de dados de treino e de teste. A tabela 5 e a tabela 6 traz de maneira resumida a composição dos conjuntos de dados de treino e teste ao final dessa divisão para o conjunto de dados 1 e 2.

Tabela 5 – Composição do conjuntos de dados de treino e teste para o conjuntos de dados 1

Dataset	Cow	NoCow	Total
Treino	663	689	1352
Teste	182	156	338

Tabela 6 – Composição do conjuntos de dados de treino e teste para o conjuntos de dados 2

Dataset	Cow	NoCow	Total
Treino	550	574	1124
Teste	152	128	280

Para a realização de todas as etapas do pré-processamento e as manipulações das imagens e das anotações citadas foram utilizadas, como ferramentas, a linguagem de programação Python 3.6 distribuído via anaconda3, juntamente, com os módulos OpenCV 4.7.0.72 e pandas 2.0.2.

#### 4.1.4 Fase 4: Construção dos modelos de detecção e contagem

Com os conjuntos de dados treino e teste, com suas respectivas anotações no formato *Yolo*, construídos adequadamente, pôde-se dar inicio a fase de treinamento do modelo de detecção.

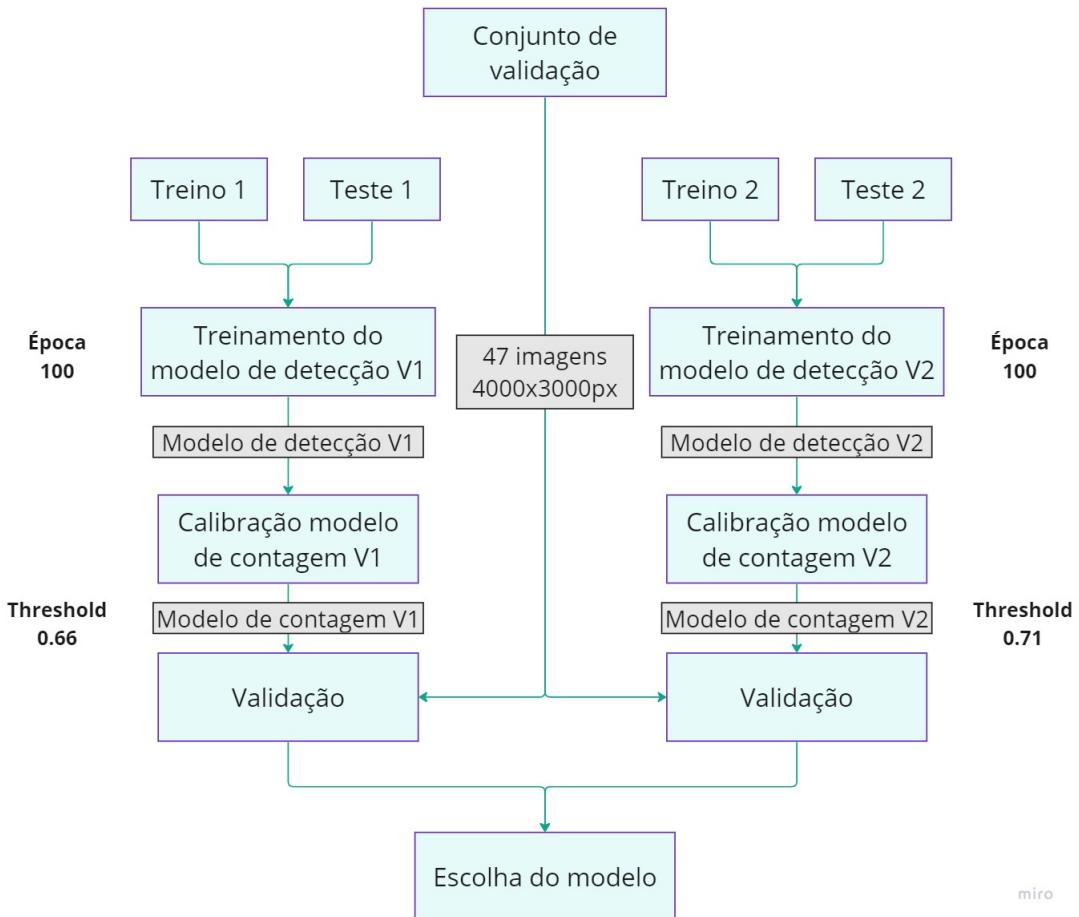


Figura 25 – Fluxograma do pré-processamento e divisão em treino e teste para os conjuntos de dados 1 e 2.

Fonte: Autoral

#### 4.1.4.1 O Modelo de Detecção:

##### 4.1.4.1.1 O processo de treinamento do modelo de detecção

Para o treinamento do modelo de detecção utilizou-se a *YoloV8 Nano*, modelo de arquitetura de rede neural convolucional criada e distribuída pela empresa *Ultralytics*. Além de ser uma arquitetura de alta performance e de fácil manipulação, a *YoloV8 Nano* é um modelo de detecção de objetos pré-treinado. Dessa forma, é possível realizar um treinamento para novas classes por meio do processo conhecido como *transfer learning*. Nesse processo, os pesos são herdados do modelo pré-treinado em seguida são ajustados para a nova tarefa. Dessa forma, há uma economia de tempo e recursos, melhor capacidade de generalização e têm uma tendência menor de *overfitting* (MA et al., 2021).

Ainda, foram selecionados os principais hiperparâmetros capazes de impactar a capacidade de aprendizagem da rede neural durante o treinamento. A tabela 7 resume as configurações utilizadas no treinamento do modelo de detecção V1 e V2.

Tabela 7 – Parâmetros de treinamento do modelo de detecção V1 e V2

Modelo	Épocas	Batch Size	LR	Otimizador	Pré-teino
Yolo V8 nano	100	16	0.01	SGD	COCO

**O número de épocas:** é a quantidade de vezes que os dados são processados pela rede neural para ajustar os pesos e melhorar a acurácia do modelo. O equilíbrio é essencial, evitando um excesso de épocas que pode levar ao *overfitting* ou poucas épocas que resultariam em *underfitting*, prejudicando a capacidade da rede de capturar padrões complexos nos dados.

**O tamanho do lote(*batch size*):** é o número de amostras processadas a cada iteração. *Batches* grandes demandam mais memória e tempo, aumentando a possibilidade de *overfitting*, enquanto *batches* pequenos economizam recursos, mas podem afetar negativamente o aprendizado do modelo. Encontrar o tamanho de lote adequado é crucial para otimizar o treinamento da rede neural.

**A taxa de aprendizado, ou *learning rate (LR)*:** tem um papel crucial no treinamento da rede neural, pois determina o tamanho da atualização dos pesos ao final de cada época. Se a taxa de aprendizado for muito alta, o desempenho do modelo pode ser prejudicado, pois ele pode não ser capaz de alcançar pontos de mínimo na função de perda. Por outro lado, uma taxa de aprendizado muito baixa exigirá mais épocas para o modelo aprender, podendo levar mais tempo para a convergência. Além disso, um valor muito baixo pode fazer com que o modelo tenha dificuldades em escapar de mínimos locais na função de perda. Encontrar a taxa de aprendizado adequada é crucial para o sucesso do treinamento e a convergência eficiente da rede neural.

**O otimizador:** é um algoritmo utilizado para atualizar os pesos durante o treinamento da rede neural, com o propósito de minimizar a função de perda.

Todo esse processo de treinamento foi realizado por meio de uma máquina composta de um processador **Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz, 16 Gb de memória Ram e GPU RTX 2060**. As GPU(*Graphics Processing Unit*) são utilizadas no treinamento de modelos de *Deep Learning* pois aceleram o processo de treinamento através do paralelismo que permite realizar mais cálculos simultaneamente do que uma CPU tradicional ([MADIAJAGAN; RAJ, 2019](#)). Ainda, usou-se o Python 3.6 distribuído via Anaconda3 como linguagem de programação.

#### 4.1.4.1.2 Processo de validação do modelo de detecção

Para avaliação do modelo de detecção, por se tratar de um problema de detecção supervisionado, onde desejamos localizar e atribuir um rótulo aos dados de entrada e, também, conhecemos o rótulo real, as métricas mais apropriadas para avaliar a perfor-

mance do modelo são a *acurácia*, *precisão*, *recall* e *F1 score* (AGHDAM; HERAVI, 2018). Essas métricas de avaliação do modelo permitem identificar se a performance do modelo resultou em *underfitting*, *overfitting* ou se já atingiu uma performance satisfatória. As métricas de avaliação estão descritas na seção 2.6.

#### 4.1.4.2 Modelo de Contagem

O modelo de contagem é responsável por realizar todo o processo descrito no primeiro paragrafo do tópico 4, representado pela figura 17. Ele recebe uma imagem, cria as janelas, realiza as detecções e retorna para o usuário a imagem com a quantidade total de vacas e suas respectivas localização.

Ainda, o modelo de contagem possui um hiperparâmetro que podemos ajustar com o objetivo de obter uma melhor performance na contagem das vacas. Esse parâmetro, chamado de *threshold*, consiste em um limiar que determina um valor mínimo de precisão que a predição que o modelo de detecção dever ter ao detectar um objeto. Um *threshold* muito baixo resulta em um modelo de detecção que considera objetos da classe *NoCow* como *Cow* e também impacta na contagem duplicada das vacas presentes em múltiplas janelas, assim como ilustrado na figura 24. Já um *threshold* muito alto, dificulta a detecção de objetos semi oclusos e a detecção em imagens pouco nítidas.

Para garantir que *threshold* do modelo de contagem foi escolhido corretamente, foi realizado uma calibração de hiperparâmetro descrita no tópico a seguir.

##### 4.1.4.2.1 Calibração do hiperparâmetro do modelo de contagem

O objetivo da calibração do hiperparâmetro do modelo de contagem é selecionar o valor *threshold* que obtenha uma melhor performance na contagem de um grupo diverso de imagens de teste.

Para isso, foram **selecionadas aleatoriamente 25 imagens para o conjunto de calibração**, com o número de vacas conhecidos, do conjunto de trabalho, imagens resultantes do processo descrito no tópico 4.1.1.2. Em seguida, foi selecionada uma lista de valores de ***threshold* variando de 0.45 até 1.0 ao passo de 0.01**.

Por fim, foram realizadas detecções para cada valor de *threshold* em cada uma das 25 imagens de teste, **comparando-se a quantidade real de vacas com a quantidade prevista pelo modelo**. O processo inteiro gerou um total de 1001 interações e uma tabela com a quantidade real e a quantidade prevista de vacas para cada imagem com cada valor de *threshold* foi construída.

Em seguida, selecionou-se dessa tabela apenas os registros onde a quantidade prevista era igual a quantidade real de vacas. Também, ordenou-se esses registros de forma descendente priorizando a combinação com o maior valor de *threshold*. Essa prioridade de

ordenação tem os seguintes objetivos: um valor de *threshold* alto resulta em um modelo de detecção mais preciso e, consequentemente, mais confiável.

#### 4.1.4.2.2 Validação do modelo de contagem

A validação do módulo de contagem é equivalente à validação da solução completa. Para isso, utilizamos um conjunto de **47 imagens pré-selecionadas**, conforme mencionado no tópico [4.1.1.2](#). Durante o processo de validação, aplicamos o modelo construído, utilizando os valores de ***threshold* obtidos na fase de calibração**, nas 47 imagens de validação. Calculamos diversas métricas, como **precisão**, **recall**, **F1 score**, **acurácia**, **erro quadrático médio (MSE)**, **erro absoluto médio (MAE)**, **a diferença de contagem (DiC)** e **tempo de contagem (TC)** para cada imagem de validação do conjunto de dados. A partir desses valores, também calculamos os valores totais e médios dessas métricas.

Consideramos o modelo de contagem satisfatório quando as métricas F1 e acurácia se aproximam de 1, enquanto as métricas MSE e MAE se aproximam de zero. Os resultados obtidos serão apresentados na próxima seção.

## 4.2 Parte II - Otimização do modelo

A baixa performance do modelo de contagem, baseado em redes neurais, pode ter origem em diversos fatores, tais como: volume insuficiente de dados de treino, imagens de treino inadequadas, número de épocas inapropriadas, má configuração dos hiperparâmetros, entre outros. As métricas de avaliação do modelo, juntamente, com a inspeção visual dos resultados obtidos dão indícios das causas que estão afetando negativamente a performance do modelo (AGHDAM; HERAVI, 2018).

Em um cenário de *overfitting* pode-se utilizar a estratégia de reduzir o número de épocas durante o treinamento do modelo, reconfigurar os hiperparâmetros do modelo e adicionar um maior e diverso volume de imagens de treino.

Já em um cenário de *underfitting*, pode-se utilizar a estratégia de aumentar o número de épocas durante o treinamento do modelo. Ainda, pode-se adicionar um maior volume de imagens de treino. E se após a inspeção visual dos resultados foi constatado que não houve a detecção dos objetos de interesse devido a presença de ruídos, tais como: sombra, luminosidade, obstáculos, entre outros, serão adicionadas imagens mais ruidosas às imagens de treinos.

Neste projeto, limitou-se a otimização no conjunto de dados e no treinamento do modelo de detecção. A figura 26 resume o fluxo de trabalho no processo de otimização do conjunto de dado que será descrito a seguir.

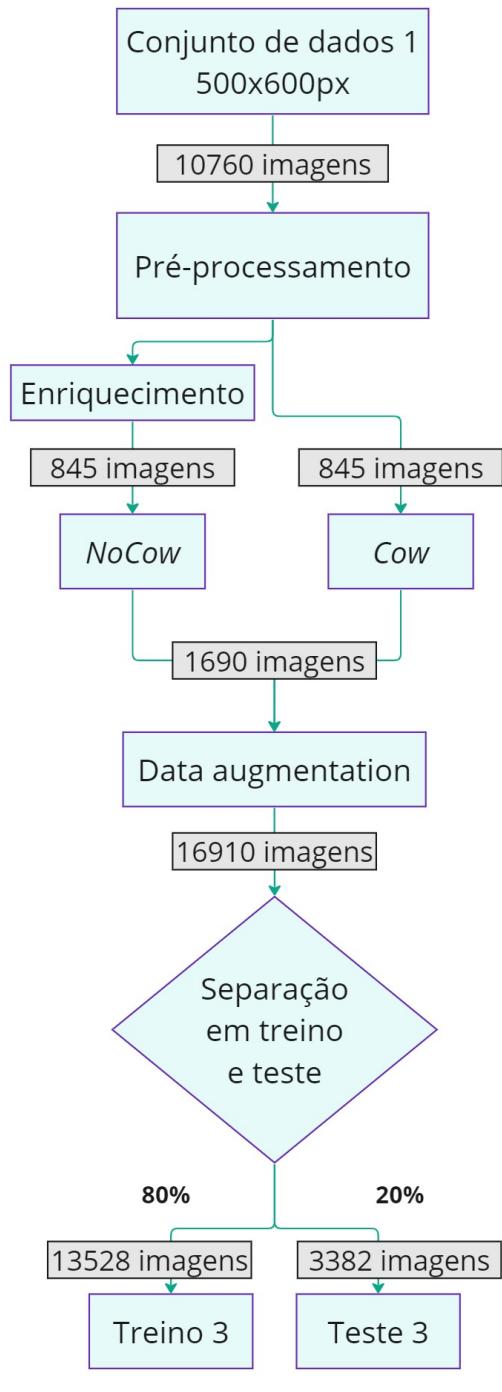


Figura 26 – Fluxograma do processo de otimização do conjunto de dados

Fonte: autoral

#### 4.2.1 Criação do conjunto de dados 3

Como veremos no tópico 5, o modelo de contagem que obteve os melhores resultados na etapa de validação, com dados inéditos, foi o **modelo de contagem V1**. Este modelo, como vimos, foi construído a partir de um janelamento de dimensão 500x600 pixels. Desse forma, criou-se um terceiro conjunto de dados, mais volumoso e diverso, a partir das imagens do conjunto de dados 1.

A etapa de pré-processamento para a criação desse novo conjunto de dados se manteve igual ao descrito no tópico 4.1.3.1, resultando em duas classes: *Cow* e *NoCow*. Entretanto, para as imagens da classe *NoCow* houve uma adição intencional de imagens mais ruidosas, com a presença de instâncias que confundiram os modelos de contagem v1 e v2. Esse processo de seleção das imagens da classe *NoCow* é chamado de enriquecimento. A figura 27 ilustra algumas imagens que foram adicionadas ao novo conjunto de imagens.



Figura 27 – Imagens da classe *NoCow* que podem confundir o modelo

Fonte: adaptado ([SHAO REI KAWAKAMI, 2020](#))

Ao final da fase de enriquecimento, é obtido um volume de 1690 imagens de 500x600 pixels, assim como no conjunto de dados 1. Desse modo, para aumentar o volume e a variabilidade dos dados efetuou-se um *data augmentation*. Essa técnica consiste em aplicar transformações no próprio conjunto de dados. Transformação de nível de pixel não alteram as anotações. Já transformação de nível espacial exigem modificações nas anotações.

**Transformação de nível espacial** tem como vantagens, além de aumentar o volume do conjunto de dados, aumentar a variabilidade das imagens e regularizar os dados. Isso permite que o modelo de detecção aprenda padrões mais robustos e melhore sua capacidade de generalização em diferentes cenários e orientações. Seguem alguns exemplos de transformação de nível espacial:

- **Espelhamento (Flip)**: inverte a imagem verticalmente ou horizontalmente, gerando versões espelhadas da imagem original.

- **Translação (*Translate*)**: deslocamento horizontal e/ou vertical aos pixels da imagem alterando a posição dos pixels em relação à sua localização original. Neste projeto, se a imagem for deslocada para a direita, os pixels à esquerda serão preenchidos com valores interpolados.
- **Rotação (variação de 0 a 45°)**: rotacionar os pixels de uma imagem em torno do centro da imagem ou de um ponto específico.

**Transformação de nível de pixel** modificam os valores dos pixels para criar novas versões da imagem original, introduzindo variações sutis e realistas nos dados. Essas transformações tem como vantagens fortalecer a capacidade do modelo lidar com pequenas perturbações de ruído, variação de iluminação e dependência das características relacionadas a coloração. Seguem alguns exemplos de transformações a nível de pixels:

- **Brilho (variação -20% a +20%)**: Modificar o brilho das imagens possibilita ao modelo se ajustar a variações na iluminação, como diferentes condições de luz em dias nublados ou ensolarados, bem como as mudanças na intensidade da luz solar ao longo do dia.
- **Ruído (variação -20% a +20%)**: Inserir ruído nas imagens simula imperfeições na captura de imagens pelo drone e fortalece a capacidade do modelo em lidar com pequenas perturbações na qualidade das imagens.
- **Variação nos canais de cores RGB (variação -20% a +20% para cada canal)**: aplicar esse deslocamento aleatório nos valores dos canais vermelho, verde e azul, permitindo que o modelo aprenda a lidar com diferentes níveis de intensidade de cores e variações de contraste. Isso ajuda o modelo a detectar vacas de diferentes colorações.

Neste projeto, as transformações de *data augmentation* foram realizadas com o auxílio da ferramenta *Albumentation*, uma biblioteca Python com diversas transformações prontas para aplicar nas imagens e nas anotações (BUSLAEV et al., 2020).

Para cada imagem do conjunto de dados foram aplicadas 10 transformações aleatórias de *data augmentation* resultando na construção do conjunto de dados 3. A tabela 8 resume os métodos referentes a cada transformação, juntamente com uma descrição e a probabilidade de ocorrência. Ainda, a figura 28 traz exemplos de imagens resultantes do *data augmentation*.

Ao final dessa otimização, conjunto de dados 3 foi dividido em imagens de treino e teste, aleatoriamente, respeitando a seguinte proporção: 20% em dados de teste e 80% em dados de treino. A tabela 6 traz o novo volume de dados para os conjuntos de treino e teste otimizados.

Tabela 8 – Transformações de *data augmentation* aplicadas para a construção do conjunto de dados V3

Métodos	Descrição	Probabilidade
Flip	Espelhamento vertical ou horizontal	0.7
ShiftScaleRotate	translação e rotação	0.5
RandomBrightnessContrast	variação de brilho e contraste	0.2
RandomGamma	variação de brilho	0.2
RGBShift	Variação da coloração	0.1



Figura 28 – Exemplo de imagens da classe *Cow* pós *data augmentation*

Fonte: adaptado ([SHAO REI KAWAKAMI, 2020](#))

Tabela 9 – Composição da versão 3 dos conjuntos de dados de treino e teste otimizado

Dataset	Cow	NoCow	Total
Treino	6728	6800	13528
Teste	1722	1660	3382

Já a tabela 10 traz, de forma resumida, o aumento do volume das imagens de treino e teste, em comparação com o conjunto de dados v1, após o processo de *data augmentation*.

Tabela 10 – Volume de imagens antes e depois do *data augmentation*

Dataset	antes	depois
Treino	1352	13528
Teste	338	3382

Com os novos conjuntos de treino e teste construídos, realizou-se novamente o treinamento do modelo de detecção, a calibração do modelo de contagem e a validação com o conjunto de imagens inéditas. A tabela 11 resume as configurações dos principais hiperparâmetros para o treinamento do modelo de detecção V3. Já a figura 29 resume o fluxo de trabalho realizado para o treinamento, calibração e avaliação dos modelos três.

Tabela 11 – Parâmetros de treinamento do modelo de detecção V1 e V2

Modelo	Épocas	Batch Size	LR	Otimizador	Pré-teino
Yolo V8 nano	150	16	0.01	SGD	COCO

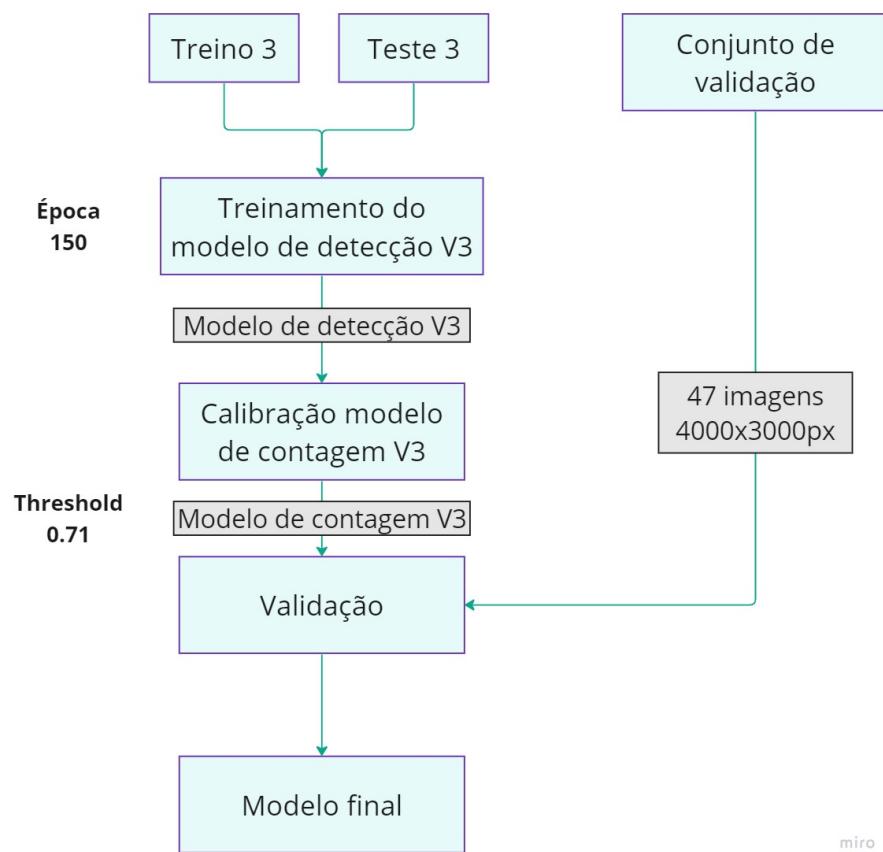


Figura 29 – Fluxograma de treinamento e validação do modelo de detecção e modelo de contagem v3

Fonte: Autoral

# 5 Resultados e Discussões

Essa seção apresenta os resultados obtidos pelos modelos de detecção e modelos de contagem para cada uma das três versões de conjuntos de dados separadamente. Por fim, há uma comparação entre as três versões dos modelos de contagem com base nas métricas de validação descritas no tópico 4.1.4.2.2 obtidas com o conjunto das 47 imagens de validação selecionadas no tópico 4.1.1.2.

## 5.0.1 Conjunto de dados 1

### 5.0.1.1 Resultados do treinamento do modelo de detecção v1

Como vimos na tabela 5, a primeira versão do modelo de detecção foi treinado com 1352 imagens de treino e 338 imagens de teste. A figura 30 ilustra o número de instâncias para cada classe no conjunto de dados utilizados para o treinamento do modelo.

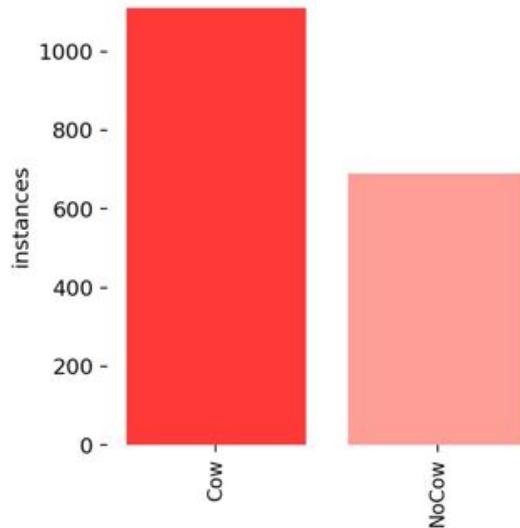


Figura 30 – Número de instâncias de cada classe no conjunto de dados utilizados no treinamento do modelo de detecção v1

Fonte: Autoral

A figura 31 mostra o comportamento de três métricas de erro no decorrer de cada uma das 100 épocas. Nessa figura, os três gráficos superiores foram obtido a partir do conjunto de dados de treino e os gráficos inferiores foram obtido a partir do conjunto de dados de teste. Em cada gráfico, o eixo vertical traz os valores das métrica de erro e o eixo horizontal traz os valores de épocas.

Analizando a figura 31 visualizamos uma queda acelerada do erro, aproximadamente, até a época 50. Em seguida, o erro continua caindo, porém, lentamente. A queda

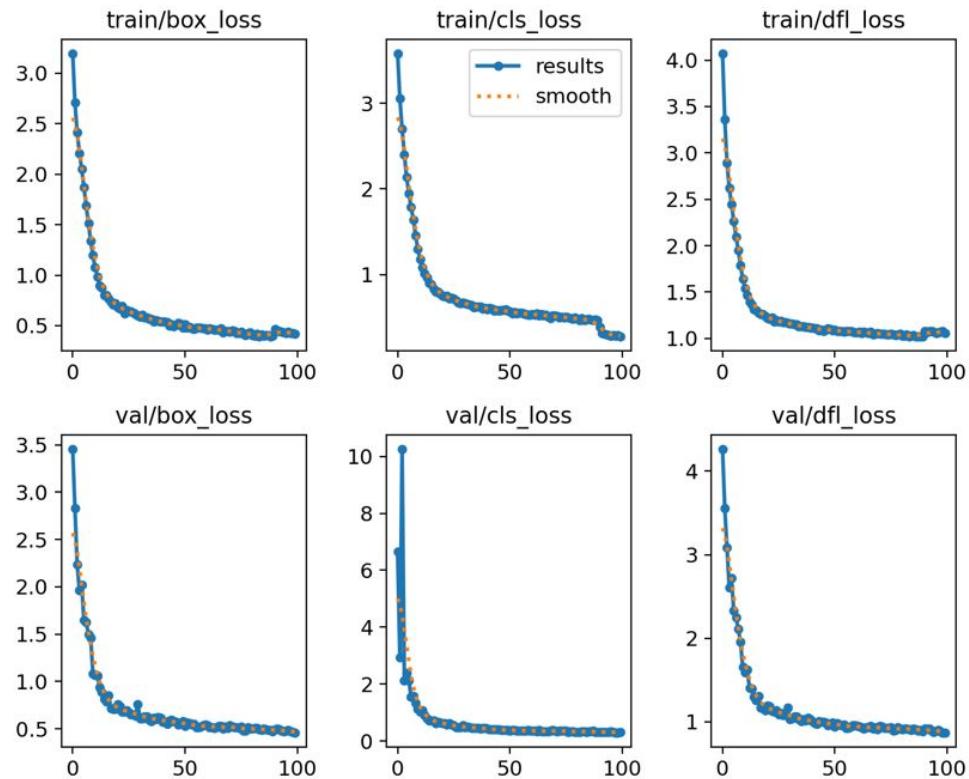


Figura 31 – Métricas de erros no decorrer de 100 épocas do treinamento do modelo de detecção v1

Fonte: Autoral

dessas métricas de erro tanto no conjunto de treino quanto no conjunto de teste mostram que o modelo de detecção está aprendendo com as características do objeto e conseguindo generalizado para dados novos.

Já a figura 32, ilustra o comportamento das métricas de precisão e *recall* no decorrer de cada uma das 100 épocas. Como podemos ver na figura, as duas métricas crescentes juntas rapidamente. Em seguida, a partir da época 50, começam a estabilizarem próximos à 0.97. Portanto, Esse comportamento indica que o modelo é confiável em suas previsões positivas e é capaz de identificar a maioria dos casos positivos.

A figura 33 associa os resultados de precisão e *recall* e constrói a curva precisão-*recall* para o modelo de detecção v1. A curva laranja representa os resultados para a classe *NoCow*. Já a curva azul celeste representa os resultados da classe *Cow*.

Analizando essa imagem 33, percebe-se que a classe de objetos *NoCow* obteve uma curva próxima da ideal, o que pode indicar um *overfitting* para essa classe em específico. Isso pode ter ocorrido por que a classe *NoCow* abrange tudo na imagem que não é vaca. Dessa forma, objetos com características muito diferentes, como por exemplo: uma vegetação e uma estrada, são considerados da mesma classe: *NoCow*. Dessa forma, isso pode dificultar a capacidade de generalização do modelo para essa classe. Por outro lado, a

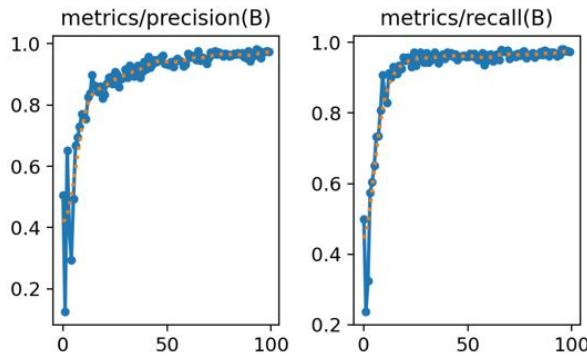


Figura 32 – Métricas de precisão e *recall* à cada época durante o treinamento do modelo de detecção v1

Fonte: Autoral

curva referente a classe *Cow* obteve resultados excelentes de 0.987. Isso indica que o modelo classifica de forma confiável a maioria das instâncias da classe *Cow*. Como o modelo, possivelmente, *overfitting* apenas para a classe de objetos *NoCow* e obteve um excelente resultado para a classe *Cow*, ainda, como o objetivo final desse projeto é contar objetos da classe *Cow*, esse modelo de detecção poderia ser utilizado.

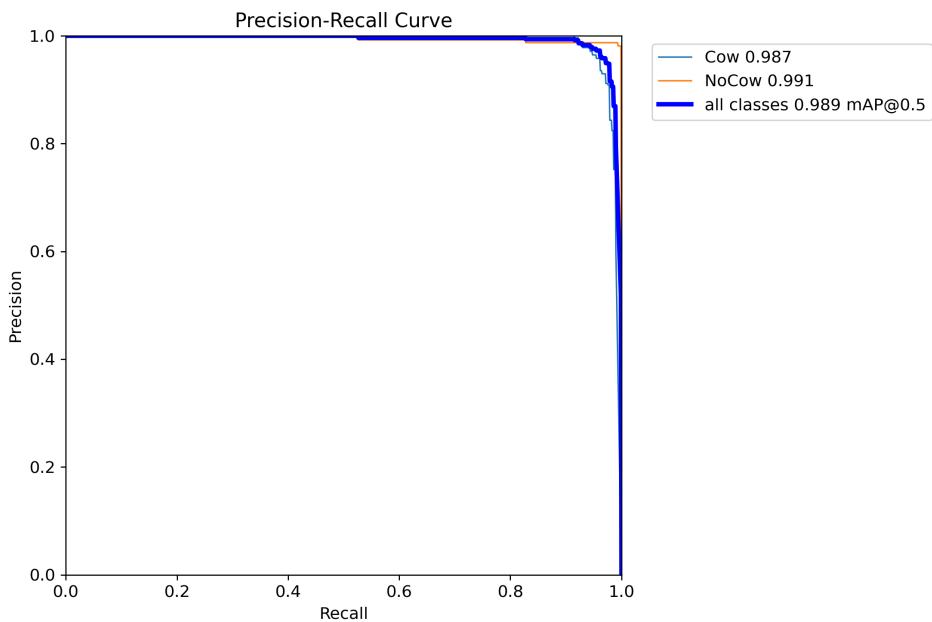


Figura 33 – Curva precisão-*recall* para o modelo de detecção v1

Fonte: Autoral

Ainda, a partir da precisão e o *recall* pode-se calcular também as curvas de mAP ilustradas na figura 34. Nesse figura, o eixo vertical representa o mAP e o eixo horizontal representa os valores de 0 a 100 épocas. Analisando os gráficos, percebe-se um crescimento acelerado da curva logo nas primeiras épocas, se estabilizando bastante próximos do valor 1.0 ao final das 100 épocas. Isso indica que o modelo é capaz de detectar objetos com alta

precisão e *recall* para as duas classes *Cow* e *NoCow*.

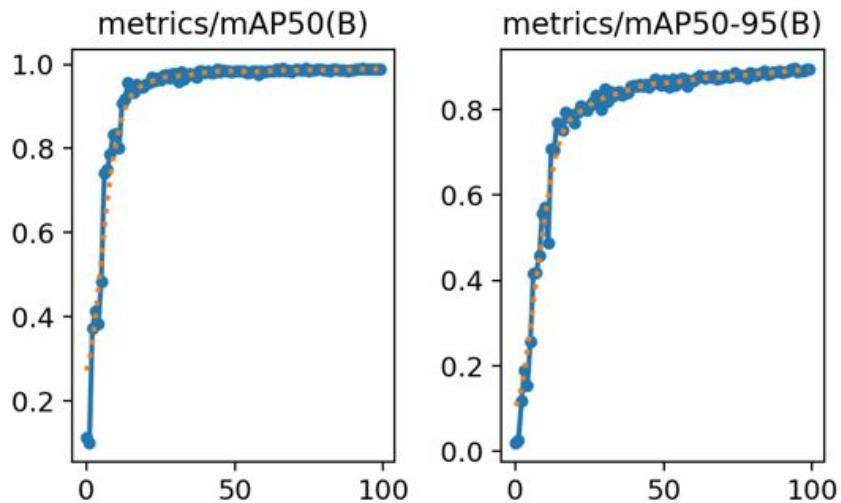


Figura 34 – Métricas de *mean Average Precision* para cada época de treinamento do modelo de detecção v1

Fonte: Autoral

Outras métricas produzidas durante a etapa de treinamento do modelo de detecção a partir do conjunto de dados 1 estão em anexo. Em seguida, na figura 35 são apresentados alguns exemplos de detecção realizados por este modelo em 14 imagens de teste. As imagens da esquerda são os objetos com suas anotações reais. Já as imagens da direita mostram as previsões do modelo juntamente com a taxa de confiança de cada predição.

Inspecionando a figura 35, é percebido um comportamento de predição interessante do modelo de detecção. As vacas que estão parcialmente nas janelas não são detectadas ou são detectadas com uma baixa taxa de confiança. Isso é interessante, pois afeta positivamente a performance do modelo de contagem evitando contar vacas repetidamente em múltiplas janelas.

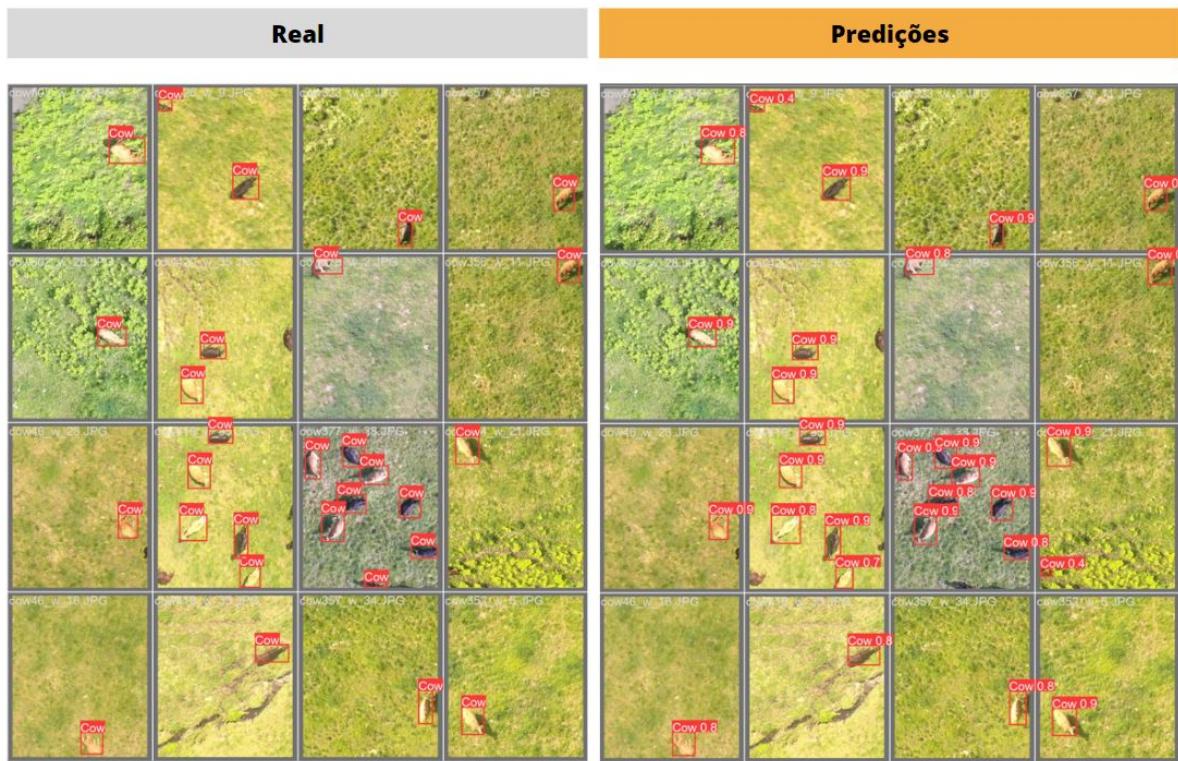


Figura 35 – Comparaçāo entre as anotações reais e as predições do modelo de detecção para um grupo de 14 imagens de teste

Fonte: Autoral

#### 5.0.1.2 Resultados do modelo de contagem v1

Tabela 12 – *Ranking* das 5 melhores combinações de *threshold* após processo de calibração do módulo de contagem v1

Acertos	Largura	Altura	threshold
16/25	500	600	0.66
15/25	500	600	0.67
15/25	500	600	0.65
15/25	500	600	0.63
15/25	500	600	0.62

Com o modelo de detecção criado, realizou-se a calibração do parâmetro *threshold* para o modelo de contagem conforme descrito no tópico 4.1.4.2.1. Como visto no tabela 12, o valor de *threshold* que obteve os melhores resultados na calibração foi de 0.66. Dessa forma, usou-se esse valor para realizar a validação do modelo de contagem v1 a partir do conjunto de dados 1. Os resultados completos podem ser consultado na imagem 69 em apêndices. Já a figura 38 apresenta um exemplo de imagem contabilizada pelo modelo de contagem durante o processo de validação. Nessa imagem, o modelo contabiliza, corretamente, 11 vacas.

A tabela 13 apresenta o número real total de vacas presente em todo o conjunto de

imagens de validação, a quantidade predita pelo modelo e a diferença de contagem entre esses dois valores. Já figura 36 mostra de maneira resumida os resultados da validação para essa primeira versão de modelo de contagem por meio da tabela matriz de confusão.

Tabela 13 – Quantidade de vacas reais e quantidade de vacas preditas pelo modelo de contagem v1

Valor real	Valor predito	Diferença de contagem (DiC)
232	209	23

		Preditos	
		Positivo	Negativo
Real	Positivo	205	27
	Negativo	4	0

Figura 36 – Matriz de confusão obtida no processo de validação do modelo de contagem v1

Fonte: Autoral

Analizando a tabela 13 e a figura 36, percebe-se que das 209 previsões que o modelo contabilizou como vaca, apenas 205 realmente foram contabilizadas corretamente. Isso significa que o modelo de contagem foi capaz de contabilizar corretamente, aproximadamente, 88,36% das vacas presentes nas 47 imagens de validação.

Ainda, como descrito no tópico 2.6.1 o número de falsos negativos neste projeto corresponde a quantidade de vacas que o modelo deixou de contar. Dessa forma, 27 vacas ficaram de fora da contagem predita pelo modelo. Já os valores de falso positivos, neste projeto, corresponde a quantidade de vacas duplicadas que o modelo contabilizou, juntamente, com os objetos da classe *NoCow* que o modelo detectou como *Cow*.

A figura 37 mostra um exemplo de uma imagem do conjunto de validação que o modelo de contagem v1 apresentou um número de vacas errado. Nessa imagem, percebemos que o modelo deixou de contabilizar 3 vacas. Ainda, o modelo contabilizou um sombra entre a vegetação e a estrada como vaca.

Analizando os erros de contagem obtidos nessa primeira versão do modelo, percebe-se que em nenhum momento ele contabilizou vacas duplicadamente. Dessa forma, os 4 valores de falso positivo identificados na figura 36 refere-se apenas a objeto da classe *NoCow* que o modelo identificou como *Cow*. Esse resultado evidencia, também, que a escolha de janelas com dimensões de 500x600 pixels, dimensões próximas aos da rede neural convolucionar, não apresenta um grande impacto negativo na contagem duplicada de vacas.



Figura 37 – Exemplo de uma contagem errada do modelo de contagem v1 em uma imagem do conjunto de validação

Fonte: Autoral



Figura 38 – Exemplo de uma contagem realizada pelo modelo de contagem v1 em uma imagem do conjunto de validação

Fonte: Autoral

Por fim, a tabela 14 apresenta as métricas de avaliação descritas no tópico 2.6 obtidas pelo modelo de contagem v1. A primeira linha apresenta os resultados totais com base no conjunto de validação. Já a segunda linha refere-se a média das métricas obtidas em cada uma das 47 imagens de validação.

Tabela 14 – Métricas de avaliação obtidos pelo modelo de contagem v1

V1	Precisão	<i>Recall</i>	<i>F1 Score</i>	Acurácia	MSE	MAE	DiC	TC
Total	0.981	0.884	0.930	0.869	2.280	0.099	23	66.219
Média	0.973	0.912	0.932	0.892	0.209	0.108	0.574	1.409

## 5.0.2 Conjunto de dados 2

### 5.0.2.1 Resultados do treinamento do modelo de detecção v2

O processo de treinamento do modelo de detecção V2 foi realizado 1124 imagens de treino e 280 imagens de teste, assim como vimos na tabela 6. A figura 39 ilustra o número de instâncias para cada classe no conjunto de dados utilizados para o treinamento do modelo.

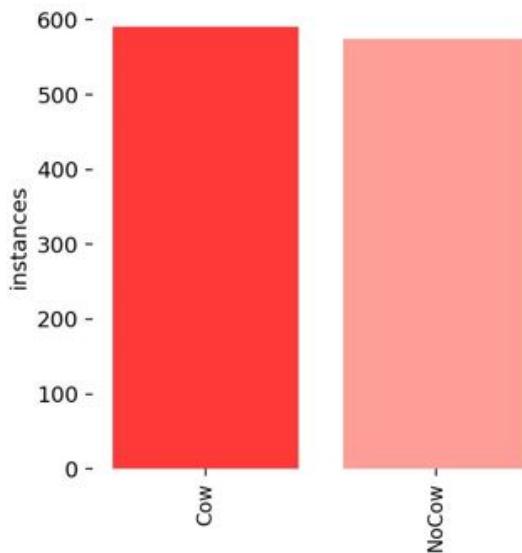


Figura 39 – Número de instâncias de cada classe no conjunto de dados utilizados no treinamento do modelo de detecção v2

Fonte: Autoral

A figura 40 mostra o comportamento de três métricas de erro no decorrer de cada uma das 100 épocas. Nessa figura, os três gráficos superiores foram obtido a partir do conjunto de dados de treino e os gráficos inferiores foram obtido a partir do conjunto de dados de teste. Em cada gráfico, o eixo vertical traz os valores das métrica de erro e o eixo horizontal traz os valores de épocas.

Analizando a figura 40, assim como no primeiro treinamento, visualizamos uma queda acelerada do erro, aproximadamente, até a época 50. Em seguida, o erro continua caindo, porém, lentamente. Como essa queda ocorre tanto no conjunto de treino quanto no conjunto de teste o modelo de detecção está aprendendo com as características do objeto e conseguindo generalizado para dados novos.

Já a figura 32, ilustra o comportamento das métricas de precisão e *recall* no decorrer de cada uma das 100 épocas. Analisando essa figura e comparando com os resultados obtidos pelo modelo anterior, apresentado na figura 32, percebemos uma maior dispersão dos valores de precisão e *recall* obtido a baixo de 50 épocas.

Mesmo disperso, as duas métricas crescente juntas e a partir da época 50, começam

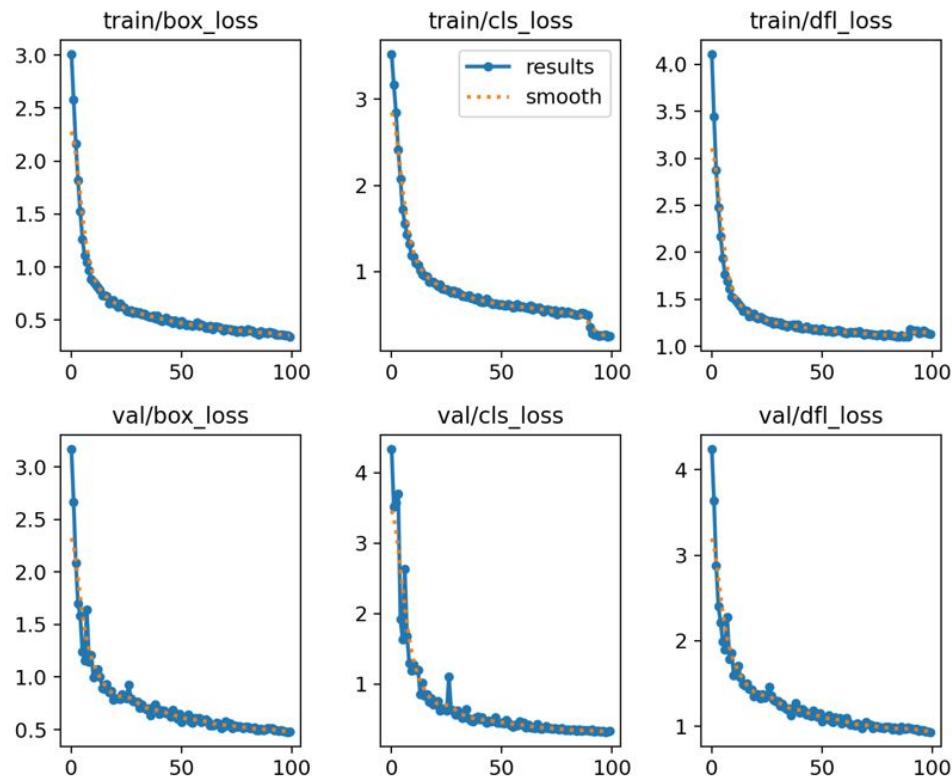


Figura 40 – Gráfico da curva de métricas de erro no decorrer de cada uma das 100 épocas

Fonte: Autoral

a estabilizarem, a precisão próximos à 0.97 e o *recall* próximo de 0.98. Esse valor de *recall* obtido após 100 épocas é superior ao valor obtido no treinamento do modelo de detecção V1.

Por fim, com base nessas métricas conclui-se que o modelo é confiável em suas previsões positivas e é capaz de identificar a maioria dos casos positivos.

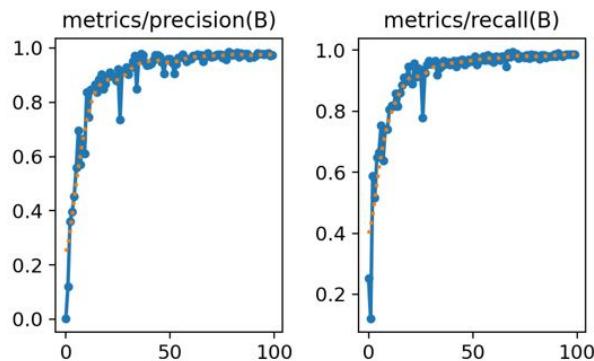


Figura 41 – Métricas de precisão e *recall* à cada época durante o treinamento do modelo de detecção v2

Fonte: Autoral

A figura 42 traz a curva precisão-*recall* para o modelo de detecção v2. Onde A curva laranja representa os resultados para a classe *NoCow*. Já a curso azul celeste

representa os resultados da classe *Cow*.

Analizando essa imagem 42, percebe-se que a classe de objetos *NoCow* também obteve uma curva praticamente ideal, o que pode indicar um *overfitting* para essa classe em específico, assim como o modelo de detecção v1. Já a curva referente a classe *Cow* obteve resultados excelentes de 0.990, valor inferior ao do modelo de detecção v1. Entretanto, isso indica uma boa performance do modelo em classificar de forma confiável a maioria das instâncias da classe *Cow*.

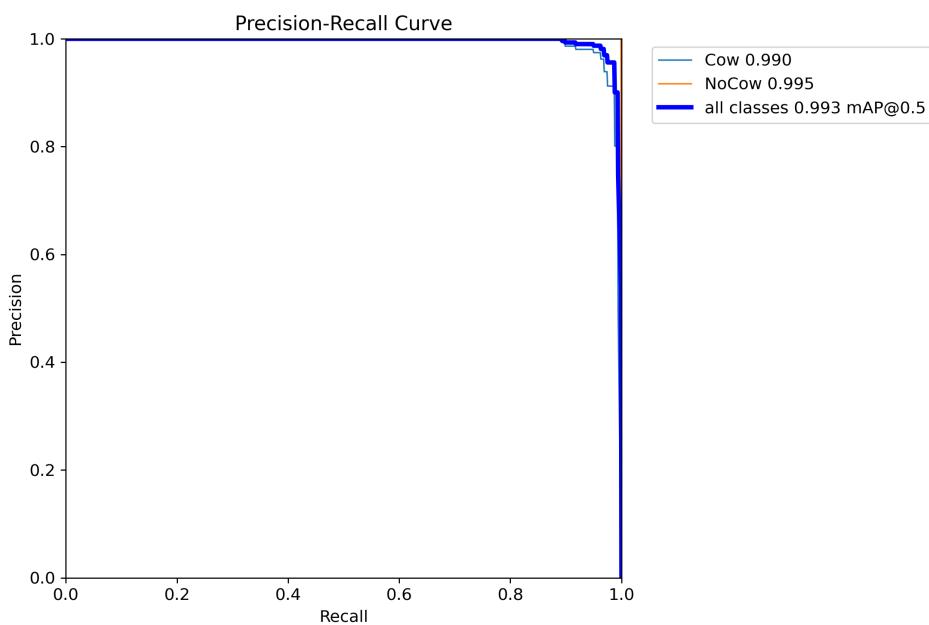


Figura 42 – Curva precisão-recall para o modelo de detecção v2

Fonte: Autoral

Analizando as métricas de *mean Average Precision* apresentadas na figura 42, percebe-se um crescimento acelerado da curva logo nas primeiras épocas, terminando, ao final das 100 épocas de treinamento no valor 0.891. Embora inferior aos resultados obtido pelo modelo de detecção v1, esse resultado indica que o modelo é capaz de detectar objetos com alta precisão e *recall* para as duas classes *Cow* e *NoCow*.

Em seguida, na figura 44 são apresentados alguns exemplos de detecção realizados por este modelo em 14 imagens de teste. As imagens da esquerda são os objetos com suas anotações reais. Já as imagens da direita mostram as predições do modelo juntamente com a taxa de confiança de cada predição.

Analizando a figura 44, notamos que o modelo detecta todas as vacas. Notamos também que, diferentemente do primeiro modelo de detecção, as vacas que estão parcialmente na janela estão sendo detectadas com uma confiança relativamente alta e isso é preocupante, pois pode afetar negativamente a performance do modelo de contagem contando vacas repetidamente em múltiplas janelas.

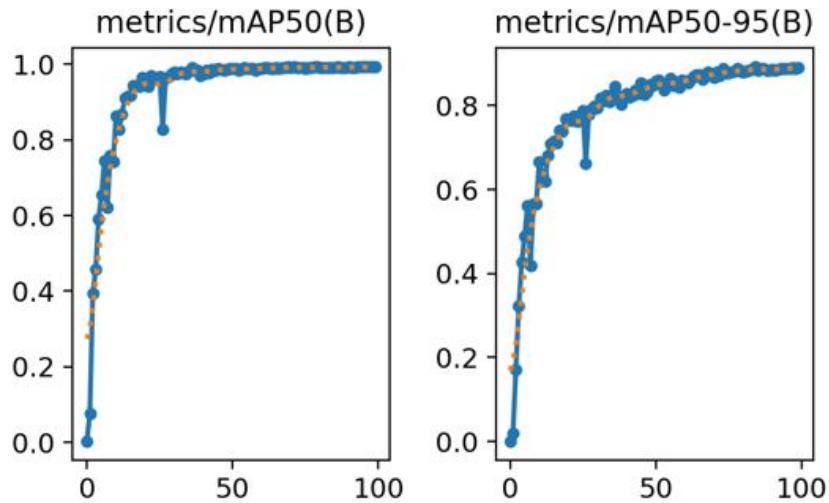


Figura 43 – Métricas de *mean Average Precision* para cada época de treinamento do modelo de detecção v2

Fonte: Autoral

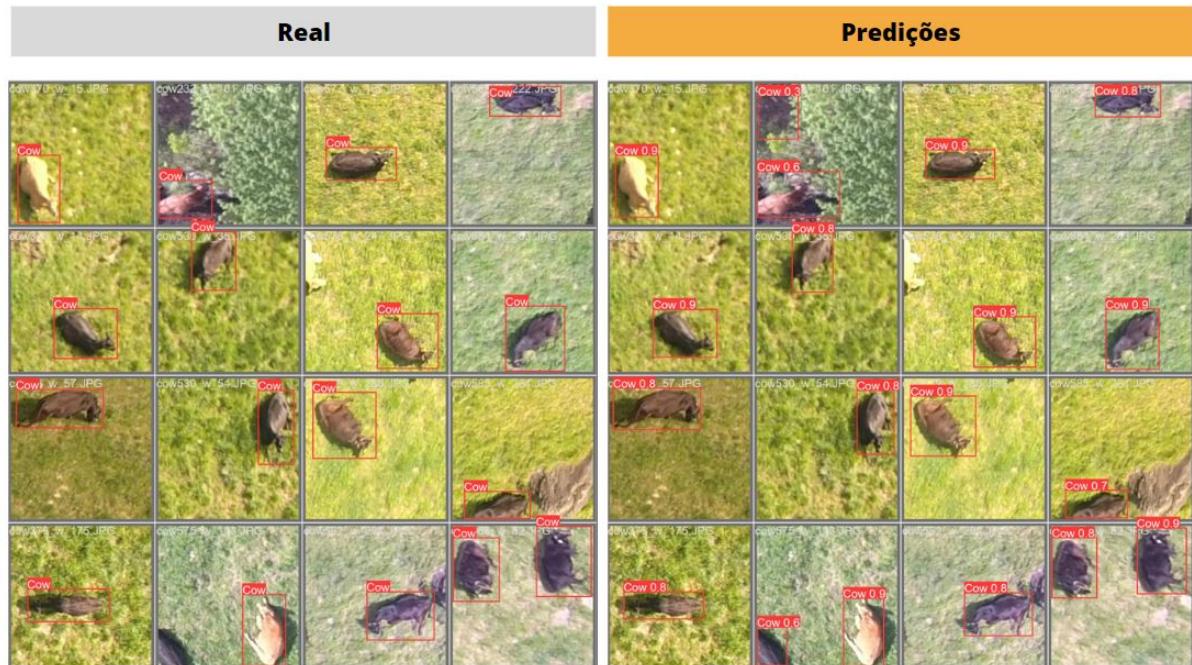


Figura 44 – Comparação entre as anotações reais e as predições do modelo de detecção v2 para um grupo de 14 imagens de teste

Fonte: Autoral

Outro comportamento preocupante, é a detecção do solo como objeto da classe *cow* com confiança de 0.3 na segunda janela. Isso pode ter ocorrido devido a semelhança do tamanho e coloração da deformação no solo e uma vaca vista de cima. Para tentar resolver esse tipo de erro, pode-se diversificar o conjunto de dados aplicando técnicas de *data augmentation* com transformações a nível de pixel descritas no tópico 4.2.1. Embora essa detecção ocorreu com uma confiança baixa, isso pode ser um indicativo de que o

modelo de contagem irá contabilizar objetos do ambiente descontrolado como vacas.

Outras métricas produzidas durante a etapa de treinamento do modelo de detecção v2 em anexo.

#### 5.0.2.2 Resultados do modelo de contagem v2

Tabela 15 – *Ranking* das 5 melhores combinações de *threshold* após processo de calibração do módulo de contagem v2

Acertos	Largura	Altura	<i>threshold</i>
7/25	200	200	0.71
7/25	200	200	0.70
6/25	200	200	0.73
6/25	200	200	0.72
5/25	200	200	0.76

Com o modelo de detecção criado, realizou-se a calibração do parâmetro *threshold*. Com base nos resultados apresentados na tabela 12 o valor de *threshold* que obteve os melhores resultados na calibração foi de 0.71. Ainda, O processo de calibração já traz resultados interessantes. Pois para um conjunto de dados de 25 imagens utilizadas para o processo de calibração, apenas 7 imagens tiveram o número de predições iguais ao número real de vacas. Isso é um indicativo de que o modelo de contagem v2 possui uma acurácia inferior ao modelo de contagem v1.

A figura 70, em apêndices, apresenta a tabela com os resultados completo do processo de validação do modelo de contagem v2 para cada uma das 47 imagens do conjunto de validação. Já a figura 47 traz, como exemplo, uma imagem de validação onde modelo de contagem v2 detectou e contou, corretamente, 4 vacas.

Curiosamente, a quantidade de vacas reais e a quantidade de vacas contabilizadas pelo modelo de contagem v2 coincidiram, como podemos observar pela tabela 16. Consequentemente, não há diferença de contagem.

Tabela 16 – Quantidade de vacas reais e quantidade de vacas preditas pelo modelo de contagem v2

Valor real	Valor predito	Diferença de contagem (DiC)
232	232	0

Por outro lado, analisando a matriz de confusão ilustrada na figura 45, concluímos que nem todos instancias contabilizados como vacas correspondem, realmente, a vacas.

Como podemos notar na figura 45, apenas 202 predições correspondem, verdadeiramente, a vaca. Isso corresponde a, aproximadamente, 87,07% das vacas presentes nas 47 imagens de validação. Ou seja, esse segundo modelo contou 7 vacas a menos em comparação com o modelo de contagem V1. Ainda, como podemos ver no quadrante referente

		Preditos	
		Positivo	Negativo
Real	Positivo	202	30
	Negativo	30	0

Figura 45 – Matriz de confusão obtida no processo de validação do modelo de contagem v2

Fonte: Autoral

ao falso positivo, este novo modelo deixou de detectar e contar 30 vacas das 232 vacas existentes no conjunto de imagens de validação.

Analizando os valores de falso positivos, vemos que o modelo contou 30 instâncias de forma errada. A partir da tabela com os resultados completo da validação, ilustrada na figura 70, percebemos que desse valor, 12 correspondem ao número de vacas contadas duplicadamente e 18 correspondem ao número de objetos da classe *NoCow* que o modelo detectou como *Cow*. O surgimento de vacas duplicadas evidencia uma pior performance do modelo de contagem com janelas menores em comparação ao modelo de contagem v1.

A figura 46 mostra um exemplo de uma imagem do conjunto de validação que o modelo de contagem v2 apresentou um número de vacas errado. Nessa imagem, percebemos que o modelo deixou de contabilizar 3 vacas. Ainda, o modelo contabilizou 6 sombras e uma estrutura como vaca. Por fim, o modelo ainda contabilizou, duplicadamente, 2 vacas, apresentando um total de 11 vacas onde, na verdade, só há 5.

A tabela 17 apresenta as métricas de avaliação descritas no tópico 2.6 obtidas pelo modelo de contagem v2, construído a partir de janelas de dimensão 200x200 pixels. A primeira linha apresenta os resultados totais com base no conjunto de validação. Já a segunda linha refere-se à média das métricas obtidas em cada uma das 47 imagens de validação.

Tabela 17 – Métricas de avaliação obtidos pelo modelo de contagem v2

V2	Precisão	Recall	F1 Score	Acurácia	MSE	MAE	DiC	TC
Total	0.871	0.871	0.871	0.0771	0.000	0.000	0	235.780
Média	0.853	0.843	0.828	0.761	0.446	0.236	0.851	5.017

Por fim, embora o modelo de contagem v2 contabilizou, exatamente, o mesmo número total de vacas reais presente no conjunto de validação, resultando em um módulo de diferença de contagem igual a zero, este modelo apresentou uma performance inferior ao modelo de contagem v1. Essa segunda versão apresentou contagens duplicadas e uma grande sensibilidade aos ruídos de um ambiente descontrolado. Consequentemente, as

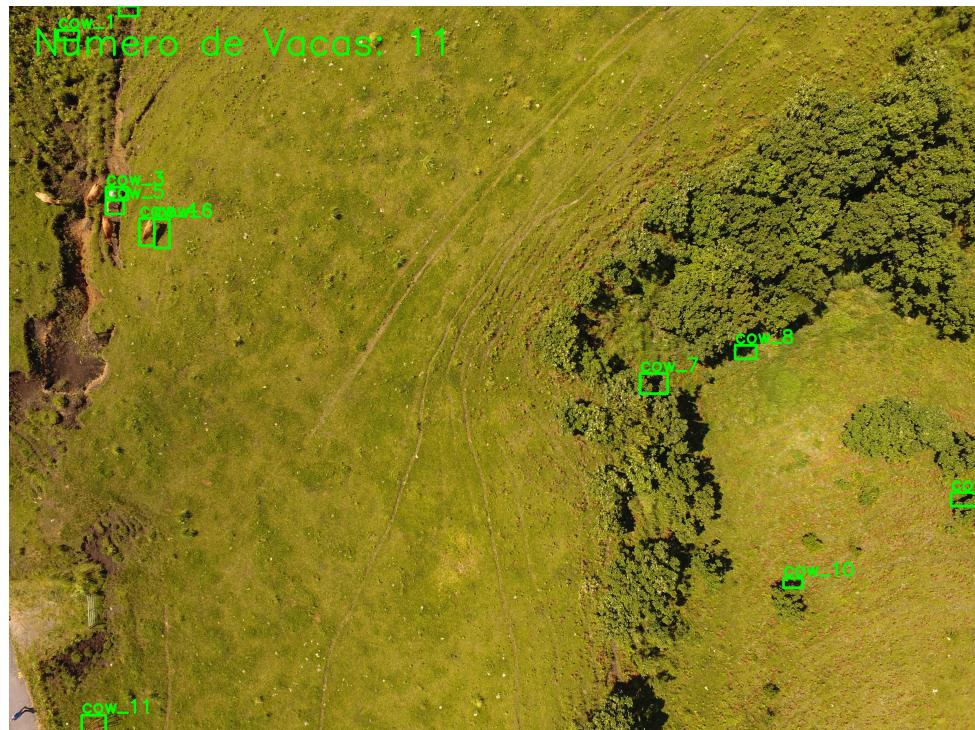


Figura 46 – Exemplo de uma contagem errada do modelo de contagem v2 em uma imagem do conjunto de validação

Fonte: Autoral



Figura 47 – Exemplo de uma contagem realizada pelo modelo de contagem v2 em uma imagem do conjunto de validação

Fonte: Autoral

métricas de avaliação precisão, *Recall*, *F1 Score* e acurácia sofreram uma redução em comparação à primeira versão.

### 5.0.3 Conjunto de dados 3

Com base nos resultados obtidos pelos conjuntos de dados 1 e 2, conclui-se que o modelo de contagem v1, construído a partir de janelas de dimensão 500x600 pixels, apresentou uma melhor performance com os dados de validação. Dessa forma, o conjunto de dados passou por um processo de otimização descrito no tópico [4.2.1](#) resultando no conjunto de dados 3.

#### 5.0.3.1 Resultados do treinamento do modelo de detecção v3

Após o processo de otimização, como podemos ver na tabela [9](#), o conjunto de dados 3 resultou em 16.910 imagens diversas, das quais 13.528 compõem o conjunto de treino e 3.382 o conjunto de teste. A figura [48](#) ilustra o número de instâncias para cada classe no conjunto de dados utilizados para o treinamento do modelo.

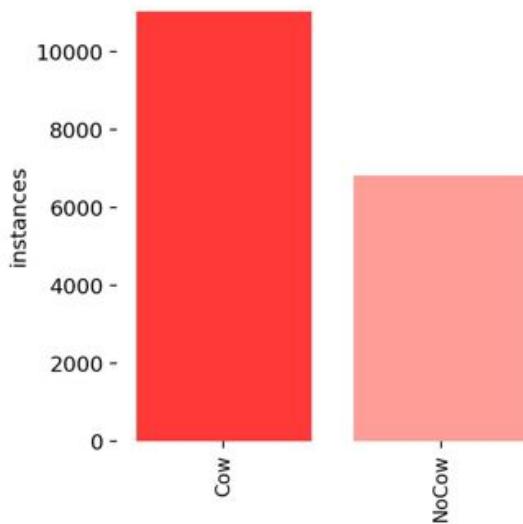


Figura 48 – Número de instâncias de cada classe no conjunto de dados utilizados no treinamento do modelo de detecção v3

Fonte: Autoral

Diferentemente do treinamento do modelo de detecção v1, o treinamento para essa terceira versão foi realizado com 150 épocas. A figura [49](#) mostra o comportamento de três métricas de erro no decorrer de cada uma das 150 épocas. Nessa figura, os três gráficos superiores foram obtido a partir do conjunto de dados de treino e os gráficos inferiores foram obtido a partir do conjunto de dados de teste. Em cada gráfico, o eixo vertical traz os valores das métrica de erro e o eixo horizontal traz os valores de épocas.

Analizando a figura [49](#), visualizamos uma queda muito acelerada do erro logo nas primeiras épocas. Essa queda acelerada é um comportamento desejado durante o processo de otimização do modelo de rede neural convolucional. Isso é um indicativo de que os dados de treinamento são bem estruturados, limpos e representativos do problema

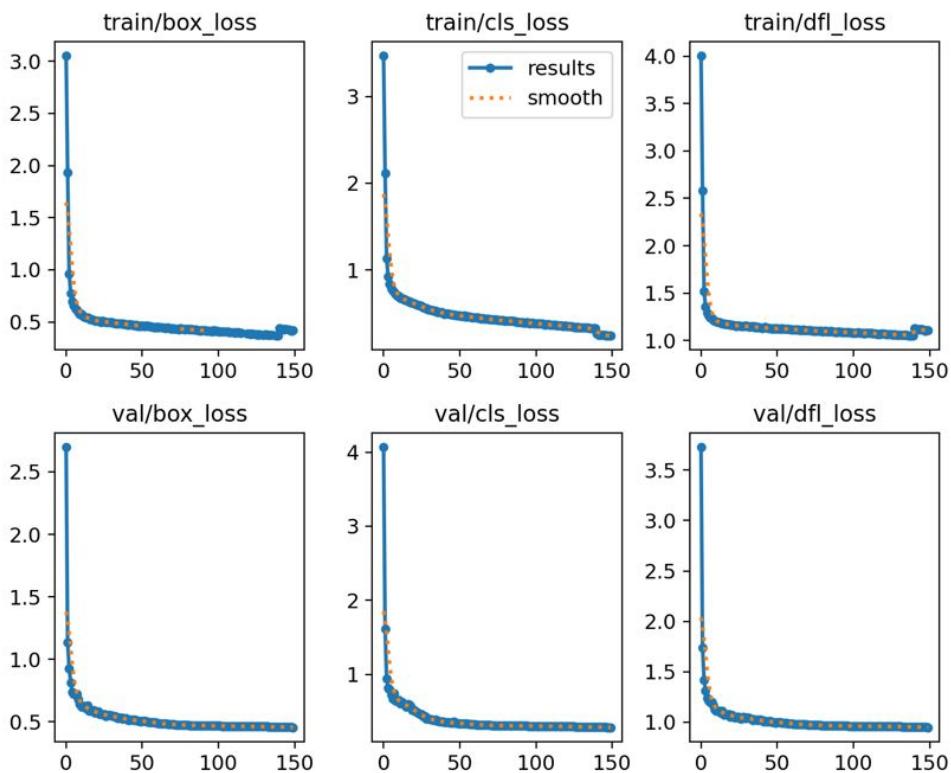


Figura 49 – Métricas de erros obtidas durante as 150 épocas de treinamento do modelo de detecção v3

Fonte: Autoral

que o modelo está tentando resolver. Dessa forma, a CNN pode aprender padrões mais facilmente, resultando em uma rápida redução do "loss" (GOODFELLOW; BENGIO; COURVILLE, 2016).

Ainda, como as quedas dessas métricas de erro ocorre tanto no conjunto de treino quanto no conjunto de teste mostram que o modelo de detecção está aprendendo com as características do objeto e conseguindo generalizado para dados novos.

Já a figura 50, ilustra o comportamento das métricas de precisão, *recall* e o mAP no decorrer de cada uma das 150 épocas. Como podemos ver na figura, que essas métricas crescente juntas muito rapidamente, logo nas primeiras épocas. Em seguida, um pouco antes da época 50, a precisão e *recall* começam a convergirem próximos aos valores 0.980 e 0.978, respectivamente. Esse comportamento indica que o modelo é bastante confiável em suas previsões positivas e é capaz de identificar a maioria dos casos positivos.

Já a *mean Average Precision at 50*, o mAP50, convergiu para o valor 0.990. Isso significa que ao se considerar apenas as detecções com confiança à cima de 50%, o modelo conseguiu acertar 99% das previsões para as múltiplas classes. Já quando se considera apenas as detecções com confiança entre 50% e 95%, o modelo acerta 0.892, como podemos ver pela mAP50-95.

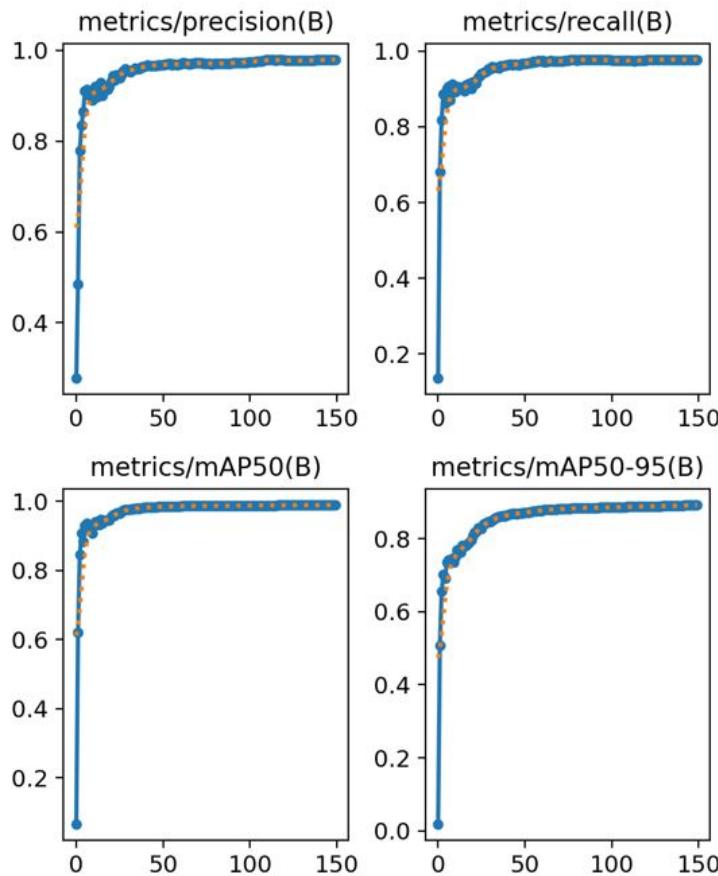


Figura 50 – Métricas de precisão, *recall* e mAP à cada época durante o treinamento do modelo de detecção v3

Fonte: Autoral

Ainda, a partir da precisão e o *recall* pode-se calcular também as curvas de mAP ilustradas na figura 42. Nesse figura, o eixo vertical representa o mAP e o eixo horizontal representa os valores de 0 a 150 épocas. Analisando os gráficos, percebe-se um crescimento acelerado da curva logo nas primeiras épocas, se estabilizando bastante próximos do valor 1.0 ao final das 150 épocas. Isso indica que o modelo é capaz de detectar objetos com alta precisão e *recall* para as duas classes *Cow* e *NoCow*.

A figura 51 apresenta a curva precisão-*recall* para o modelo de detecção v3. A curva laranja representa os resultados para a classe *NoCow*. Já a curso azul celeste representa os resultados da classe *Cow*.

Analizando essa imagem 51, percebe-se que a classe de objetos *NoCow* obteve uma curva próxima da ideal, o que pode indicar um *overfitting* para essa classe em específico. Por outro lado, a curva referente a classe *Cow* obteve resultados excelentes de 0.987. Isso indica que o modelo classifica de forma confiável a maioria das instâncias da classe *Cow*. Como o objetivo final desse projeto é contar objetos da classe *Cow*, esse modelo de detecção poderia ser utilizado para o módulo de contagem.

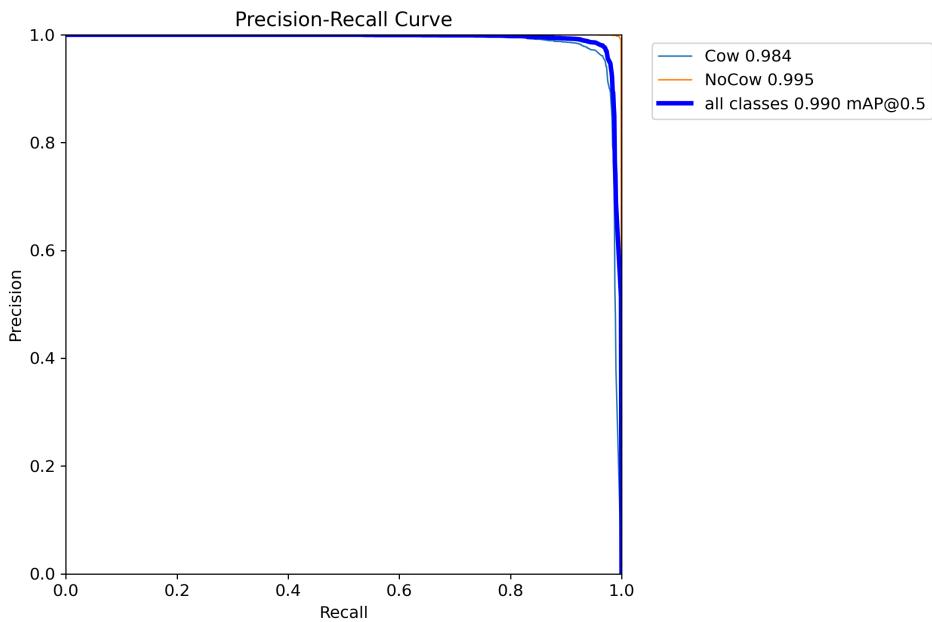


Figura 51 – Curva precisão-*recall* para o modelo de detecção v3

Fonte: Autoral

Outras métricas produzidas durante a etapa de treinamento do modelo de detecção a partir do conjunto de dados 3 estão em anexo. Em seguida, na figura 52 são apresentado alguns exemplos de detecção realizados por este modelo em 14 imagens de teste. As imagens da esquerda são os objetos com suas anotações reais. Já as imagens da direita mostram as predições do modelo juntamente com a taxa de confiança de cada predição.

Inspecionando a figura 52, é percebido que modelo de detecção detecta as vacas que estão parcialmente nas janelas com uma taxa de confiança relativamente baixa. Isso é interessante, pois afeta positivamente a performance do modelo de contagem evitando contar vacas repetidamente em múltiplas janelas.



Figura 52 – Comparaçāo entre as anotações reais e as predições do modelo de detecção v3 para um grupo de 14 imagens de teste

Fonte: Autoral

#### 5.0.3.2 Resultados do modelo de contagem v3

Tabela 18 – *Ranking* das 5 melhores combinações de *threshold* após processo de calibração do módulo de contagem v3

Acertos	Largura	Altura	threshold
14/25	500	600	0.71
14/25	500	600	0.70
14/25	500	600	0.69
13/25	500	600	0.72
13/25	500	600	0.68

Com o modelo de detecção v3 treinado, atingindo métricas superiores ao modelos de detecção v1 e v2, como pode ser observado na tabela 22, realizou-se a calibração do parâmetro *threshold* para o modelo de contagem conforme descrito no tópico 4.1.4.2.1. Como visto no tabela 18, o valor de *threshold* que obteve os melhores resultados na calibração foi de 0.71. Esse valor obtido é superior ao *threshold* utilizado na primeira versão do modelo de contagem, o que já evidencia uma melhoria. Pois um *threshold* mais alto significa que as contagem do modelo de contagem v3 devem ser mais precisas.

Com o modelo de contagem v3 calibrado, avaliou-se a performance do modelo com as 47 imagens do conjunto dados de validação. Os resultados completos podem ser

consultado na imagem 71 em apêndices.

já a figura 55 apresenta um exemplo de imagem contabilizada pelo modelo de contagem v3 durante o processo de validação. Essa imagem, corresponde a mesma foto apresentada na figura 37, onde o modelo de contagem v1 havia contabilizado erradamente. Já o terceiro modelo ignorou qualquer interferência de ruído da imagem e contabilizou corretamente o grupo das 6 vacas.

A tabela 19 apresenta o número real total de vacas presente em todo o conjunto de imagens de validação, a quantidade predita pelo modelo e a diferença de contagem entre esses dois valores. Já figura 53 mostra de maneira resumida os resultados da validação para essa primeira versão de modelo de contagem por meio da tabela matriz de confusão.

Tabela 19 – Quantidade de vacas reais e quantidade de vacas preditas pelo modelo de contagem v3

Valor real	Valor predito	Diferença de contagem (DiC)
232	215	17

		Preditos	
		Positivo	Negativo
Real	Positivo	212	20
	Negativo	3	0

Figura 53 – Matriz de confusão obtida no processo de validação do modelo de contagem v3

Fonte: Autoral

Analizando a tabela 19 e a figura 53, percebe-se que das 215 previsões que o modelo contabilizou como vaca, 212 realmente foram contabilizadas corretamente. Isso significa que o modelo de contagem foi capaz de contabilizar corretamente, aproximadamente, 91,14% das vacas presentes nas 47 imagens de validação. Ainda, analisando o número de falsos negativos, apenas 20 vacas ficaram de fora da contagem predita pelo modelo.

Quando olhamos os valores de falso positivos, percebe-se que essa terceira versão teve uma performance muito superior em relação aos modelos de contagem v1 e v2. Isso porque o modelo de contagem v3 apresentou apenas 3 contagens associadas a erros de duplicidade na contagem ou classificação errada de objetos da classe *NoCow*. Isso reflete a precisão do modelo de detecção ilustrada na figura 50.

Investigando a natureza desses 3 falso positivos por meio da tabela completa dos resultados da validação, percebe-se o surgimento de 3 contagens duplicadas. Por outro lado, percebe-se a ausência de classificação errada de instâncias da classe *NoCow*, o que

é positivo. A figura 54 mostra um exemplo de uma imagem do conjunto de validação que o modelo de contagem v2 apresentou um número de vacas errado. Nessa imagem, percebemos que o modelo deixou de contabilizar 3 vacas. Ainda, o modelo contabilizou um sombra entre a vegetação e a estrada como vaca.

A figura 54 mostra um exemplo de uma imagem do conjunto de validação que o modelo de contagem v3 apresentou um número de vacas errado. Nessa imagem, embora a contagem apresentada coincida com o número real de vacas, percebemos que o modelo deixou de contabilizar 1 vaca. Ainda, o modelo contabilizou uma vaca duplicadamente, pois os com rótulos "*cow3\_*" e "*cow\_13*" refere-se a mesma vaca. Por fim, percebe-se também que as sombras, a vegetação, a deformação no solo e outros ruído do ambiente descontrolado não interferiram na contagem.



Figura 54 – Exemplo de uma contagem errada do modelo de contagem v3 em uma imagem do conjunto de validação

Fonte: Autoral

Por fim, a tabela 20 apresenta as métricas de avaliação descritas no tópico 2.6 obtidas pelo modelo de contagem v3. A primeira linha apresenta os resultados totais com base no conjunto de validação. Já a segunda linha refere-se a média das métricas obtidas em cada uma das 47 imagens de validação.

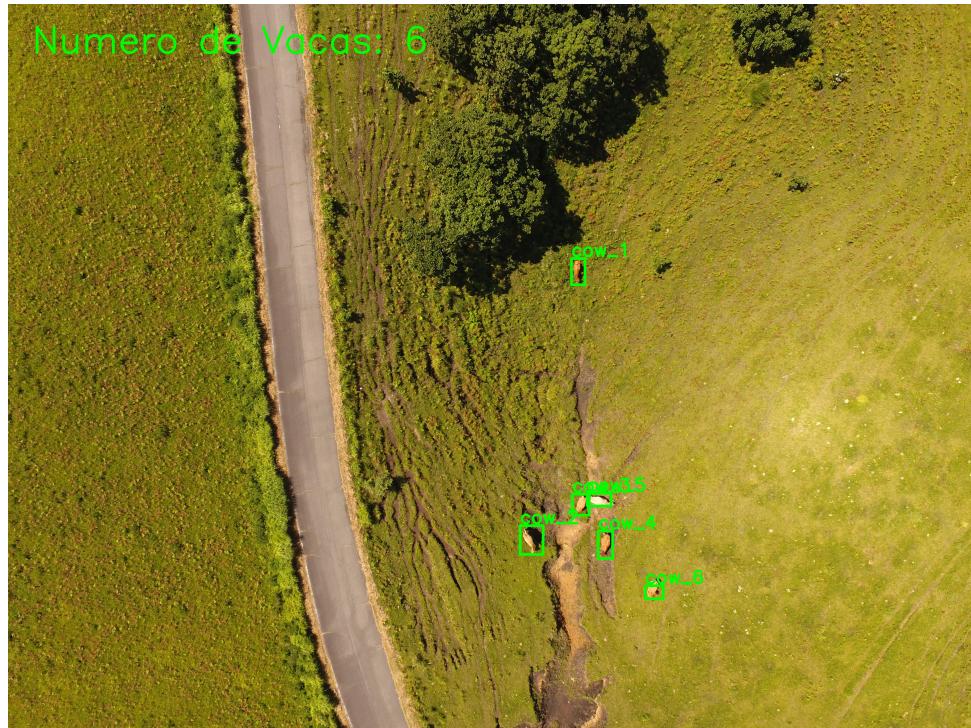


Figura 55 – Exemplo de uma contagem realizada pelo modelo de contagem v3 em uma imagem do conjunto de validação

Fonte: Autoral

Tabela 20 – Métricas de avaliação obtidos pelo modelo de contagem v3

V3	Precisão	Recall	F1 Score	Acurácia	MSE	MAE	DiC	TC
Total	0.986	0.914	0.949	0.902	1.246	0.073	17	68.998
Média	0.992	0.934	0.957	0.927	0.122	0.071	0.447	1.468

## 5.0.4 Comparação entre os modelos

### 5.0.4.1 Comparação entre os modelos de detecção

A tabela 21, apresenta as principais diferenças no treinamento de cada uma das três versões do modelo de detecção. Analisando essa tabela, embora janelas menores produzem uma maior quantidade de imagens, percebe-se que apenas uma pequena fração é realmente útil para o treinamento. Isso justifica o fato do conjunto de dados do modelo v2 ser inferior aos das outras versões. Ainda, percebe-se que o conjunto de dados utilizado na construção do modelo v3 é dez vezes maior que o conjunto de dados utilizado no modelo v1. Isso é consequência da técnica *data augmentation* realizado no tópico 4.2.1.

A tabela 22 apresenta os resultados de avaliação obtido por cada versão de modelo de detecção ao final do treinamento. Com base nesses resultados, concluímos que o modelo de detecção v3 obteve uma melhor performance em comparação com as outras versões. Isso porque apresentou as métricas de precisão, *recall* e mAP altas. Por outro lado, as métricas de erro são, de maneira geral, mais baixas que as outras versões. Portanto,

Tabela 21 – Dimensão do janelamento, quantidades totais de imagens nos conjuntos de treino e teste e o número de épocas utilizado no treinamento para cada versão do modelo de detecção

Modelos de detecção	Dimensão das janelas [px]	Treino	Teste	Épocas
Modelo V1	500x600	1352	338	100
Modelo V2	200x200	1124	280	100
Modelo V3	500x600	13528	3382	150

conclui-se que o modelo de detecção v3 é o mais confiável em suas previsões positivas e é capaz de identificar a maioria dos casos positivos.

Tabela 22 – Métricas de avaliação ao final do treinamento dos modelos de detecção. Métricas de erros avaliados com conjunto dados de teste.

Modelos de detecção	Precisão	recall	mAP50	mAP50-95	box loss	cls loss	dfl loss
Modelo V1	0.974	0.974	0.989	0.894	0.460	0.319	0.870
Modelo V2	0.974	0.987	0.993	0.892	0.479	0.332	0.933
Modelo V3	0.980	0.978	0.990	0.892	0.454	0.282	0.949

#### 5.0.4.2 Comparação entre os modelos de contagem

A robustez do modelo de detecção v3 é evidenciada também no processo de calibração do modelo de contagem. Analisando a tabela 23, percebe-se que o valor mais alto de *Threshold*, juntamente com a maior quantidade de acerto corresponde ao modelo de contagem v3. Um modelo de contagem com um *Threshold* alto, resulta em contagens mais precisas e confiáveis.

Tabela 23 – Resultados da calibração do parâmetro *Threshold* dos modelos de contagem V1, V2 e V3 para um conjunto de 25 imagens

Modelos	Threshold	Imagens corretas
Modelo V1	0.66	16
Modelo V2	0.71	7
Modelo V3	0.71	17

Analizando a tabela 24, onde FP duplicados representa a quantidade de falsos positivos referentes a contagem duplicadas e FP *NoCow* representa a quantidade de instâncias *NoCow* que o modelo contabilizou como *Cow*, e comparando os resultados obtidos pelo modelo de contagem v1 e v2, é possível perceber uma forte relação entre a dimensão das janelas e o número de contagens duplicadas. Quanto menor a janela, maior a probabilidade de uma vaca estar localizada em múltiplas janelas. Outro resultado interessante é que, embora o modelo de contagem v2 contabilizou, exatamente, o mesmo número real de vacas no conjunto de validação, foi o modelo que mais obteve previsões erradas.

Tabela 24 – Resumo dos resultados das contagens obtidos por cada modelo de contagem no conjunto de 47 imagens de validação.

Modelos de contagem	Real	Predita	FP Duplicados	FP <i>NoCow</i>	Predições erradas
Modelo V1	232	209	0	4	31
Modelo V2	232	232	12	18	60
Modelo V3	232	215	3	0	23

A tabela 25 apresenta os resultados da matriz de confusão obtidos com as imagens de validação pelas três versões de modelo de contagem. Nessa tabela, TP e TN refere-se aos valores verdadeiro positivos e verdadeiros negativos, respectivamente. Já os valores FP e FN representam os falso positivo e o falso negativo, respectivamente.

Tabela 25 – Resultados da matriz de confusão obtidos com as imagens de validação pelas três versões de modelo de contagem.

Modelos de contagem	Real	TP	FP	TN	FN
Modelo V1	232	205	4	0	27
Modelo V2	232	202	30	0	30
Modelo V3	232	212	3	0	20

Analizando a tabela 25, é possível calcular uma métrica interessante: a porcentagem de vacas contabilizadas corretamente por cada modelo. O modelo v1, construído a partir do janelamento 500x600 pixels, foi capaz de contar 88.36% das vacas presente em todo conjunto de dados de validação. O modelo v2, construído a partir do janelamento 200x200 pixels, contabilizou uma menor porcentagem 87.07%. O modelo v2 teve um agravante, pois, aproximadamente, 13% das predições realizadas pelo modelo referem-se a falso positivos, contagens de vacas duplicadas e instâncias da classe *NoCow* rotulados como *Cow*. Por fim, o modelo v3 obteve os melhores resultados, contabilizando, aproximadamente, 91.39% das vacas presentes no conjunto de validação. Ainda, das predições realizadas, apenas 1.42% referem-se a contagem de falsos positivos.

A tabela 26 apresenta as métricas de avaliação descritas no tópico 2.6 obtidas pelos modelos de contagem ao final do processo de validação com um conjunto de 47 imagens. Já a tabela 27 apresenta a média desses valores.

Tabela 26 – Resultado total das métricas de avaliação dos modelos de contagem.

Modelos	Precisão	Recall	F1 Score	Acurácia	MSE	MAE	DiC	TC
Modelo V1	0.981	0.884	0.930	0.869	2.280	0.099	23	66.219
Modelo V2	0.871	0.871	0.871	0.771	0.000	0.000	0	235.780
Modelo V3	0.986	0.914	0.949	0.902	1.246	0.073	17	68.998

A escolha do melhor modelo deve ser baseada em um conjunto de métricas. Dessa forma, mesmo apresentando um módulo de diferença de contagem ( $|DiC|$ ) iguais a zero na tabela 26, o modelo v2 não é o mais performático. Isso porque a acurácia e o *F1 Score*

Tabela 27 – Médias das métricas de avaliação dos modelos de contagem.

Modelos	Precisão	<i>Recall</i>	<i>F1 Score</i>	Acurácia	MSE	MAE	DiC	TC
Modelo V1	0.973	0.912	0.932	0.892	0.209	0.108	0.574	1.409
Modelo V2	0.853	0.843	0.828	0.761	0.446	0.236	0.851	5.017
Modelo V3	0.992	0.934	0.957	0.927	0.122	0.071	0.447	1.468

do modelo v2 são inferiores ao modelos v1 e v3. Esses resultados nos mostram que modelo de contagem v2 apresenta contagens menos confiáveis e é incapaz de identificar a maioria dos casos positivos em comparação com os outros modelos. Ainda, quando analisamos o tempo de contagem gasto pelo modelo v2, indicado pela coluna TC, concluímos que ele também é o mais lento.

Dessa forma, o modelo que obteve a melhor resultados no processo de validação foi o modelo de contagem v3. Os valores médios de 92.7% de acurácia e 93.7% *F1 Score* indicados pela tabela 27, mostram que o modelo de contagem v3, construído a partir de um janelamento com dimensão 500x600 pixels e em seguida submetido ao processo de otimização por meio de *data augmentation*, resultou em contagens mais confiáveis e o modelo foi capaz de identificar e contar a maioria das vacas em comparação com os outros modelos.

## 6 Conclusão

O objetivo deste trabalho consistiu em aplicar a arquitetura de rede neural convolucional especializada no reconhecimento de objetos junto à técnica de janelamento, a fim de construir um modelo capaz de reconhecer e contar gados em imagens aéreas com dimensões, relativamente, grandes em diferentes situações de ambiente descontrolado de pastagens de produção extensiva.

A contagem de objetos pequenos em imagens grandes por meio de visão computacional é uma tarefa desafiadora. Isso porque uma parte da informação é perdida logo nas primeiras camadas de uma rede neural convolucional devido as operações de *downsample*. O janelamento das imagens, quando aplicado corretamente, mostrou-se uma técnica promissora para minimizar essas perdas de características dos objetos de interesse nas imagens.

A fim de escolher um tamanho de janela adequado para a construção do modelo de contagem final, neste trabalho foram construídos, inicialmente, duas versões de modelo de contagem de gado a partir diferentes dimensões de janelas. A primeira versão foi construído a partir de janelas de 500x600 pixels e a segunda versão foi construído a partir de janelas de 200x200 pixels.

Com base nas tabelas 26 e 27, concluiu-se que a janelas de 500x600 pixels resultaram em um modelo de contagem mais preciso e acurado que a segunda versão do modelo, mesmo apresentando uma diferença de contagem maior. Isso pode ser justificado devido ao tamanho da janela de 500x600 pixels ser próximo ao tamanho da dimensão da rede neural convolucional utilizada para o modelo de detecção. Neste projeto usou-se para o para o modelo de detecção a arquitetura Yolo V8 nano cujo a dimensão é de 640x640 pixels. Dessa forma, houve pouca distorção nas imagens durante as transformações de redimensionamento realizados pela rede neural. Fazendo com que as características dos objetos de interesse fossem preservadas e aprendidas pelo agente inteligente.

Outra justificativa para a melhor performance da primeira versão de modelo de contagem em comparação com a segunda está evidenciada na tabela 24, onde vemos que a segunda versão do modelo de contagem foi fortemente afetada pelo número de contagem duplicadas. Enquanto o primeiro modelo não apresentou nenhuma ocorrência dessa natureza. Isso tem como justificativa o aumento da probabilidade de uma vaca estar localizada em múltiplas janelas quando aplica-se um janelamento menor.

Com o tamanho da janela 500x600 pixels escolhido, foram aplicados técnicas de otimização para melhorar a performance do modelo. A primeira consistiu em enriquecer o conjunto de dados com imagens que poderiam confundir o modelo. Dessa forma, foram

adicionada mais imagens com vegetação, sombras, estradas, deformações do solo, pessoas, entre outros. Em seguida, a fim de diversificar o conjunto de dados, forma aplicadas transformações de *data augmentation* no conjunto de dados. Por fim, aumentou-se o número de épocas para 150 no treinamento do modelo de detecção, resultando em uma versão final mais precisa e acurada em relação a primeira versão de modelo de detecção. A tabela 27 evidencia isso.

O terceiro modelo de contagem também obteve melhores métricas de validação quando avaliado com o conjunto de 47 novas imagens. O modelo final apresentou uma acurácia média de 92.7% e contou corretamente 91.4% das vacas presente e todo o conjunto de dados.

Os erros cometido por esse modelo são referentes a apenas 3 contagens duplicadas. O que representa, aproximadamente, 1.49% das predições. Por outro lado, em nenhum momento o modelo detectou e contabilizou instancias que não são os objetos de interesse como vacas. Dessa forma, a vegetação, as sombras, as deformações do solo e outros ruídos característico de um ambiente descontrolado não afetaram negativamente na contagem realizada pelo terceiro modelo.

Os resultados obtido neste trabalho mostraram-se satisfatório. Entretanto, como o conjunto de dados utilizado possuem imagens de rebanhos de gado em uma mesma pastagem, é interessante repetir o experimento para um conjunto de imagens mais diversa, com imagens aéreas de rebanhos de diversas raças e coletados em diferentes pastagens e em diferentes estações do ano. Por fim, aplicar o novo modelo de contagem em um mosaico de imagens construído a partir do mapeamento usando drones, a fim de avaliar a performance do modelo em imagens de dimensões mais elevadas que as imagens utilizadas neste estudo.

## Referências

- AGGARWAL, C. C. *Neural Networks and Deep Learning*. Springer International Publishing, 2018. Disponível em: <<https://doi.org/10.1007/978-3-319-94463-0>>. Citado 11 vezes nas páginas 23, 24, 101, 104, 105, 106, 107, 108, 112, 113 e 114.
- AGHDAM, H. H.; HERAVI, E. J. *Guide to convolutional neural networks: A practical application to traffic-sign detection and classification*. Cham, Switzerland: Springer International Publishing, 2018. Citado 5 vezes nas páginas 22, 23, 31, 55 e 57.
- AGRISHOW, R. *Produção agrícola conectada com o universo digital: entenda a tendência da Agricultura 4.0*. 2016. <<https://digital.agrishow.com.br/tecnologia/producao-agricola-conectada-com-o-universo-digital-entenda-tendencia-da-agricultura-40>>. Accessed: 2022-4-23. Citado na página 17.
- ALONSO, J.; VILLA, A.; BAHAMONDE, A. Improved estimation of bovine weight trajectories using support vector machine classification. *Computers and Electronics in Agriculture*, v. 110, p. 36–41, 2015. ISSN 0168-1699. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0168169914002488>>. Citado na página 16.
- AMARA, J.; BOUAZIZ, B.; ALGERGAWY, A. A deep learning-based approach for banana leaf diseases classification. In: . [S.l.: s.n.], 2017. p. 79–88. Citado na página 16.
- ANDREW, W.; GREATWOOD, C.; BURGHARDT, T. Visual localisation and individual identification of holstein friesian cattle via deep learning. In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. [S.l.: s.n.], 2017. p. 2850–2859. Citado na página 18.
- ANTONUCCI, F. et al. A review on blockchain applications in the agri-food sector. *Journal of the Science of Food and Agriculture*, v. 99, n. 14, p. 6129–6138, 2019. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/jsfa.9912>>. Citado na página 15.
- AVUÇLU, E.; BAŞÇIFTÇI, F. New approaches to determine age and gender in image processing techniques using multilayer perceptron neural network. *Applied Soft Computing*, Elsevier BV, v. 70, p. 157–168, set. 2018. Disponível em: <<https://doi.org/10.1016/j.asoc.2018.05.033>>. Citado na página 114.
- BARBEDO, J. et al. Counting cattle in uav images—dealing with clustered animals and animal/background contrast changes. *Sensors*, v. 20, p. 2126, 04 2020. Citado 2 vezes nas páginas 17 e 18.
- BARBEDO, J. G. A. et al. A study on the detection of cattle in uav images using deep learning. *Sensors*, v. 19, n. 24, 2019. ISSN 1424-8220. Disponível em: <<https://www.mdpi.com/1424-8220/19/24/5436>>. Citado na página 40.
- BRAHIMI, M.; KAMEL, B.; MOUSSAOUI, A. Deep learning for tomato diseases: Classification and symptoms visualization. *Applied Artificial Intelligence*, v. 31, p. 1–17, 05 2017. Citado na página 16.

- BUSLAEV, A. et al. Albumentations: Fast and flexible image augmentations. *Information*, v. 11, n. 2, 2020. ISSN 2078-2489. Disponível em: <<https://www.mdpi.com/2078-2489/11/2/125>>. Citado na página 60.
- CHEN, J. et al. Detection of rice plant diseases based on deep transfer learning. *J. Sci. Food Agric.*, v. 100, n. 7, p. 3246–3256, 2020. Citado na página 16.
- CHEN, S. W. et al. Counting apples and oranges with deep learning: A data-driven approach. *IEEE Robotics and Automation Letters*, v. 2, n. 2, p. 781–788, 2017. Citado na página 16.
- CIRESAN, D. et al. Flexible, high performance convolutional neural networks for image classification. In: . [S.l.: s.n.], 2011. p. 1237–1242. Citado na página 24.
- CUN, Y. L. et al. Handwritten digit recognition: Applications of neural net chips and automatic learning. In: *Neurocomputing*. Springer Berlin Heidelberg, 1990. p. 303–318. Disponível em: <[https://doi.org/10.1007/978-3-642-76153-9\\_35](https://doi.org/10.1007/978-3-642-76153-9_35)>. Citado na página 21.
- EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, v. 88, n. 2, p. 303–338, jun. 2010. Citado na página 29.
- FENG, J.; LU, S. Performance analysis of various activation functions in artificial neural networks. *Journal of Physics: Conference Series*, IOP Publishing, v. 1237, p. 022030, jun. 2019. Disponível em: <<https://doi.org/10.1088/1742-6596/1237/2/022030>>. Citado 5 vezes nas páginas 104, 105, 107, 108 e 109.
- FRANKE, G.; MUCHER, S. *Video files for cow detection using yolov3*. Harvard Dataverse, 2021. Disponível em: <<https://doi.org/10.7910/DVN/YFDJRO>>. Citado 2 vezes nas páginas 37 e 38.
- FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, Springer Science and Business Media LLC, v. 36, n. 4, p. 193–202, abr. 1980. Disponível em: <<https://doi.org/10.1007/bf00344251>>. Citado na página 21.
- GAO, B.; PAVEL, L. On the properties of the softmax function with application in game theory and reinforcement learning. 04 2017. Citado na página 103.
- GIRSHICK, R. B. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. Disponível em: <<http://arxiv.org/abs/1504.08083>>. Citado na página 18.
- GIRSHICK, R. B. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. Disponível em: <<http://arxiv.org/abs/1311.2524>>. Citado na página 18.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <<http://www.deeplearningbook.org>>. Citado 8 vezes nas páginas 22, 80, 101, 102, 103, 110, 111 e 112.
- GRAVES, A. et al. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In: . [S.l.: s.n.], 2006. v. 2006, p. 369–376. Citado na página 21.

- GU SHUANG BAI, L. K. W. A review on 2d instance segmentation based on deep neural networks. *Image and Vision Computing*, Elsevier BV, v. 120, p. 157–168, 2022. Disponível em: <<https://doi.org/10.1016/j.imavis.2022.104401>>. Citado 2 vezes nas páginas 33 e 34.
- HAN, L.; TAO, P.; MARTIN, R. R. Livestock detection in aerial images using a fully convolutional network. *Comput. Vis. Media (Beijing)*, v. 5, n. 2, p. 221–228, 2019. Citado na página 17.
- HAYKIN, S. *Redes Neurais: Princípios e Prática*. [S.I.]: Bookman, 2003. ASIN B0006AXUII. Citado 2 vezes nas páginas 21 e 102.
- HAYKIN, S. *Neural network and machine learning*. [S.I.]: Pearson, 2009. ASIN B0006AXUII. Citado 5 vezes nas páginas 100, 101, 110, 112 e 114.
- HE, K. et al. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. Disponível em: <<http://arxiv.org/abs/1703.06870>>. Citado na página 18.
- HE, K. et al. Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. Disponível em: <<https://doi.org/10.1109/cvpr.2016.90>>. Citado na página 22.
- HOCHREITER, S. Untersuchungen zu dynamischen neuronalen netzen. 04 1991. Citado na página 21.
- HÄNI, N.; ROY, P.; ISLER, V. A comparative study of fruit detection and counting methods for yield mapping in apple orchards. *Journal of Field Robotics*, v. 37, n. 2, p. 263–282, 2020. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21902>>. Citado na página 16.
- IVAKHNENKO, A. G. The group method of data handling—a rival of the method of stochastic approximation. *Soviet Automatic Control*, v. 1, n. 3, p. 43–55, 1968. Citado na página 21.
- JOCHER, G.; CHAURASIA, A.; QIU, J. *YOLO by Ultralytics*. 2023. Disponível em: <<https://github.com/ultralytics/ultralytics>>. Citado 2 vezes nas páginas 26 e 27.
- KAMILARIS, A.; PRENAFETA-BOLDU, F. X. Deep learning in agriculture: A survey. *CoRR*, abs/1807.11809, 2018. Disponível em: <<http://arxiv.org/abs/1807.11809>>. Citado na página 16.
- KAWASAKI, Y. et al. Basic study of automated diagnosis of viral plant diseases using convolutional neural networks. In: BEBIS, G. et al. (Ed.). *Advances in Visual Computing*. Cham: Springer International Publishing, 2015. p. 638–645. ISBN 978-3-319-27863-6. Citado na página 16.
- KOYUN, O. C. et al. Focus-and-detect: A small object detection framework for aerial images. *Signal Processing: Image Communication*, Elsevier BV, v. 104, p. 116675, may 2022. Disponível em: <<https://doi.org/10.1016%2Fj.image.2022.116675>>. Citado na página 36.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. Imagenet classification with deep convolutional neural networks. In: . [S.I.: s.n.], 2012. p. 1097–1105. Citado na página 22.

- KUSSUL, N. et al. Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, v. 14, n. 5, p. 778–782, 2017. Citado na página 16.
- KUWATA, K.; SHIBASAKI, R. Estimating crop yields with deep learning and remotely sensed data. In: *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. [S.l.: s.n.], 2015. p. 858–861. Citado na página 16.
- LECUN, Y. et al. Handwritten digit recognition with a back-propagation network. In: TOURETZKY, D. (Ed.). *Advances in Neural Information Processing Systems*. Morgan-Kaufmann, 1990. v. 2. Disponível em: <<https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf>>. Citado na página 21.
- LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998. Citado 2 vezes nas páginas 21 e 22.
- LECUN, Y. et al. Learning algorithms for classification: A comparison on handwritten digit recognition. In: \_\_\_\_\_. *Neural networks*. [S.l.]: World Scientific, 1995. p. 261–276. Citado na página 35.
- LEE JINSU E BANG, J. e. Y. S.-I. Detecção de objeto com janela deslizante em imagens incluindo vários objetos semelhantes. In: *2017 Conferência Internacional sobre Convergência de Tecnologia da Informação e Comunicação (ICTC)*. [S.l.]: SBC, 2017. p. 803–806. ISSN 0000-0000. Citado na página 48.
- LIN, T.-Y. et al. *Microsoft COCO: Common Objects in Context*. 2015. Citado 3 vezes nas páginas 26, 27 e 30.
- LINNAINMAA, S. Taylor expansion of the accumulated rounding error. *BIT*, Springer Science and Business Media LLC, v. 16, n. 2, p. 146–160, jun. 1976. Disponível em: <<https://doi.org/10.1007/bf01931367>>. Citado na página 21.
- LU, J. et al. An in-field automatic wheat disease diagnosis system. *Computers and Electronics in Agriculture*, v. 142, p. 369–379, 2017. ISSN 0168-1699. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0168169917305999>>. Citado na página 16.
- MA, X. et al. Deep learning method for makeup style transfer: A survey. *Cognitive Robotics*, v. 1, p. 182–187, 2021. ISSN 2667-2413. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S266724132100015X>>. Citado na página 53.
- MADIAJAGAN, M.; RAJ, S. S. Parallel computing, graphics processing unit (GPU) and new hardware for deep learning in computational intelligence research. In: *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. Elsevier, 2019. p. 1–15. Disponível em: <<https://doi.org/10.1016/b978-0-12-816718-2.00008-7>>. Citado na página 54.
- MASSRUHÁ, S. M. F. S.; LEITE, M. A. de A. Agro 4.0 - rumo à agricultura digital. *Artigo em anais de congresso (CNPTIA)*, Elsevier BV, v. 139, p. 364–387, mar. 2017. Disponível em: <<http://www.alice.cnptia.embrapa.br/alice/handle/doc/1073150>>. Citado na página 15.

- MASSRUHÁ, S. M. F. S. et al. Os novos desafios e oportunidades das tecnologias da informação e da comunicação na agricultura (agrotic). *Capítulo em livro científico (CNPTIA)*, Elsevier BV, v. 139, p. 364–387, mar. 2014. Disponível em: <<http://www.alice.cnptia.embrapa.br/alice/handle/doc/1010685>>. Citado na página 15.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, Springer Science and Business Media LLC, v. 5, n. 4, p. 115–133, dez. 1943. Disponível em: <<https://doi.org/10.1007/bf02478259>>. Citado na página 20.
- MINSKY, M.; PAPERT, S. *Perceptrons: An introduction to computational geometry*. Londres, England: MIT Press, 1969. Citado na página 20.
- MOHANTY, S. P.; HUGHES, D. P.; SALATHÉ, M. Using deep learning for image-based plant disease detection. *Front. Plant Sci.*, v. 7, p. 1419, 2016. Citado na página 16.
- MORALES, I. R. et al. Early warning in egg production curves from commercial hens: A svm approach. *Computers and Electronics in Agriculture*, v. 121, p. 169–179, 2016. ISSN 0168-1699. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0168169915003919>>. Citado na página 16.
- NASR, G.; BADR, E.; JOUN, C. Cross entropy error function in neural networks: Forecasting gasoline demand. In: . [S.l.: s.n.], 2002. p. 381–384. Citado na página 110.
- ONU. World population prospects 2019, online edition. rev. 1. *Department of Economic and Social Affairs, Population Division* (2019), 2019. Disponível em: <<https://brasil.un.org/pt-br/83427-populacao-mundial-deve-chegar-97-bilhoes-de-pessoas-em-2050-diz-relatorio-da-onu>>. Citado na página 15.
- ONWUDE, D. I. et al. Combination of computer vision and backscattering imaging for predicting the moisture content and colour changes of sweet potato (*ipomoea batatas* l.) during drying. *Computers and Electronics in Agriculture*, v. 150, p. 178–187, 2018. ISSN 0168-1699. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0168169917312486>>. Citado na página 16.
- PANDEY, C. et al. Chapter 1 - smart agriculture: Technological advancements on agriculture—a systematical review. In: POONIA, R. C.; SINGH, V.; NAYAK, S. R. (Ed.). *Deep Learning for Sustainable Agriculture*. Academic Press, 2022, (Cognitive Data Science in Sustainable Computing). p. 1–56. ISBN 978-0-323-85214-2. Disponível em: <<https://www.sciencedirect.com/science/article/pii/B9780323852142000021>>. Citado 2 vezes nas páginas 15 e 17.
- PEREZ, L.; WANG, J. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017. Disponível em: <<http://arxiv.org/abs/1712.04621>>. Citado na página 35.
- REDMON, J. et al. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. Disponível em: <<http://arxiv.org/abs/1506.02640>>. Citado 3 vezes nas páginas 25, 26 e 28.

- REDMON, J.; FARHADI, A. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016. Disponível em: <<http://arxiv.org/abs/1612.08242>>. Citado na página 17.
- 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 na página 18.
- RIBEIRO, N. G. V.; GUEDES, G. B.; BARBIERI, T. T. S. Aplicação de algoritmos de Visão Computacional na contagem de gado por meio de processamento de imagens aéreas. *Revista eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação*, v. 1, n. 10, jul. 2019. Disponível em: <<https://doi.org/10.5281/zenodo.3338098>>. Citado na página 40.
- ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, American Psychological Association (APA), v. 65, n. 6, p. 386–408, 1958. Disponível em: <<https://doi.org/10.1037/h0042519>>. Citado 2 vezes nas páginas 20 e 100.
- ROSENBLATT, F. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. [S.l.]: Spartan Books, 1962. ASIN B0006AXUII. Citado na página 20.
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Springer Science and Business Media LLC, v. 323, n. 6088, p. 533–536, out. 1986. Disponível em: <<https://doi.org/10.1038/323533a0>>. Citado na página 21.
- RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015. Citado na página 27.
- SANTO, J. do E.; FILHO, J. de O. Detecção e contagem de bovinos em imagens aéreas utilizando visão computacional. In: *Anais da XX Escola Regional de Computação Bahia, Alagoas e Sergipe*. Porto Alegre, RS, Brasil: SBC, 2020. p. 71–78. ISSN 0000-0000. Disponível em: <<https://sol.sbc.org.br/index.php/erbase/article/view/15462>>. Citado na página 40.
- SANTONI, M. M. et al. Cattle race classification using gray level co-occurrence matrix convolutional neural networks. *Procedia Computer Science*, v. 59, p. 493–502, 2015. ISSN 1877-0509. International Conference on Computer Science and Computational Intelligence (ICCSCI 2015). Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1877050915020542>>. Citado na página 16.
- SANTOS PABLO; RAVEANE WILLIAM, P. V. V. G. A. A. C. Uavs applied to the counting and monitoring of animals. In: . [S.l.]: Springer, 2014. v. 291, p. 71–80. ISBN 978-3-319-07595-2 (Print), 978-3-319-07596-9 (Online). Citado na página 17.
- SEPO, L. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. Dissertação (Mestrado) — University of Helsinki, 1970. Citado na página 21.
- SHAO REI KAWAKAMI, R. Y. S. Y. H. K. T. N. W. *Cattle detection and counting in UAV images based on convolutional neural networks*. Naemura Lab. The University of

- Tokyo, 2020. Disponível em: <[https://doi.org/10.1080/01431161.2019.1624858\(2020\)](https://doi.org/10.1080/01431161.2019.1624858(2020))>. Citado 10 vezes nas páginas 4, 5, 36, 37, 40, 44, 45, 51, 59 e 61.
- SHAO, W. et al. Cattle detection and counting in UAV images based on convolutional neural networks. *Int. J. Remote Sens.*, v. 41, n. 1, p. 31–52, 2020. Citado na página 17.
- SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. Citado na página 22.
- SLADOJEVIC, S. et al. Deep neural networks based recognition of plant diseases by leaf image classification. *Comput. Intell. Neurosci.*, v. 2016, p. 3289801, 2016. Citado na página 16.
- SOARES, V. et al. Cattle counting in the wild with geolocated aerial images in large pasture areas. *Computers and Electronics in Agriculture*, v. 189, p. 106354, 2021. ISSN 0168-1699. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0168169921003719>>. Citado na página 17.
- SZEGEDY, C. et al. Going deeper with convolutions. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. p. 1–9. Citado na página 22.
- TERVEN, J.; CORDOVA-ESPARZA, D. *A Comprehensive Review of YOLO: From YOLOv1 and Beyond*. 2023. Citado na página 25.
- TING, K. M. Confusion matrix. In: \_\_\_\_\_. *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010. p. 209–209. ISBN 978-0-387-30164-8. Disponível em: <[https://doi.org/10.1007/978-0-387-30164-8\\_157](https://doi.org/10.1007/978-0-387-30164-8_157)>. Citado 2 vezes nas páginas 31 e 34.
- TZUTALIN. *demo3.jpg*. 2018. GitHub Repository. Disponível em: <<https://github.com/heartexlabs/labelImg>> (Acessado em: 4 de julho de 2023). Citado na página 29.
- WEBER, F. de L. et al. Recognition of pantaneira cattle breed using computer vision and convolutional neural networks. *Computers and Electronics in Agriculture*, v. 175, p. 105548, 2020. ISSN 0168-1699. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0168169919321490>>. Citado na página 17.
- XU, B. et al. Automated cattle counting using mask r-cnn in quadcopter vision system. *Computers and Electronics in Agriculture*, v. 171, p. 105300, 2020. ISSN 0168-1699. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0168169919320149>>. Citado na página 18.
- XU, B. et al. Automated cattle counting using mask r-cnn in quadcopter vision system. *Computers and Electronics in Agriculture*, v. 171, p. 105300, 2020. ISSN 0168-1699. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0168169919320149>>. Citado na página 41.
- YAMASHITA, R. et al. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, Springer Science and Business Media LLC, v. 9, n. 4, p. 611–629, jun. 2018. Disponível em: <<https://doi.org/10.1007/s13244-018-0639-9>>. Citado 2 vezes nas páginas 23 e 24.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014. p. 818–833. Disponível em: <[https://doi.org/10.1007/978-3-319-10590-1\\_53](https://doi.org/10.1007/978-3-319-10590-1_53)>. Citado na página 22.

# APÊNDICE A – Conceitos fundamentais de *deep learning*

## A.1 Rede neural de uma camada

A rede neural de uma camada, também conhecida como *Perceptron* é a precursora das poderosas arquiteturas de *Deep Learning* tais como: as redes neurais convolucionais. Como citado no tópico 2.1 e descritas na seção 2.2, o *Perceptron* foi introduzido por Frank Rosenblatt em seu artigo de 1958 ([ROSENBLATT, 1958](#)) e consiste em uma modelagem matemática de um neurônio biológico.

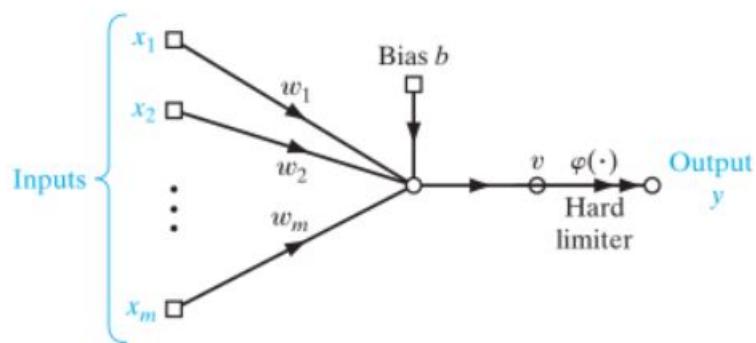


Figura 56 – Arquitetura de um Perceptron.

Fonte: ([HAYKIN, 2009](#))

Como podemos observar na figura 56, os sinais de entrada são representados por um vetor  $x$  que pode carregar consigo características de pixels de uma imagem, que por sua vez, carregam características de um objeto. Em seguida, os elementos do vetor de entrada são multiplicados por pesos sinápticos tensoriais representados por  $w$ . Quando multiplicados, os sinais de entrada são transmitidos analogamente às sinapses biológicas. Adiante, os sinais transmitidos são juntados no nó somador por meio da equação A.1 que, por sua vez, são ajustados por um Bias, representados por  $b$ , gerando o sinal  $v$ . Esse Bias possui um valor constante e pode ser entendido como um viés que permite um grau de liberdade maior à equação. Por fim, o sinal  $v$  passa por uma função de ativação  $\varphi(\cdot)$  e gera a classificação representada por  $y$  ([HAYKIN, 2009](#)). As funções de ativação serão abordadas com mais detalhes na seção A.3.

$$h = \sum_{i=1}^n x_i w_i + b \quad (\text{A.1})$$

Essa modelagem consiste em um combinador linear seguido por um limitador rígido, também chamado de função de ativação, que nesse caso está executando a função sinal  $\text{sgn}$ , conforme ilustrado na figura 57. O nó somador do modelo neural calcula uma combinação linear das entradas aplicadas às suas sinapses, bem como incorpora um viés aplicado. A soma resultante é aplicada a uma função de ativação. Assim, de acordo com a figura 57, o neurônio produz uma saída igual a 1 se a entrada da função de ativação é positiva, e -1 se for negativa (HAYKIN, 2009).

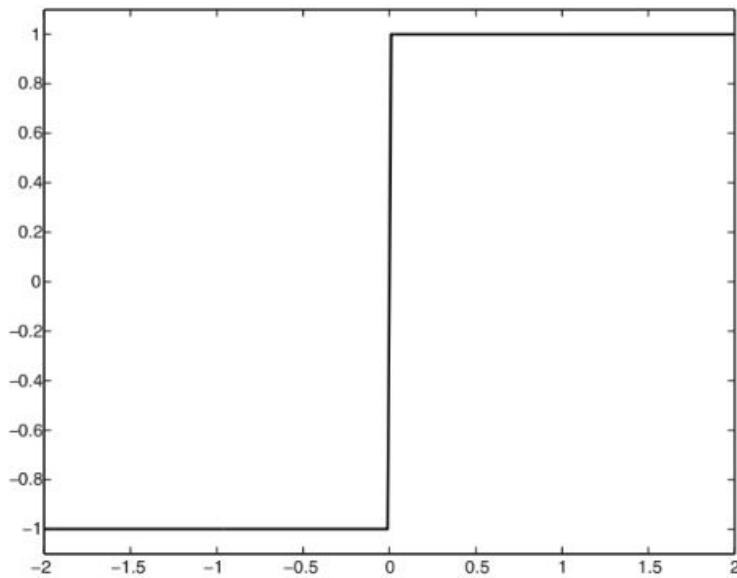


Figura 57 – Curva característica da função sinal.

Fonte:(AGGARWAL, 2018)

## A.2 Rede neural de múltiplas camadas

A rede neural de múltiplas camadas, também chamada de *multilayer perceptron*, é uma arquitetura constituída de pelo menos três nós: os nós de entrada, os nós das camadas ocultas e os nós de saída. Nesse tipo de arquitetura, cada nó é um neurônio que utiliza como função de ativação uma operação não linear, exceto pelo o nó de entrada. Esse tipo de função de ativação faz com que as redes neurais de múltiplas camadas sejam capazes de modelar problemas de natureza não linear, diferentemente da arquitetura simples do perceptron apresentado na seção A (GOODFELLOW; BENGIO; COURVILLE, 2016). Ainda, só é possível a modelagem de agentes inteligentes utilizando o *multilayer perceptron* devido a técnica de otimização do algoritmo *backpropagation* que será abordado na seção A.5.2.

A figura 58 exibe um exemplo de arquitetura de rede neural de múltiplas camadas. Nessa arquitetura em questão está desenvolvida para modelagem de problemas de classi-

ficação categórica com três classes de saída. Ainda, utiliza a operação não linear *softmax* como função de ativação. A função *softmax* será descrita na seção A.3.

A camada de entrada  $x$ , similar ao perceptron da seção A, são vetores que carregam características de um objeto que se deseja modelar. Essa camada de entrada define os neurônios.

As camadas intermediárias são as camadas ocultas. Essas camadas ocultas nada mais são que neurônios que propagam os sinais de entrada com maior ou menor intensidade de acordo com a resposta a função de ativação de natureza não linear. O sinal propagado por um neurônio é utilizado como o sinal de entrada para outras camadas de neurônios mais profundas. Quanto maior o número de camadas ocultas, mais poderosa é a arquitetura da rede neural. Entretanto, requer maiores recursos computacionais. Esse tipo de arquitetura com o número elevado de camadas ocultas são chamadas de redes neurais profundas (HAYKIN, 2003). A arquitetura YOLO e suas evoluções que serão abordadas na seção 2.3.2 são exemplos de redes neurais profundas.

A camada de saída, representada pelo vetor  $y$  na figura 58, retorna um valor numérico e um erro associado. Durante a aprendizagem do modelo, por meio do gradiente descendente, os pesos  $w$  de cada nó são atualizados de maneira iterativa diversas vezes, a fim de minimizar a função de perda. O sentido dessa otimização ocorre partindo da camada de saída até a camada de entrada, denominando-se assim *backpropagation* (GOODFELLOW; BENGIO; COURVILLE, 2016). Esses conceitos de otimização serão abordados na seção A.4.2.

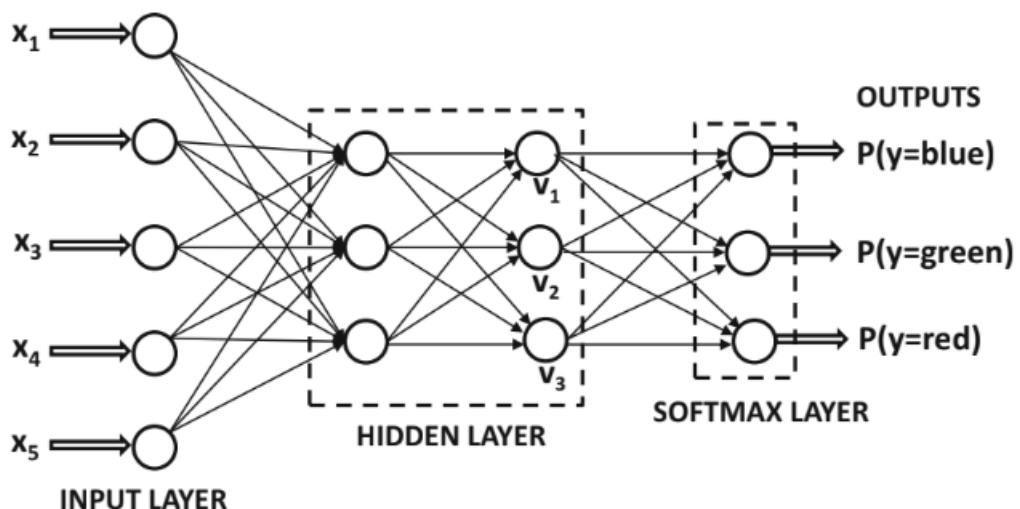


Figura 58 – Arquitetura de um multilayer perceptron.

Fonte: (GOODFELLOW; BENGIO; COURVILLE, 2016)

## A.3 Funções de ativação

A função de ativação em uma rede neural artificial tem como objetivo definir se uma determinada característica de entrada será ou não transferida para camadas mais profundas da rede, normalmente, atribuindo valores à saída que variam entre 0 e 1 ou entre -1 e 1. Devido a esse comportamento, a função de ativação também é conhecida como função de transferência. Ainda, a função de ativação, normalmente, adiciona um componente não linear à arquitetura da rede, aumentando a performance do aprendizado de padrões mais complexos (GOODFELLOW; BENGIO; COURVILLE, 2016).

Cada função de ativação tem suas vantagens e desvantagens. Desse modo, é importante conhecer as particularidades de cada uma e escolher a função de ativação mais adequada à arquitetura da rede que está sendo desenvolvida. A seguir é demonstrado uma descrição das funções comumente utilizadas como função de ativação em redes neurais.

### A.3.1 Softmax

A função softmax retorna na sua saída uma distribuição de probabilidade para cada classe envolvida no problema (GAO; PAVEL, 2017). Isso faz com que a função softmax seja uma das funções de ativação mais utilizadas em problemas de classificação multiclasse. Essa função de ativação está presente na arquitetura YOLO que é descrita na seção 2.3.2.

A função softmax e sua derivada é dada pela equação A.2 e pela equação A.3 respectivamente. Onde  $x_i$  representa a probabilidade da classe naquele índice e  $x_j$  representa a probabilidade de todas as demais classes.

$$S_i(x_i) := \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)} \quad (\text{A.2})$$

$$S'_i(x_i) = S_i(\delta_{ji} - S_j) \quad (\text{A.3})$$

### A.3.2 Sigmóide

A função Sigmóide tem na sua saída valores variando entre 0 e 1 de acordo com a equação A.4. A derivada é dada pela equação A.5. Nas figuras 59 e 60 são representados a curva característica e a derivada da função sigmóide respectivamente.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (\text{A.4})$$

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} \quad (\text{A.5})$$

No contexto de redes neurais, como sua curva possui uma natureza não linear e varia entre 0 e 1, a função sigmóide é utilizada em problemas de classificação binários onde temos que prever a probabilidade de uma determinada saída, já que a probabilidade possui o mesmo intervalo. Entretanto, a função sigmóide possui um risco de saturação de aprendizagem. Ou seja, redução da capacidade de otimização da função de perda causado pelo fenômeno conhecido como “*vanishing gradients*” (dissipação do gradiente). A dissipação do gradiente acontece pois a derivada tende a zero fazendo com que a saída responda muito menos as pequenas variações na entrada ([FENG; LU, 2019](#)).

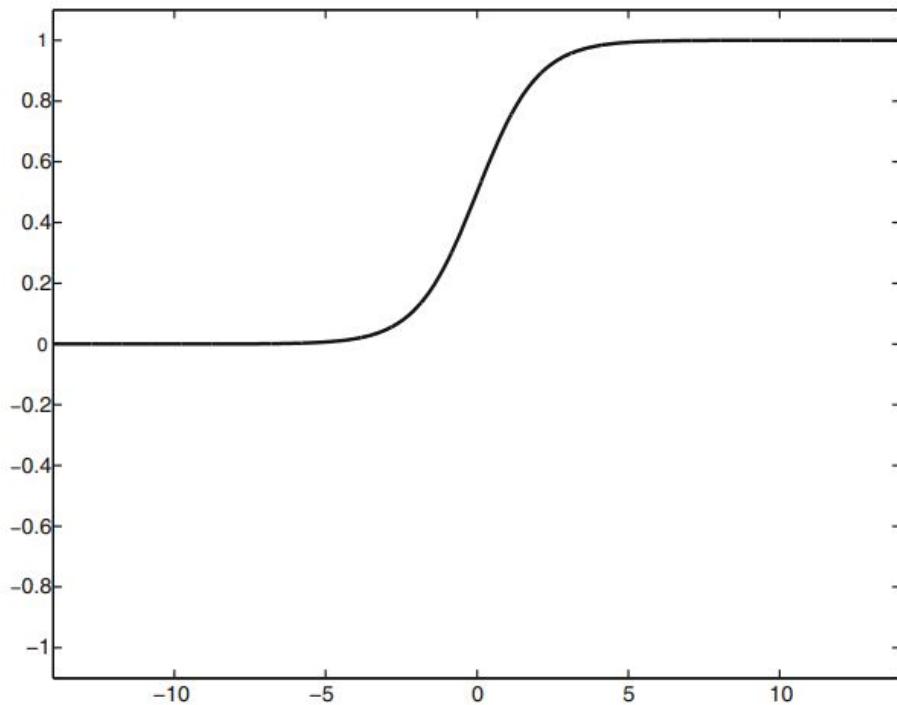


Figura 59 – Curva característica da função sigmóide.

Fonte: ([AGGARWAL, 2018](#))

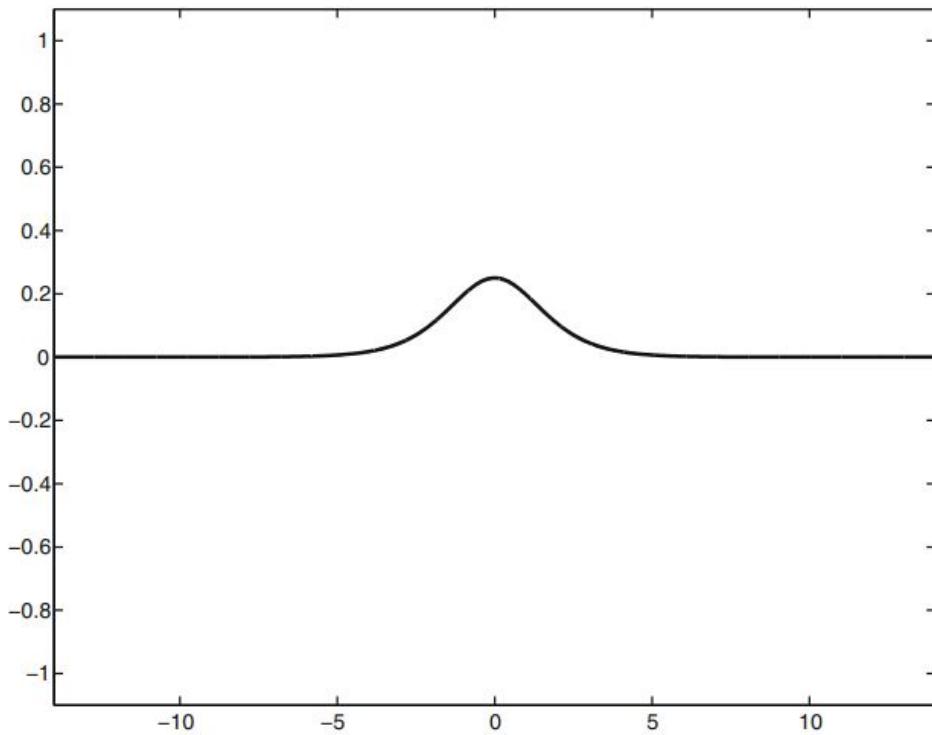


Figura 60 – Curva característica da derivada da função sigmóide.

Fonte: ([AGGARWAL, 2018](#))

### A.3.3 Tangente Hiperbólica

A tangente hiperbólica é dada pela equação [A.6](#) e sua derivada é dada pela equação [A.7](#)

$$\tanh(x) = \frac{2}{1 + e^{-2x}} = 2\text{sigmoide}(2x) - 1 \quad (\text{A.6})$$

$$\tanh'(x) = \frac{4e^{-2x}}{(1 + e^{-2x})^2} \quad (\text{A.7})$$

Como podemos observar na equação [A.6](#) a função da tangente hiperbólica deriva da função sigmóide. Ao olharmos as figuras [61](#) e [62](#), que representam a sua curva característica e da sua derivada respectivamente, vemos que a função hiperbólica é centrada em zero e possui um intervalo que varia entre -1 e 1. No contexto de redes neurais, esse comportamento torna os dados mais centralizados facilitando o treinamento do algoritmo. ([FENG; LU, 2019](#))

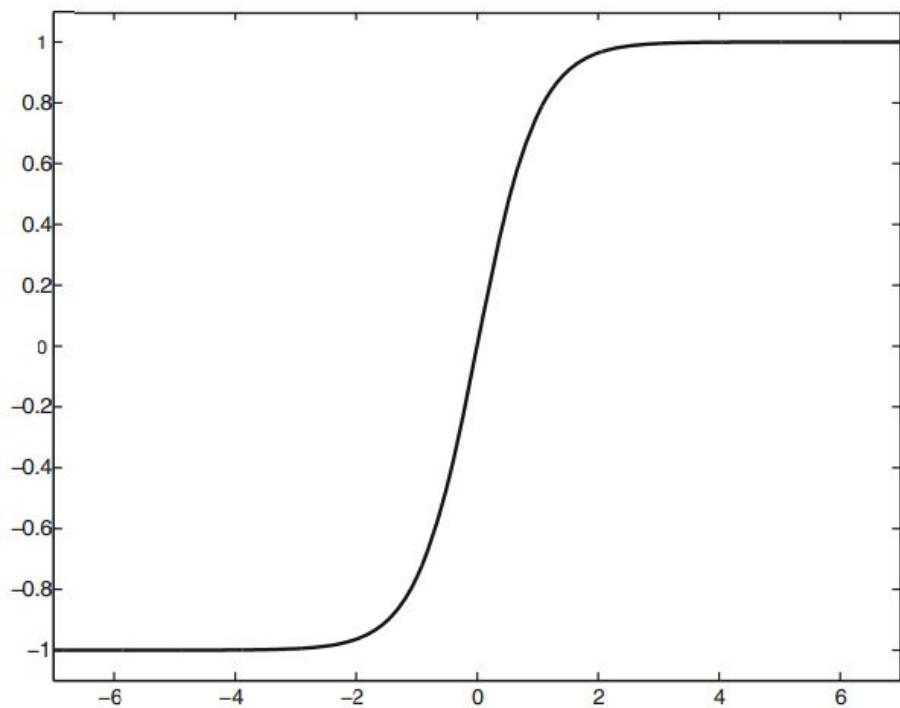


Figura 61 – Curva característica da função tangente hiperbólica.

Fonte: ([AGGARWAL, 2018](#))

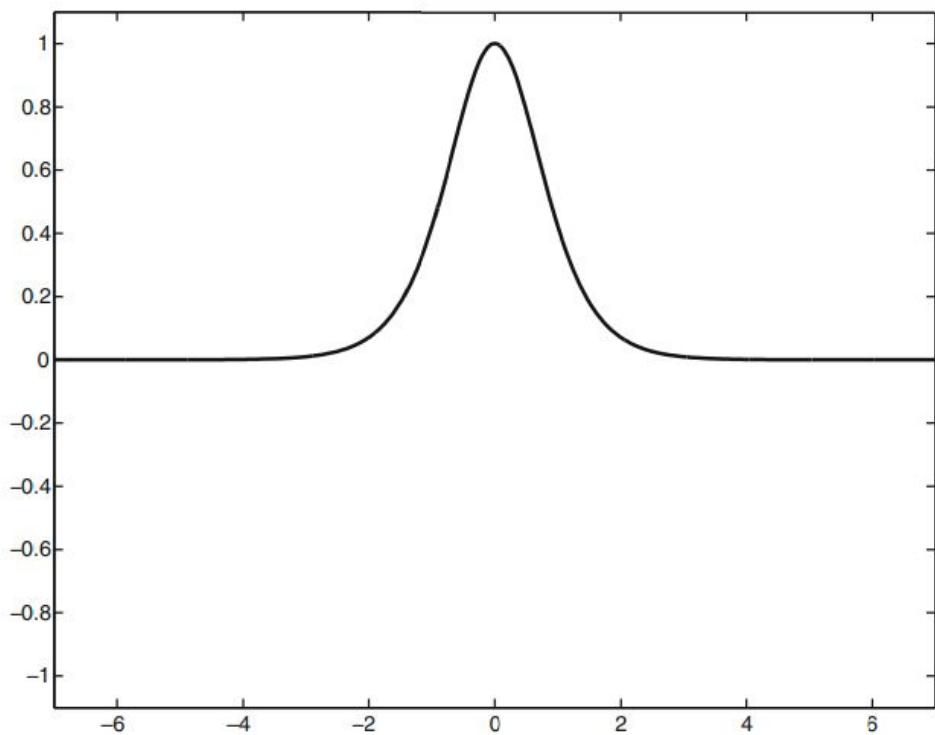


Figura 62 – Curva característica da derivada da função tangente hiperbólica.

Fonte: ([AGGARWAL, 2018](#))

### A.3.4 ReLU

A função ReLU (*rectified linear unit*, ou unidade linear retificada) é dada pela equação A.8 e sua derivada é dada pela equação A.9. As figuras 63 e 64 representam a sua curva característica e da sua derivada, respectivamente.

$$f(x_i) = \max(0, x_i) = \begin{cases} x_i, & x_i > 0 \\ 0, & x_i \leq 0 \end{cases} \quad (\text{A.8})$$

$$f'(x_i) = \begin{cases} 1, & x_i > 0 \\ 0, & x_i \leq 0 \end{cases} \quad (\text{A.9})$$

Como podemos observar na figura 63, a função ReLU é zero quando a entrada  $x$  é menor que zero e é igual a  $x$  para valores de entrada maiores que zero. Desse modo, a função ReLU varia seu intervalo entre zero e infinito. Esse comportamento, no contexto das redes neurais, evita e corrige o efeito de dissipação do gradiente presente na função sigmóide e tangente hiperbólica. Por outro lado, o fato de não possuir limite superior, pode gerar instabilidade. Ainda, devido ao gradiente ser zero para entradas negativas, os pesos não podem ser ajustados durante a descida do gradiente, fazendo com que o neurônios deixe de responder a variações de entradas. Isso é conhecido como problema “*Dying ReLU*” (FENG; LU, 2019).

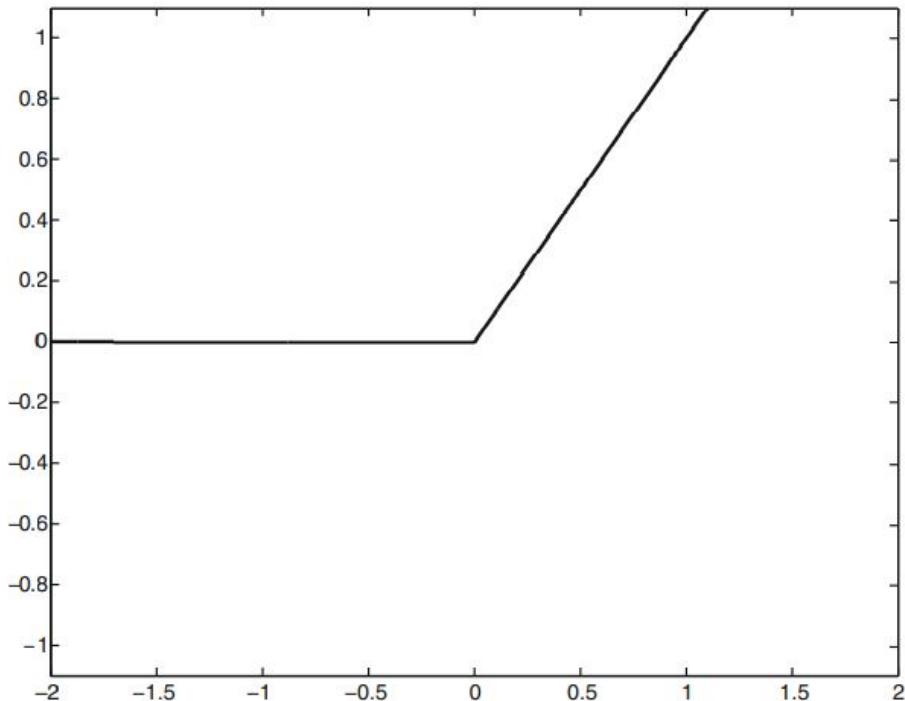


Figura 63 – Curva característica da função ReLU.

Fonte: (AGGARWAL, 2018)

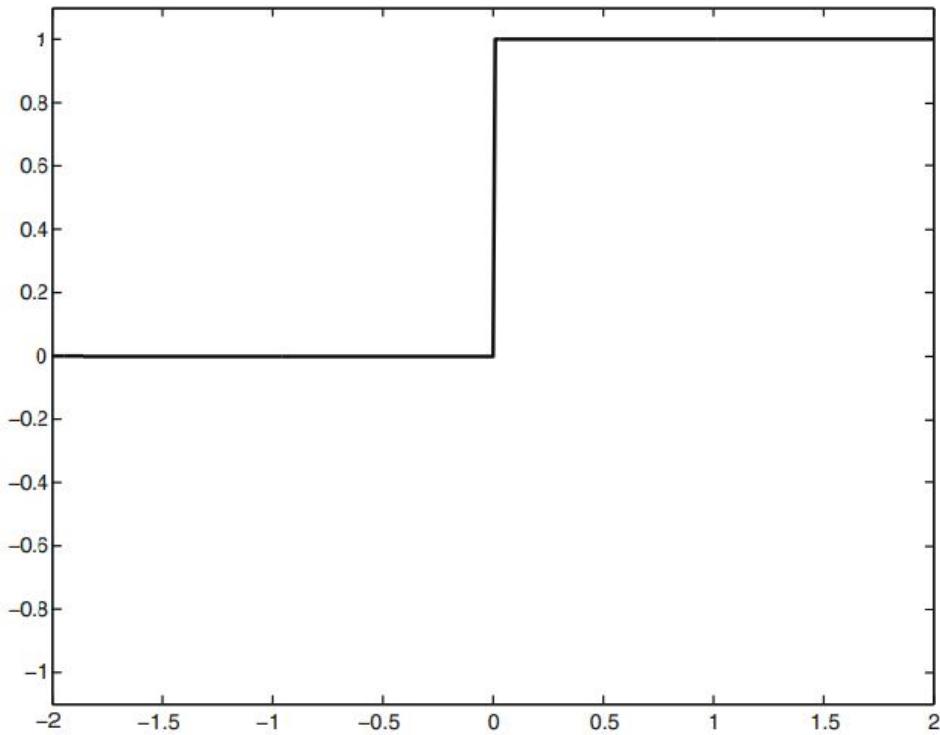


Figura 64 – Curva característica da derivada da função ReLU.

Fonte: ([AGGARWAL, 2018](#))

### A.3.5 Leaky ReLU e PReLU

As funções Leaky ReLU (ReLU com vazamento) e PReLU são dadas pela equação A.10 e sua derivada é dada pela equação A.11. As figuras 65 e 66 representam a sua curva característica e da sua derivada, respectivamente.

$$f(x_i) = \begin{cases} x_i, & x_i > 0 \\ \alpha_i x_i, & x_i < 0 \end{cases} \quad (\text{A.10})$$

$$f'(x_i) = \begin{cases} 1, & x_i > 0 \\ \alpha_i, & x_i < 0 \end{cases} \quad (\text{A.11})$$

Essas funções têm como objetivo minimizar o problema “*Dying ReLU*” adicionando um parâmetro  $\alpha_i$  que fornece uma leve inclinação, fazendo com que a saída seja diferente de zero para as entradas negativas. Quando  $\alpha_i$  possui um valor constante e pequeno, geralmente 0.01, a função é do tipo Leaky ReLU. Caso o valor do parâmetro  $\alpha_i$  for pequeno e variável, trata-se de uma PReLU ([FENG; LU, 2019](#)).

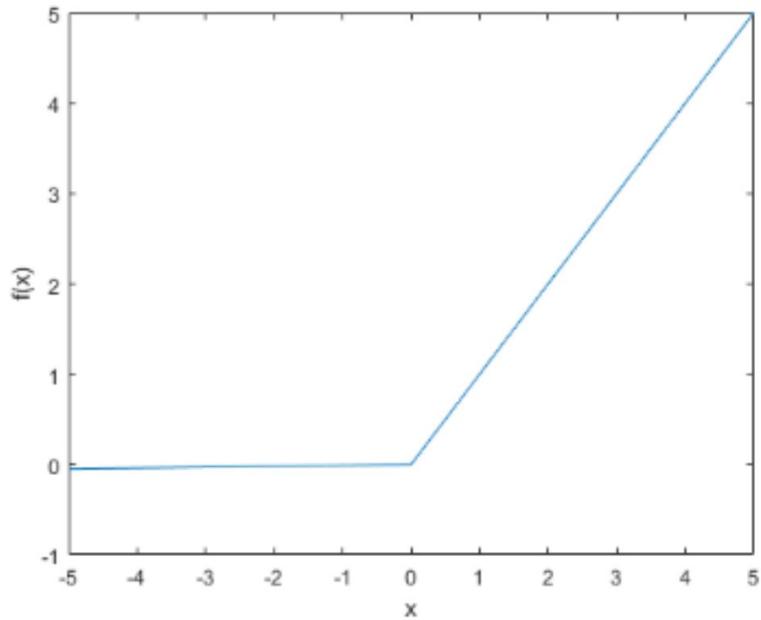


Figura 65 – Curva característica da função Leaky ReLU e PReLU.

Fonte: ([FENG; LU, 2019](#))

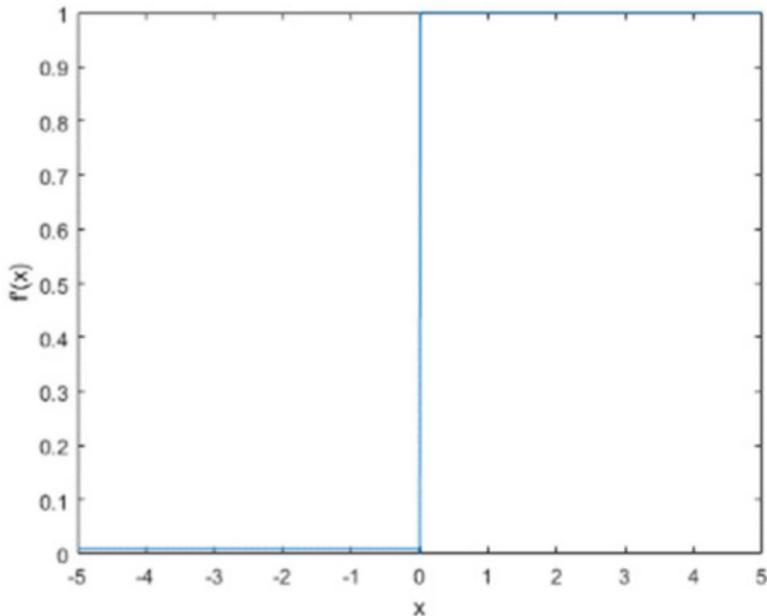


Figura 66 – Curva característica da derivada da função Leaky ReLU e PReLU.

Fonte: ([FENG; LU, 2019](#))

## A.4 Função de perda

A função de perda, do inglês *Loss function*, é um método para avaliar o quanto as previsões de um algoritmo desviou dos resultados reais. Desse modo, e com ajuda de alguma função de otimização, a função de perda aprende a reduzir o erro na previsão tornando possível o aprendizado da máquina. Existem diversos tipos de função de perda, mas

elas podem ser divididas em dois grupos de acordo com a natureza do problema: funções de perdas para problemas de regressão e funções de perdas de classificação (HAYKIN, 2009).

#### A.4.1 Funções de perdas para problemas de regressão

Problemas de regressão tem como objetivo prever valores numéricos contínuos. As funções de perda amplamente utilizadas para esse tipo de problema são: erro quadrático médio e erro absoluto médio.

O erro quadrático médio, do inglês *Mean Square Error (MSE)*, é calculado pela equação A.12. Essa função de perda mede a magnitude das previsões e tem como característica penalizar mais previsões que se distanciam dos valores reais devido a quadratura (GOODFELLOW; BENGIO; COURVILLE, 2016).

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \quad (\text{A.12})$$

O erro absoluto médio, do inglês *Mean Absolute Error (MAE)*, é calculado pela equação A.13. Essa função de perda, assim como o erro quadrático médio, mede a magnitude das previsões, entretanto, por não haver o fator quadrático, não penaliza tanto as previsões que se distanciam dos valores reais em comparação com a MSE (GOODFELLOW; BENGIO; COURVILLE, 2016).

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n} \quad (\text{A.13})$$

#### A.4.2 Funções de perdas para problemas de classificação

Problemas de classificação tem como objetivo prever um conjunto de valores categóricos finitos. As funções de perda amplamente utilizadas para esse tipo de problema são: Perda de Entropia Cruzada e *Multi class SVM Loss*.

A perda de entropia cruzada, do inglês *Cross entropy loss*, é calculada pela equação A.14, onde  $y_i$  representa o valor real e  $\hat{y}_i$  representa o valor previsto pelo modelo. essa função de perda tem como característica aumenta à medida que a probabilidade prevista diverge do rótulo real. Desse modo, a perda de entropia cruzada penaliza fortemente as previsões que são confiáveis, mas erradas (NASR; BADR; JOUN, 2002).

$$CrossEntropyLoss = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \quad (\text{A.14})$$

A função de perda Multi class SVM Loss é dada pela equação A.15 e tem como objetivo realizar penalidades diferentes no ponto que não são previstos corretamente ou muito fechados do hiperplano (GOODFELLOW; BENGIO; COURVILLE, 2016).

$$\text{MultiClassSVMLoss} = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad (\text{A.15})$$

## A.5 Otimização

Em redes neurais, os valores dos pesos sinápticos de todas as camadas são iniciados aleatoriamente, resultando em um valor numérico na camada de saída e um erro associado. Em seguida, por meio de um processo iterativo, os pesos são ajustados com objetivo de aproximar o novo valor da saída ao valor real, reduzindo o erro e minimizando a função de perda. Esse processo é chamado de otimização e existem várias formas de realizá-lo, a mais usual é por meio da backpropagation associado ao gradiente descendente ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)).

### A.5.1 Feedforward

*Feedforward* é uma forma de propagação em que cada camada se conecta à próxima camada, sem conexão com o caminho de volta. Ou seja, a informação flui em um mesmo sentido, partindo da camada de entrada, passando pelas camadas intermediárias em direção à camada de saída sem a existência de *loops*. Durante essa etapa de propagação, os valores são atribuídos ao vetor de saída  $y$  por meio da definição de uma função  $f$ . A figura [67](#) representa um exemplo de arquitetura de multilayar de duas camadas com esse tipo de propagação ([HAYKIN, 2009](#)).

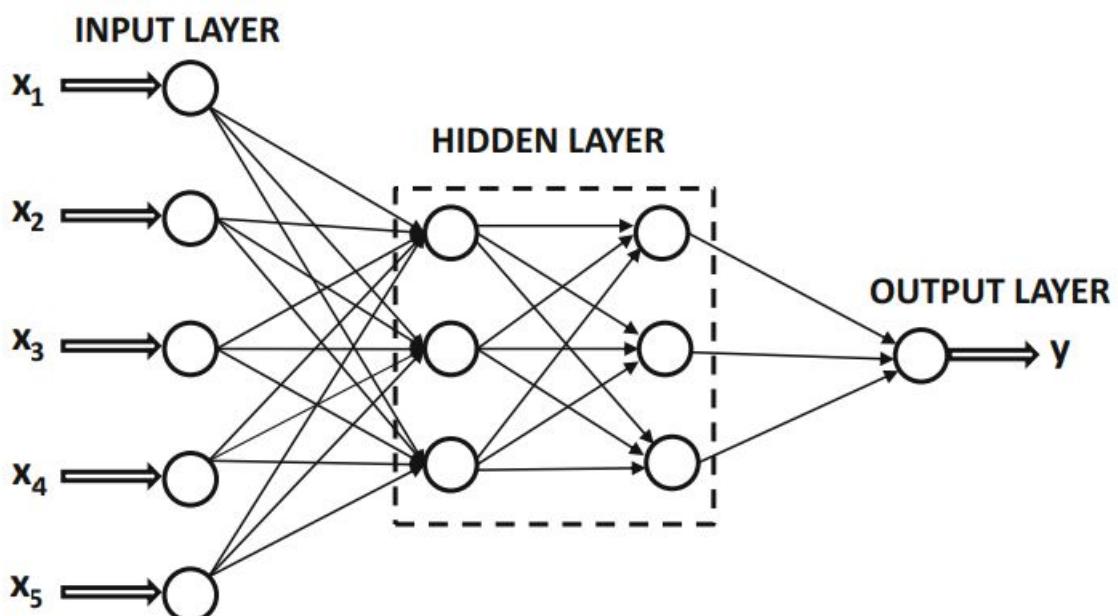


Figura 67 – Arquitetura de rede neural feedforward.

Fonte: ([AGGARWAL, 2018](#))

### A.5.2 Gradiente Descendente

O treinamento de uma rede neural consiste em minimizar a função de perda. Isso é possível por meio do gradiente descendente, do inglês *gradient descent*. Ele é um algoritmo de otimização iterativa de primeira ordem e tem a finalidade de encontrar o mínimo de uma função arbitrária diferenciável (AGGARWAL, 2018).

Considerando um vetor tensorial na entrada de uma rede neural composto por diversas características  $(x_{(1)}, x_{(2)}, x_{(3)}, x_{(4)} \dots, x_{(n)})$  e que a saída predita é dada pela equação A.16

$$\hat{y}_i = \lambda (w_i x_{(i)} + b) \quad (\text{A.16})$$

Como o treinamento da rede neural é um processo iterativo onde na primeira iteração são atribuídos valores aleatórios aos pesos sinápticos ( $w$ ) e ao bias ( $b$ ), deve-se calcular o erro ( $E$ ) e em seguida ajustar os novos parâmetros de ( $w$ ) e ( $b$ ). O cálculo do erro pode ser dado pela função de perda de entropia cruzada descrita na seção A.4.1 e é dada pela equação A.17

$$E = -y \ln(\hat{y}) - (1 - y) \ln(1 - \hat{y}) \quad (\text{A.17})$$

Em seguida, deve-se calcular o gradiente da função de perda  $E$  nos pontos do vetor  $x$ . Para isso, usa-se a equação A.18 onde é calculado as derivadas parciais em relação aos pesos sinápticos  $w$  e em relação ao bias  $b$ , resultando na equação A.19.

$$\nabla E = \left( \frac{\partial}{\partial w_1} E, \dots, \frac{\partial}{\partial w_n} E, \frac{\partial}{\partial b} E \right) \quad (\text{A.18})$$

$$\nabla E = -(y - \hat{y})(x_1, x_2, x_3, \dots, x_n, 1) \quad (\text{A.19})$$

Quanto mais próximo o valor predito do valor real menor é o gradiente. Ainda, o gradiente descendente consiste em subtrair um múltiplo do gradiente da função de perda em cada ponto, atualizando os pesos sinápticos  $w$  com a equação A.20 e o bias  $b$  com a equação A.21, onde  $w'_i$  representa o novo peso sináptico,  $w_i$  o peso sináptico atual,  $\hat{y}$  é o valor predito,  $y$  é o valor real,  $x_i$  é a característica da entrada, o parâmetro  $\beta$  é chamado de Learning Rate que determina o tamanho do passo em cada iteração (AGGARWAL, 2018).

$$w'_i \leftarrow w_i + \beta(y - \hat{y})x_i \quad (\text{A.20})$$

$$b' \leftarrow b + \beta(y - \hat{y})x_i \quad (\text{A.21})$$

### A.5.3 Backpropagation

O *Backpropagation* é um algoritmo iterativo que possibilita que as redes neurais aprendam. Esse algoritmo tem como objetivo ajustar os pesos sinápticos de todas as camadas de uma rede neural com base na taxa de erro obtida na iteração anterior. Para isso, o erro é propagado no sentido contrário em relação ao *feedforward*, geralmente utilizando o cálculo do gradiente descendente, descrito na subseção A.5.2, pela regra da cadeia (HAYKIN, 2009).

A figura 68 ilustra o sentido de propagação *feedforward* e *backpropagation* em uma arquitetura de *multlayer perceptron* com várias camadas.

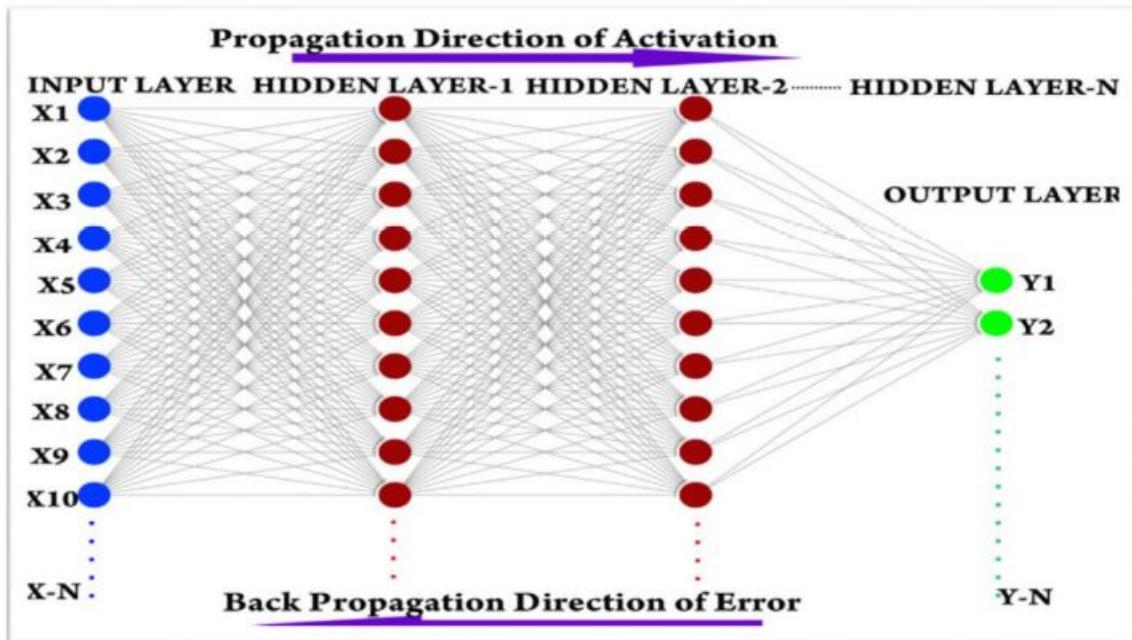


Figura 68 – Backpropagation.

Fonte: (AVUÇLU; BAŞÇIFTÇI, 2018)

No processo de backpropagation é necessário que tanto a função de erro quanto a função de transferência sejam diferenciáveis. Ainda, o ajuste adequado dos pesos garante menores taxas de erro, tornando o modelo confiável ao aumentar sua generalização (AGGARWAL, 2018).

O algoritmo de retropropagação é considerado convergente quando a norma euclidiana do vetor gradiente atinge um limiar de gradiente suficientemente pequeno. E considera-se que o algoritmo de retropropagação, de fato, convergiu quando a taxa absoluta de mudança no erro quadrático médio por época é suficientemente pequena (HAYKIN, 2009). Na prática, para simplificar, esse processo iterativo ocorre quantas vezes for necessário até se obter um erro satisfatório na camada de saída  $y$  ou então até completar o número  $n$  de iterações estabelecido pelo desenvolvedor.

# APÊNDICE B – Resultados completos da validação dos modelos de contagem

Validação do Modelo de Contagem V1																	
Imagen	Qt. Real [un.]	Previsões do Modelo				Métricas Auxiliares				Métricas de Avaliação							
		Qt. Prevista [un.]	Qt. Duplicadas [un.]	Qt. Falso Positivo (NoCow) [un.]	Verdadeiro Positivo (TP) [un.]	Falso Positivo (FP) [un.]	Falso Negativo (FN) [un.]	Qt. Previsões Erradas [un.]	PRECISAO	RECALL	F1 SCORE	ACURÁCIA	Erro Quadrático Médio (MSE)	Erro Absoluto Médio (MAE)	Diferença de contagem ( ΔC ) [un]	Tempo de Contagem (TC) [s]	
cow1	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,748	
cow2	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,427	
cow3	5	3	0	0	3	0	2	2	1,000	0,600	0,750	0,600	0,800	0,400	2	1,393	
cow4	5	4	0	0	4	0	1	1	1,000	0,800	0,889	0,800	0,200	0,200	1	1,459	
cow5	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,381	
cow6	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,450	
cow7	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,419	
cow8	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,463	
cow9	4	4	0	0	4	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,414	
cow10	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,345	
cow11	4	4	0	0	4	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,369	
cow12	5	4	0	0	4	0	1	1	1,000	0,800	0,889	0,800	0,200	0,200	1	1,396	
cow13	11	9	0	0	9	0	2	2	1,000	0,818	0,900	0,818	0,364	0,182	2	1,415	
cow14	8	7	0	0	7	0	1	1	1,000	0,875	0,933	0,875	0,125	0,125	1	1,365	
cow15	9	9	0	0	9	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,411	
cow16	15	14	0	0	14	0	1	1	1,000	0,933	0,966	0,933	0,067	0,067	1	1,341	
cow17	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,384	
cow18	16	15	0	0	15	0	1	1	1,000	0,938	0,966	0,938	0,063	0,063	1	1,366	
cow19	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,391	
cow20	3	2	0	0	2	0	1	1	1,000	0,667	0,800	0,667	0,333	0,333	1	1,368	
cow21	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,377	
cow22	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,382	
cow23	5	5	0	0	5	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,383	
cow24	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,371	
cow25	15	14	0	0	14	0	1	1	1,000	0,933	0,966	0,933	0,067	0,067	1	1,417	
cow26	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,386	
cow27	11	11	0	0	11	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,413	
cow28	9	9	0	0	9	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,393	
cow29	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,406	
cow30	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,400	
cow31	4	3	0	0	3	0	1	1	1,000	0,750	0,857	0,750	0,250	0,250	1	1,400	
cow32	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,398	
cow33	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,525	
cow34	2	1	0	0	1	0	1	1	1,000	0,500	0,667	0,500	0,500	0,500	1	1,380	
cow35	15	14	0	0	14	0	1	1	1,000	0,933	0,966	0,933	0,067	0,067	1	1,390	
cow36	3	4	0	1	3	1	0	1	0,750	1,000	0,857	0,750	0,333	0,333	1	1,371	
cow37	7	7	0	0	7	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,395	
cow38	1	2	0	1	1	1	0	1	0,500	1,000	0,667	0,500	1,000	1,000	1	1,409	
cow39	7	6	0	0	6	0	1	1	1,000	0,857	0,923	0,857	0,143	0,143	1	1,418	
cow40	9	9	0	0	9	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,418	
cow41	5	5	0	0	5	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,381	
cow42	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,406	
cow43	5	5	0	0	5	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,397	
cow44	6	4	0	1	3	1	3	4	0,750	0,500	0,667	0,500	0,429	0,667	0,333	2	1,423
cow45	5	4	0	1	3	1	2	3	0,750	0,600	0,667	0,500	0,200	0,200	1	1,423	
cow46	3	3	0	0	3	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,423	
cow47	11	4	0	0	4	0	7	7	1,000	0,364	0,533	0,364	4,455	0,636	7	1,424	
TOTAL	232	209	0	4	205	4	27	31	0,981	0,884	0,930	0,869	2,280	0,099	23	66,219	
TOTAL médio	4,936	4,447	0,000	0,085	4,362	0,085	0,574	0,660	0,973	0,912	0,932	0,892	0,209	0,108	0,574	1,409	

Figura 69 – Resultados totais da validação do modelo de contagem v1

Fonte: Autoral

Validação do Modelo de Contagem V2																	
Imagen	Qt. Real [un.]	Previsões do Modelo				Métricas Auxiliares				Métricas de Avaliação							
		Qt. Prevista [un.]	Qt. Duplicadas [un.]	Qt. Falso Positivo (NoCow) [un.]	Verdadeiro Positivo (TP) [un.]	Falso Positivo (FP) [un.]	Falso Negativo (FN) [un.]	Qt. Predições Erradas [un.]	PRECISAO	RECALL	F1 SCORE	ACURÁCIA	Erro Quadrático Médio (MSE)	Erro Absoluto Médio (MAE)	Diferença de contagem ( ΔC ) [un]	Tempo de Contagem (TC) [s]	
cow1	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	4,797	
cow2	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	4,630	
cow3	5	3	0	0	0	3	0	2	1,000	0,600	0,750	0,600	0,800	0,400	2	5,052	
cow4	5	2	0	0	0	2	0	3	1,000	0,400	0,571	0,400	1,800	0,600	3	5,427	
cow5	1	0	0	0	0	0	1	1	0,000	0,000	0,000	0,000	1,000	1,000	1	5,346	
cow6	2	1	0	0	0	1	0	1	1,000	0,500	0,667	0,500	0,500	0,500	1	5,423	
cow7	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	5,490	
cow8	1	2	0	1	1	1	0	1	0,500	1,000	0,667	0,500	1,000	1,000	1	5,460	
cow9	4	5	1	0	4	1	0	1	0,800	1,000	0,889	0,800	0,250	0,250	1	5,336	
cow10	1	2	0	1	1	1	0	1	0,500	1,000	0,667	0,500	1,000	1,000	1	4,537	
cow11	4	4	0	0	0	4	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	4,343	
cow12	5	5	0	1	4	1	1	2	0,800	0,800	0,800	0,667	0,000	0,000	0	4,426	
cow13	11	11	0	0	11	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	4,572	
cow14	8	8	0	0	8	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	4,492	
cow15	9	11	2	0	9	2	0	2	0,818	1,000	0,900	0,818	0,444	0,222	2	4,529	
cow16	15	14	0	1	13	1	2	3	0,929	0,867	0,897	0,813	0,067	0,067	1	4,565	
cow17	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	4,465	
cow18	16	15	1	0	14	1	2	3	0,933	0,875	0,903	0,824	0,063	0,063	1	4,630	
cow19	2	1	0	0	0	1	0	1	1,000	0,500	0,667	0,500	0,500	0,500	1	4,613	
cow20	3	2	0	0	2	0	1	1	1,000	0,667	0,800	0,667	0,333	0,333	1	4,586	
cow21	1	2	0	1	1	0	1	1	0,500	1,000	0,667	0,500	1,000	1,000	1	4,545	
cow22	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	4,744	
cow23	5	6	1	0	5	1	0	1	0,833	1,000	0,909	0,833	0,200	0,200	1	4,842	
cow24	1	1	0	0	0	1	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	4,906	
cow25	15	19	3	1	15	4	0	4	0,789	1,000	0,882	0,789	1,067	0,267	4	5,560	
cow26	2	2	0	1	1	1	1	2	0,500	0,500	0,500	0,333	0,000	0,000	0	5,050	
cow27	11	10	0	0	10	0	1	1	1,000	0,809	0,952	0,909	0,091	0,091	1	4,841	
cow28	9	10	1	0	9	1	0	1	0,900	1,000	0,947	0,900	0,111	0,111	1	4,705	
cow29	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	4,796	
cow30	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	4,866	
cow31	4	4	0	0	4	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	5,142	
cow32	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	5,116	
cow33	1	1	0	1	0	1	1	2	0,000	0,000	0,000	0,000	0,000	0,000	0	5,378	
cow34	2	1	0	0	0	1	0	1	1,000	0,500	0,667	0,500	0,500	0,500	1	5,456	
cow35	15	13	0	0	13	0	2	2	1,000	0,867	0,929	0,867	0,267	0,133	2	5,402	
cow36	3	3	0	0	3	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	5,349	
cow37	7	7	0	0	7	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	5,575	
cow38	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	5,432	
cow39	7	6	0	0	6	0	1	1	1,000	0,857	0,923	0,857	0,143	0,143	1	5,312	
cow40	9	9	1	0	8	1	1	2	0,889	0,889	0,889	0,800	0,000	0,000	0	5,405	
cow41	5	5	0	0	5	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	5,243	
cow42	1	2	0	1	1	1	0	1	0,500	1,000	0,667	0,500	1,000	1,000	1	5,117	
cow43	5	5	0	0	5	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	5,317	
cow44	6	7	0	2	5	2	1	3	0,714	0,833	0,769	0,625	0,167	0,167	1	5,364	
cow45	5	11	2	7	2	9	3	12	0,182	0,400	0,250	0,143	7,200	1,200	6	5,194	
cow46	3	3	0	0	3	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	5,164	
cow47	11	7	0	0	7	0	4	4	1,000	0,636	0,778	0,636	1,455	0,364	4	5,239	
TOTAL	232	232	12	18	202	30	30	60	0,871	0,871	0,871	0,771	0,000	0,000	0	235,780	
TOTAL médio	4,936	4,936	0,255	0,383	4,298	0,638	0,638	1,277	0,853	0,843	0,828	0,761	0,446	0,236	0,851	5,017	

Figura 70 – Resultados totais da validação do modelo de contagem v2

Fonte: Autoral

Validação do Modelo de Contagem V3																			
Imagen	Qt. Real [un.]	Previsões do Modelo				Métricas Auxiliares				Métricas de Avaliação									
		Qt. Prevista [un.]	Qt. Duplicadas [un.]	Qt. Falso Positivo (NoCow) [un.]	Verdadeiro Positivo (TP) [un.]	Falso Positivo (FP) [un.]	Falso Negativo (FN) [un.]	Qt. Previsões Erradas [un.]	PRECISAO	RECALL	F1 SCORE	ACURÁCIA	Erro Quadrático Médio (MSE)	Erro Absoluto Médio (MAE)	Diferença de contagem ( DiC ) [un]	Tempo de Contagem (TC) [s]			
cow1	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,835			
cow2	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,411			
cow3	5	3	0	0	3	0	2	2	1,000	0,600	0,750	0,600	0,800	0,400	2	1,466			
cow4	5	4	0	0	4	0	1	1	1,000	0,800	0,889	0,800	0,200	0,200	1	1,499			
cow5	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,477			
cow6	2	1	0	0	1	0	1	1	1,000	0,500	0,667	0,500	0,500	0,500	1	1,477			
cow7	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,491			
cow8	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,473			
cow9	4	4	0	0	4	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,470			
cow10	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,393			
cow11	4	4	0	0	4	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,428			
cow12	5	5	0	0	5	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,380			
cow13	11	10	0	0	10	0	1	1	1,000	0,909	0,952	0,909	0,091	0,091	1	1,405			
cow14	8	7	0	0	7	0	1	1	1,000	0,875	0,933	0,875	0,125	0,125	1	1,414			
cow15	9	9	0	0	9	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,380			
cow16	15	14	0	0	14	0	1	1	1,000	0,933	0,966	0,933	0,067	0,067	1	1,409			
cow17	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,416			
cow18	16	15	0	0	15	0	1	1	1,000	0,938	0,968	0,938	0,063	0,063	1	1,399			
cow19	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,408			
cow20	3	3	0	0	3	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,418			
cow21	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,401			
cow22	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,403			
cow23	5	4	0	0	4	0	1	1	1,000	0,800	0,889	0,800	0,200	0,200	1	1,391			
cow24	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,415			
cow25	15	15	1	0	14	1	1	2	0,933	0,933	0,933	0,933	0,000	0,000	0	1,461			
cow26	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,438			
cow27	11	11	0	0	11	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,525			
cow28	9	9	0	0	9	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,503			
cow29	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,453			
cow30	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,443			
cow31	4	3	0	0	3	0	1	1	1,000	0,750	0,857	0,750	0,250	0,250	1	1,526			
cow32	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,470			
cow33	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,472			
cow34	2	2	0	0	2	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,449			
cow35	15	13	0	0	13	0	2	2	1,000	0,867	0,929	0,867	0,267	0,133	2	1,480			
cow36	3	3	0	0	3	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,451			
cow37	7	7	0	0	7	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,434			
cow38	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,477			
cow39	7	8	1	0	7	1	0	1	0,875	1,000	0,933	0,875	0,143	0,143	1	1,465			
cow40	9	9	0	0	9	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,583			
cow41	5	4	0	0	4	0	1	1	1,000	0,800	0,889	0,800	0,200	0,200	1	1,565			
cow42	1	1	0	0	1	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,662			
cow43	5	6	1	0	5	1	0	1	0,833	1,000	0,909	0,833	0,200	0,200	1	1,617			
cow44	6	6	0	0	6	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,457			
cow45	5	5	0	0	5	0	0	0	1,000	1,000	1,000	1,000	0,000	0,000	0	1,465			
cow46	3	2	0	0	2	0	1	1	1,000	0,667	0,800	0,667	0,333	0,333	1	1,453			
cow47	11	6	0	0	6	0	5	5	1,000	0,545	0,706	0,545	2,273	0,455	5	1,488			
<b>TOTAL</b>	<b>232</b>	<b>215</b>	<b>3</b>	<b>0</b>	<b>212</b>	<b>3</b>	<b>20</b>	<b>23</b>	<b>0,986</b>	<b>0,914</b>	<b>0,949</b>	<b>0,902</b>	<b>1,246</b>	<b>0,073</b>	<b>17</b>	<b>68,998</b>			
<b>TOTAL médio</b>	<b>4,936</b>	<b>4,574</b>	<b>0,064</b>	<b>0,000</b>	<b>4,511</b>	<b>0,064</b>	<b>0,426</b>	<b>0,489</b>	<b>0,992</b>	<b>0,934</b>	<b>0,957</b>	<b>0,927</b>	<b>0,122</b>	<b>0,071</b>	<b>0,447</b>	<b>1,468</b>			

Figura 71 – Resultados totais da validação do modelo de contagem v3

Fonte: Autoral