Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Predição de Escore de Condição Corporal em Bovinos Guzerá Utilizando Redes Neurais Convolucionais

Enzo Faceroli Marques Moreira

JUIZ DE FORA
JULHO, 2026

Predição de Escore de Condição Corporal em Bovinos Guzerá Utilizando Redes Neurais Convolucionais

Enzo Faceroli Marques Moreira

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Luiz Maurílio da Silva Maciel
Coorientador: Frank Angelo Tomita Bruneli

JUIZ DE FORA
JULHO, 2026

PREDIÇÃO DE ESCORE DE CONDIÇÃO CORPORAL EM BOVINOS GUZERÁ UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

Enzo Faceroli Marques Moreira

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Luiz Maurílio da Silva Maciel
Doutor em Engenharia de Sistemas e Computação

Frank Angelo Tomita Bruneli
Doutor em Medicina Veterinária

Marcelo Bernardes Vieira
Doutor em Ciência da Computação

Wagner Antonio Arbex
Doutor em Engenharia de Sistemas e Computação

JUIZ DE FORA

09 DE JULHO, 2026

A Cristo.

Aos pais, irmão e amigos.

Resumo

A avaliação do Escore de Condição Corporal (ECC) é uma prática essencial na pecuária, mas o método tradicional apresenta limitações inerentes à subjetividade. Este trabalho objetivou construir um conjunto de dados original e investigar a aplicação de modelos de redes neurais convolucionais (CNNs) para a predição automatizada do ECC. A metodologia envolveu a aquisição de imagens da região dorsal em ambiente produtivo não controlado, anotação por avaliadores treinados e o pré-processamento dos dados. Foram implementadas e comparadas estratégias de classificação de imagens (VGG-16, ResNet-18, ResNet-50 e DenseNet-121) e de detecção de objetos (YOLO11n e YOLO26n) utilizando a técnica de transferência de aprendizado. Os experimentos evidenciaram a elevada complexidade do problema em condições reais, uma vez que o tamanho reduzido e o desbalanceamento do conjunto de dados limitaram a capacidade de generalização dos modelos. Constatou-se que a redução da granularidade da escala de escores contribuiu significativamente para o aumento da acurácia, com a arquitetura VGG-16 atingindo o melhor desempenho, com 50,76% de acerto para três classes. Por fim, espera-se que os resultados contribuam para o avanço de soluções na pecuária de precisão.

Palavras-chave: Escore de Condição Corporal, Pecuária de Precisão, Redes Neurais Convolucionais, Classificação de Imagens.

Abstract

The evaluation of the Body Condition Score (BCS) is an essential practice in livestock farming, but the traditional method presents limitations inherent to subjectivity. This work aimed to build an original dataset and investigate the application of convolutional neural network (CNN) models for the automated prediction of BCS. The methodology involved the acquisition of images of the dorsal region in an uncontrolled production environment, annotation by trained evaluators, and data pre-processing. Image classification (VGG-16, ResNet-18, ResNet-50, and DenseNet-121) and object detection (YOLO11n and YOLO26n) strategies were implemented and compared using the transfer learning technique. The experiments highlighted the high complexity of the problem under real conditions, as the small size and the imbalance of the dataset limited the models' generalization capacity. We found that reducing the granularity of the score scale contributed significantly to the increase in accuracy, with the VGG-16 architecture achieving the best performance, with a 50.76% accuracy rate for three classes. Finally, it is expected that the results will contribute to the advancement of solutions in precision livestock farming.

Keywords: Body Condition Score, Precision Livestock Farming, Convolutional Neural Networks, Image Classification.

Agradecimentos

Esses agradecimentos são, senão, agradecimentos ao Deus único, que por meio de Cristo me derramou todas as graças que mencionarei.

Antes de tudo, gostaria de agradecer à toda minha família. Em especial, aos meus pais, por acreditarem em mim e me sustentarem durante toda minha trajetória até aqui.

Aos amigos Vitor Vale e Lukas de Sá, por todo o apoio e parceria firmados durante a graduação.

Aos produtores rurais que se disponibilizaram a nos receber e auxiliar na coleta dos dados.

Às colegas Cleuza Samai e Michele Couto, por fazerem parte de todo o processo de coleta dos dados. Sem vocês, a realização desse projeto não seria possível.

Aos mentores Dr. Claudio Napolis e Dra. Maria Gabriela Peixoto, por todo o ensinamento e apoio que me deram ao longo do meu período na Embrapa.

Ao meu orientador, Prof. Dr. Luiz Maurílio, por aceitar fazer parte desse trabalho, pela disponibilidade e por todas as orientações.

Por fim, ao Dr. Frank Angelo Tomita Bruneli. Além de ter sido meu orientador de iniciação científica nos últimos dois anos, também financiou e apoiou a execução deste trabalho. Meus mais sinceros agradecimentos pela parceria e amizade firmada.

Ademais, gostaria de apontar que as palavras aqui usadas não são, nem de perto, capazes de expressar a gratidão verdadeira que sinto. Tampouco é suficiente o espaço de uma página para a menção nominal devida de todos aqueles que me apoiaram até aqui.

"E a Palavra se fez carne e habitou entre nós".

São João (1,14)

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

Adam	*Adaptive Moment Estimation*
AdamW	*Adaptive Moment Estimation Weighted*
ANN	*Artificial Neural Network*
CBIR	*Content-Based Image Retrieval*
CNN	*Convolutional Neural Network*
ECC	Escore de Condição Corporal
FPS	*Frames per Second*
PCA	*Principal Component Analysis*
PNMGuL	Programa Nacional de Melhoramento Genético do Guzerá para Leite
ReLU	*Rectified Linear Unit*
SGD	*Stochastic Gradient Descent*
VGG	*Visual Geometry Group*
YOLO	*You Only Look Once*

1 Introdução

Na última década, com os recentes avanços no poder computacional e o contínuo desenvolvimento de sistemas tecnológicos, o conceito de Pecuária de Precisão emerge como um novo paradigma de manejo na pecuária brasileira e no mundo, tendo como objetivo maximizar a produtividade, aumentar o retorno financeiro e promover o desenvolvimento sustentável (BERCKMANS, 2014). Essa abordagem caracteriza-se pela integração de sensores e pela aplicação de tecnologias de ponta no ambiente produtivo. Essa integração possibilita o monitoramento em tempo real do rebanho, viabilizando a tomada de decisões estratégicas embasada em dados, impulsionando a eficiência da propriedade rural. Dentre os meios de obtenção de dados sensoriais, o emprego de câmeras oferece alto valor ao permitir o monitoramento contínuo e não invasivo dos animais por meio de imagens.

A Visão Computacional é uma subárea da ciência da computação dedicada ao desenvolvimento de modelos matemáticos e algoritmos capazes de descrever o mundo visível a partir de imagens (SZELISKI, 2022). Dessa forma, a área não se limita à mera aquisição de informações visuais, mas estende-se ao processamento, análise e extração destas. Desta forma, sistemas de visão computacional buscam replicar e otimizar a capacidade humana de percepção visual. Nesse contexto, técnicas de visão computacional têm sido cada vez mais empregadas na Pecuária de Precisão. Soluções automatizadas para tarefas como o reconhecimento individual (QIAO et al., 2019), a classificação racial (ACHOUR et al., 2020), a predição de peso (COMINOTTE et al., 2020) e a detecção de comportamento inadequado (CHEN et al., 2021) vêm sendo amplamente divulgadas na literatura.

Entre as diversas tarefas desempenhadas por sistemas de visão computacional, a classificação de imagens apresenta grande potencial de aplicação na Pecuária de Precisão. O objetivo desta tarefa consiste em atribuir um rótulo a uma imagem de entrada a partir de um conjunto de categorias predefinido (GONZALEZ; WOODS, 2018). Historicamente, a extração de informações a partir de imagens digitais exigia o emprego de extratores manuais de características (SZELISKI, 2022). Em contrapartida, no paradigma moderno de visão computacional, mecanismos de classificação de imagem são predominantemente

baseados em redes neurais convolucionais (*convolutional neural networks* – CNN), as quais são capazes de extrair e identificar padrões visuais de forma autônoma e, a partir disso, realizar a predição. Na prática, a dificuldade na realização da tarefa de classificação é diretamente influenciada por fatores como ruído de fundo, variações de iluminação e localização do objeto de interesse. Uma das aplicações da classificação de imagens na Pecuária de Precisão é a avaliação objetiva e automatizada do Escore de Condição Corporal.

1.1 Caracterização do Problema

A avaliação do Escore de Condição Corporal (ECC) (*Body Condition Score* – BCS), é uma prática amplamente adotada na pecuária leiteira para estimar as reservas corporais de gordura e músculo dos animais. Trata-se de um indicador essencial na tomada de decisões relacionadas à nutrição, reprodução, saúde e bem-estar, sendo considerado uma ferramenta estratégica de manejo (ROCHE et al., 2009). O sistema mais difundido para a aferição do ECC de bovinos leiteiros foi inicialmente proposto por Wildman et al. (1982), que classifica os animais em uma escala de 1 (muito magro) a 5 (muito gordo), a intervalos de 0,25 ponto. Os escores são definidos por meio de observações visuais e táteis realizadas por avaliadores treinados. A Figura 1.1 ilustra as diferenças morfológicas observadas na região traseira das vacas para estimativa do ECC.

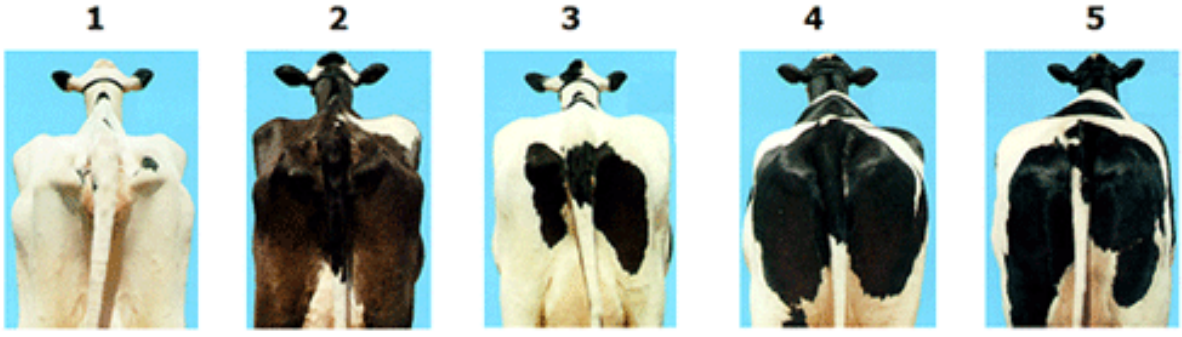

Figura 1.1: Ilustração da escala proposta por Wildman et al. (1982) (MilkPoint, 2026).

Embora possa ser eficiente, o método tradicional apresenta limitações inerentes à subjetividade, como variações entre avaliadores, dependência de treinamento especializado e desafios logísticos em sistemas de produção com grande número de animais. Na prática,

a distinção entre intervalos tão estreitos (0,25 ponto) é de difícil execução, levando muitos avaliadores a adotarem incrementos de 0,5 ou até 1 ponto para reduzir a incerteza da classificação.

Com o avanço das tecnologias de captura e processamento de imagens, bem como dos algoritmos de aprendizado de máquina, soluções promissoras surgiram para a avaliação automatizada do ECC. Sistemas baseados em imagens tridimensionais têm sido amplamente investigados, demonstrando potencial para maior objetividade e precisão (ALVAREZ et al., 2018; SONG et al., 2019; FISCHER et al., 2015; SPOLIANSKY et al., 2016; KOJIMA et al., 2022; LIU et al., 2020; MARTINS et al., 2020).

Mais recentemente, abordagens baseadas em imagens bidimensionais (2D), técnicas de visão computacional e aprendizado profundo para estimar o ECC de bovinos têm ganhado destaque pela simplicidade operacional e menor custo (HUANG et al., 2019; MAHONY et al., 2022). O método proposto por Arbex et al. (2015) também utiliza imagens RGB, porém explora a extração de características lineares como alternativa aos métodos de aprendizado de máquina, a fim de reduzir a demanda de poder computacional e acelerar o processamento.

Entretanto, a maioria dos estudos concentra-se em bovinos da raça Holandesa, não havendo registros de aplicações semelhantes voltadas à raça Guzerá. Essa lacuna evidencia a necessidade de desenvolver métodos tecnológicos automatizados, adaptados às particularidades morfológicas da raça, com viabilidade econômica e aplicabilidade em sistemas produtivos que não dispõem de infraestrutura avançada.

Este trabalho tem como foco principal estruturar um conjunto de dados e investigar o desempenho de diferentes modelos de CNNs na determinação do ECC de bovinos da raça Guzerá a partir de imagens da região dorsal dos animais. É importante destacar que as imagens em que os bovinos foram avaliados não provêm de um ambiente controlado, podendo apresentar variações de iluminação, ângulo, posição, plano de fundo e distância de captura, o que torna a tarefa de classificação mais desafiadora, porém mais semelhante ao dia a dia do sistema produtivo.

1.2 Objetivos

O objetivo geral deste trabalho consiste em construir um conjunto de imagens de bovinos Guzerá e aplicar modelos de redes neurais convolucionais de detecção e classificação de imagens para a predição automatizada do Escore de Condição Corporal. Para a concretização deste propósito, delinearam-se os seguintes objetivos específicos:

- Realizar revisão bibliográfica sobre ECC, visão computacional e algoritmos de aprendizado de máquina aplicados à zootecnia;
- Coletar e organizar um conjunto de dados composto por imagens RGB de bovinos da raça Guzerá, incluindo a anotação de medidas zootécnicas (peso e perímetro torácico) e valores de ECC atribuídos a cada animal por três avaliadores treinados;
- Executar o pré-processamento das imagens, incluindo padronização e segmentação;
- Treinar modelos de redes neurais de classificação e detecção para estimativa do ECC com base nas imagens coletadas;
- Avaliar o desempenho dos modelos, considerando métricas de desempenho e a influência da granularidade da escala de escores na acurácia das predições;

2 Fundamentação Teórica

O objetivo deste capítulo é apresentar os principais conceitos teóricos para a compreensão dos métodos desenvolvidos neste trabalho. Primeiramente, apresenta-se a tarefa de classificação de imagens dentro do escopo da Visão Computacional e o conceito de CNN. Adicionalmente, são apresentadas as arquiteturas de CNNs para classificação de imagens utilizadas neste estudo, destacando suas particularidades em relação às demais. Por fim, a arquitetura *You Only Look Once* (YOLO) é detalhada e apresentada como uma solução para o problema de detecção de objetos em imagens.

2.1 Redes Neurais para Classificação de Imagens

O aprendizado profundo (*deep learning*) é um tipo específico de aprendizado de máquina cujo grande potencial na resolução de problemas se dá pela criação de camadas hierárquicas de processamento e informação. Através do aprendizado profundo, problemas que são intuitivos para o humano, mas cuja descrição formal ou formulação em regras é complexa, como o reconhecimento de fala e tarefas visuais, tornam-se possíveis de serem resolvidos utilizando ferramentas computacionais (GOODFELLOW; BENGIO; COURVILLE, 2016).

Para viabilizar a organização dos dados hierarquicamente, estruturas denominadas redes neurais artificiais (*artificial neural networks* – ANNs) são empregadas. Inspiradas no funcionamento do cérebro humano, as ANNs têm como unidade fundamental de processamento os neurônios artificiais (*artificial neurons*), os quais são dispostos em camadas interconectadas e têm seus parâmetros ajustados através de um processo de otimização baseado em funções matemáticas. A variante de redes neurais que mais se adequa à resolução de problemas envolvendo tarefas visuais são as CNNs.

As CNNs diferem das ANNs tradicionais principalmente pela forma de processamento dos dados de entrada, baseada na operação de convolução. O primeiro modelo de CNN recebeu o nome LeNet e foi desenvolvido por LeCun et al. (1989) como solução

para o problema de reconhecimento de dígitos de códigos de endereçamento postal. A arquitetura se destacou por ter sido pioneira na extração automatizada de informações semânticas a partir de imagens.

Uma imagem pode ser definida como uma função bidimensional $f(x, y)$ onde x e y são coordenadas espaciais e o valor de f é o valor de cor em um ponto (GONZALEZ; WOODS, 2018). A definição da operação de convolução em imagens é descrita pela utilização de um filtro (*kernel*) matricial rotacionado em 180 graus. O filtro é aplicado iterativamente sobre os *pixels* da imagem de forma que um certo *pixel* na imagem resultante seja o resultado da soma dos produtos entre os coeficientes do filtro e os *pixels* da vizinhança estabelecida pelo tamanho do filtro na imagem original (GONZALEZ; WOODS, 2018). A Figura 2.1 ilustra a operação de convolução aplicando um filtro $g(x, y)$ sobre uma região de uma imagem bidimensional $f(x, y)$.

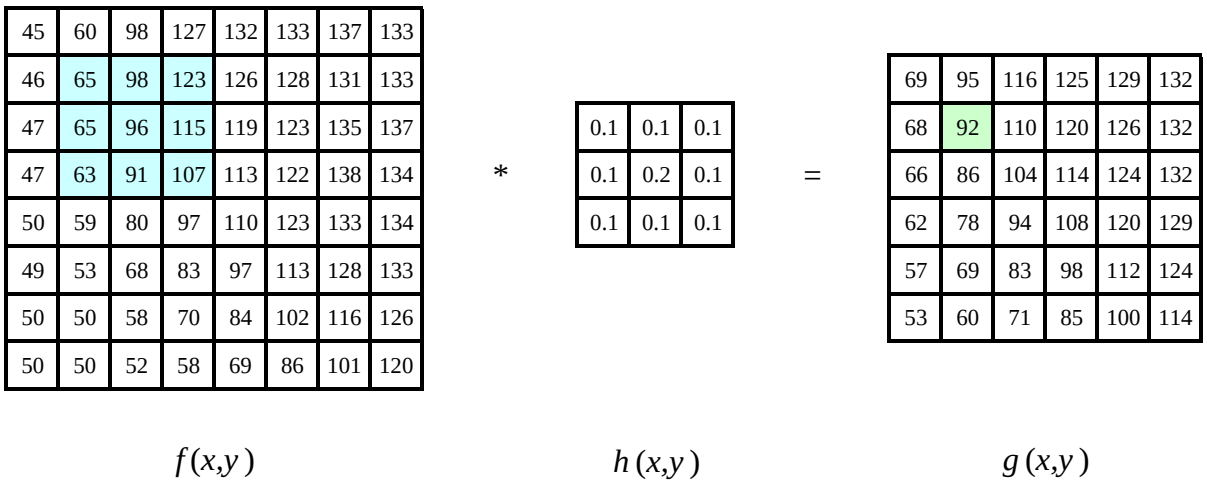

Figura 2.1: Ilustração da operação de convolução 2D (SZELISKI, 2022).

Tipicamente, uma camada convolucional de uma CNN possui três estágios: o estágio de convolução, o estágio de detecção e o estágio de agrupamento (*pooling*) (GOODFELLOW; BENGIO; COURVILLE, 2016). No estágio de convolução, a rede realiza uma série de operações de convolução sobre a entrada da camada, gerando, portanto, um conjunto de ativações lineares. No estágio de detecção, estas ativações lineares passam por uma função de ativação não linear. Desta classe de funções, destacam-se a unidade linear retificada (*Rectified Linear Unit – ReLU*) (KRIZHEVSKY; SUTSKEVER; HINTON, 2017) e a função sigmoide (CYBENKO, 1989). No último estágio, a rede utiliza

uma função de *pooling* para reduzir a dimensão da saída da camada.

As funções de *pooling* desempenham um papel crítico no funcionamento de uma CNN, permitindo a substituição de um valor em uma localização específica no mapa de características por uma síntese estatística das saídas vizinhas (GOODFELLOW; BENGIO; COURVILLE, 2016). A função *max pooling* (ZHOU; CHELLAPPA, 1988), por exemplo, transmite o maior valor dentro de uma vizinhança retangular.

Ao final das camadas convolucionais, a rede conclui a etapa de extração de características. Neste ponto, a imagem de entrada é representada por mapas de características de dimensão reduzida, mas que carregam informações de alto nível. Para possibilitar o processo de classificação, o mapa de características é achatado (processo de *flattening*), sendo transformado em um vetor unidimensional. Esse vetor atua como a entrada para o próximo estágio da rede: o classificador.

O classificador é usualmente composto por uma ou mais camadas totalmente conectadas (*fully connected layers*) e é a estrutura responsável por mapear as ativações do mapa de características para o espaço de probabilidade das classes do problema.

O presente trabalho fez uso de arquiteturas de CNNs consolidadas na literatura. As próximas subseções têm como objetivo detalhar cada uma delas com maior profundidade.

2.1.1 *Visual Geometry Group Network* (VGGNet)

A arquitetura *Visual Geometry Group Network* (VGGNet) é uma CNN desenvolvida pelo *Visual Geometry Group* (VGG) da Universidade de Oxford. Proposta por Simonyan e Zisserman (2014), a estrutura da rede consiste em uma pilha de camadas convolucionais, seguida por três camadas totalmente conectadas. Como entrada, a rede aceita imagens RGB de dimensão 224×224.

A principal inovação estrutural da arquitetura reside na redução da dimensão dos filtros convolucionais utilizados para 3×3, rompendo com o padrão das arquiteturas da época, que utilizavam filtros maiores, como 7×7 e 11×11. Para garantir que o mapa de características preserve a resolução da imagem de entrada, foi utilizado um preenchimento (*padding*) espacial de 1 *pixel*. Ao todo, os autores propuseram seis configurações da

arquitetura, ilustradas na Figura 2.2.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figura 2.2: Configurações da arquitetura VGGNet (SIMONYAN; ZISSERMAN, 2014).

A parte convolucional de todas as configurações possui uma estrutura comum que consiste em cinco blocos de convolução. Em grande parte, os blocos são compostos por múltiplas execuções da operação de convolução. Tal fato é possibilitado pela pequena dimensão do filtro de convolução. A função de ativação utilizada na arquitetura é a ReLU. Outra grande vantagem do uso do filtro 3×3 é a redução do número de parâmetros. Após cada bloco de convolução, a rede aplica uma operação de *max-pooling* com janelas de dimensão 2×2.

O classificador da rede é composto por três camadas totalmente conectadas e uma

camada *softmax*. A primeira e a segunda camada totalmente conectadas são compostas por 4096 neurônios. A última camada totalmente conectada é originalmente composta por 1000 neurônios, que corresponde ao número de classes do conjunto de dados *ImageNet*. A última camada é uma camada *softmax*, responsável por transformar o vetor bruto gerado pela última camada em uma distribuição de probabilidade.

Simonyan e Zisserman (2014) perceberam, através da experimentação, que as configurações C e D se sobressaíam em relação às configurações A e B, indicando que o aumento da profundidade da rede impactava positivamente a acurácia do modelo. Entretanto, o salto de 16 para 19 camadas não apresentou uma melhora significativa no resultado.

2.1.2 *Deep Residual Networks* (ResNet)

À medida que as CNNs se tornavam mais profundas, pesquisadores observavam uma degradação significativa do gradiente nas últimas camadas. Diferente do esperado, algumas arquiteturas apresentavam perda de acurácia em suas versões mais profundas quando comparadas as suas versões mais rasas. Este fenômeno é causado por problemas intrínsecos aos otimizadores, como o gradiente descendente estocástico (*stochastic gradient descent* – SGD). Para resolver esse problema, He et al. (2016) propuseram as redes residuais profundas (*Deep Residual Networks* – ResNets).

Diferentemente das arquiteturas VGG e de outras CNNs tradicionais, as ResNets introduzem o conceito de aprendizado residual (*residual learning*), no qual o objetivo da rede deixa de ser aprender diretamente uma transformação $H(x)$ e passa a aprender uma função residual tal que $H(x) = F(x) + x$. Para isso, empregam-se conexões de atalho (*skip connections*) que implementam um mapeamento de identidade (*identity mapping*), permitindo que a entrada original x seja propagada diretamente para a saída do bloco residual, sem sofrer transformações, sendo então somada à função residual aprendida. Essa reformulação torna o processo de otimização mais eficiente e permite que a informação e os gradientes percorram a rede por caminhos mais curtos, facilitando o treinamento de arquiteturas profundas e contribuindo para mitigar os problemas de degradação do desempenho e de desaparecimento do gradiente.

Os conceitos mencionados anteriormente permitiram um aprofundamento significativo das redes. Originalmente, quatro variações da arquitetura foram propostas, variando de 18 a 152 camadas, conforme a Figura 2.3.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura 2.3: Configurações da arquitetura ResNet (HE et al., 2016).

2.1.3 *Densely Connected Convolutional Networks* (DenseNet)

A arquitetura de redes convolucionais densamente conectadas (*Densely Connected Convolutional Networks* – DenseNets) foi desenvolvida por Huang et al. (2017) a fim de mitigar o problema de degradação do desempenho e a dificuldade de propagação dos gradientes em redes mais profundas. Para isso, a rede introduz o conceito de conectividade densa, em que cada camada recebe como entrada a concatenação das saídas produzidas por todas as camadas anteriores. De maneira análoga, cada camada transmite a todas as camadas posteriores sua própria saída. Formalmente, a saída da camada l é definida como $x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$, em que H_l corresponde à transformação aplicada pela camada l.

Em arquiteturas de redes convolucionais tradicionais, cada camada se conecta apenas à camada subsequente. Dessa forma, uma rede com L camadas possuiria $L - 1$ conexões. As conexões densas presentes nas DenseNets fazem com que uma rede densa com L camadas apresente $L \times (L + 1)/2$ conexões.

Do ponto de vista estrutural, a arquitetura começa com uma camada convolucio-

nal 7×7, seguida por uma camada de *max pooling* 3×3. Em seguida, a rede é composta por quatro blocos densos (*dense blocks*) separados por camadas de transição (*transition layers*), que reduzem as dimensões espaciais dos mapas de características por meio de uma convolução 1×1 seguida de uma operação de agrupamento médio (*average pooling*) 2×2. Os blocos densos são formados pelo agrupamento de camadas densas (*dense layers*), em que cada camada recebe como entrada a concatenação dos mapas de características produzidos pelas camadas anteriores dentro do próprio bloco. Cada camada densa é composta por uma convolução 1×1 (*bottleneck layer*), que fixa o número de canais dos mapas de características, e uma convolução 3×3. Ao final da rede, aplica-se uma camada de agrupamento global médio (*global average pooling*) 7×7, seguida por uma camada totalmente conectada com ativação *softmax*. A Figura 2.4 ilustra um bloco denso composto por cinco camadas densamente conectadas.

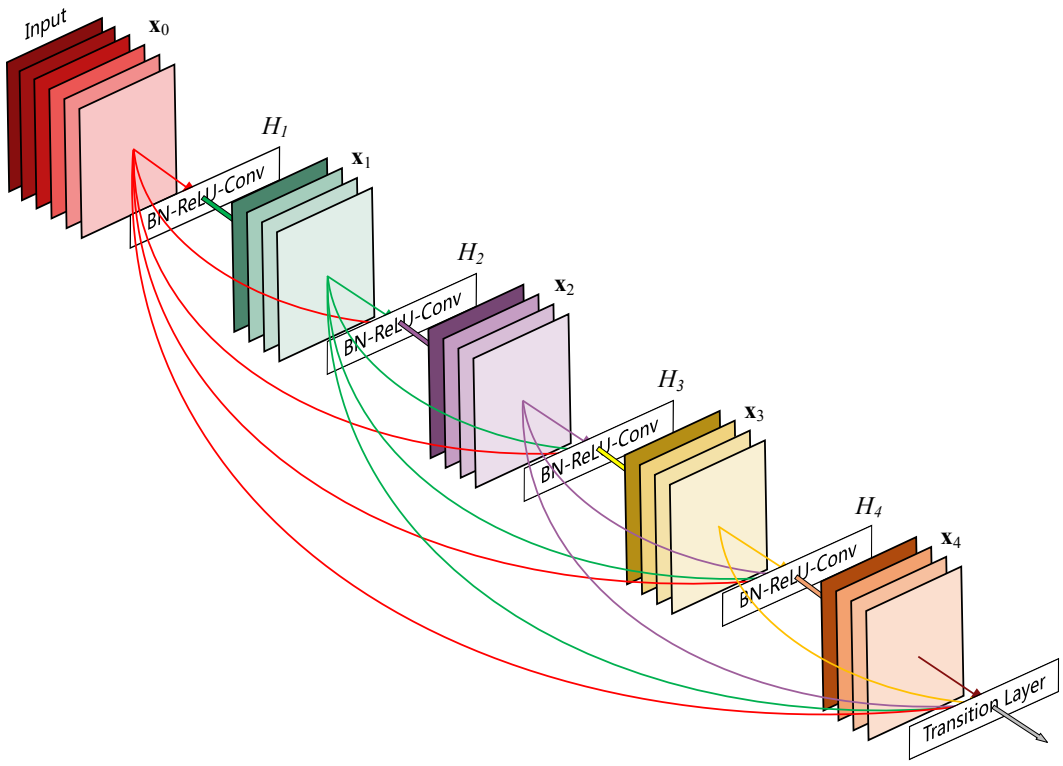

Figura 2.4: Bloco denso de cinco camadas (HUANG et al., 2017).

Para controlar o crescimento da largura da rede, a arquitetura DenseNet emprega a taxa de crescimento (*growth rate*), representada por k, que define a quantidade de novos mapas de características produzidos por cada camada densa. Como cada camada recebe como entrada a concatenação dos mapas gerados por todas as camadas anterio-

res dentro do mesmo bloco, o número total de canais cresce progressivamente à medida que novas camadas são adicionadas. Arquiteturas DenseNet tendem a utilizar valores relativamente baixos de k, explorando o reaproveitamento de características como alternativa ao aumento excessivo do número de filtros. Huang et al. (2017) destacam que essa abordagem apresenta vantagem significativa ao permitir que a arquitetura alcance um desempenho competitivo em tarefas de classificação de imagens com um número reduzido de parâmetros quando comparada a redes como a VGG e ResNet. Huang et al. (2017) propuseram inicialmente quatro variações da DenseNet, variando de 121 a 264 camadas, conforme ilustrado na Figura 2.5.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Figura 2.5: Variações da arquitetura DenseNet (HUANG et al., 2017).

2.2 *You Only Look Once* (YOLO)

As arquiteturas descritas na Seção 2.1 são CNNs para classificação de imagens. A arquitetura *You Only Look Once* (YOLO) (REDMON et al., 2016), por sua vez, foi desenvolvida para o problema de detecção de objetos. O objetivo desta seção é descrever a tarefa de detecção de objetos e detalhar a arquitetura YOLO.

A detecção de objetos consiste em uma tarefa de Visão Computacional cujo objetivo é identificar e localizar instâncias de objetos pertencentes a uma ou mais classes previamente conhecidas em imagens (AMIT; FELZENSZWALB; GIRSHICK, 2020). Di-

ferentemente da classificação de imagens, que atribui um único rótulo à cena completa, a detecção de objetos busca determinar simultaneamente a classe e a posição de cada objeto presente, por meio de caixas delimitadoras (*bounding boxes*) ou outras formas de representação. A complexidade desta tarefa se deve à grande variabilidade de aparência dos objetos e ao número de possíveis posições e escalas em que podem ocorrer na imagem.

As arquiteturas baseadas em CNNs para detecção de objetos podem ser organizadas, de forma geral, em métodos de um estágio (*one-stage*) e dois estágios (*two-stage*). Nos métodos de dois estágios, o processo de detecção é dividido em duas etapas sucessivas: inicialmente são geradas regiões candidatas que potencialmente contêm objetos e, em seguida, essas regiões são refinadas e classificadas. Essa estratégia tende a produzir elevada precisão de detecção, porém, com maior custo computacional. Por outro lado, os métodos de um estágio realizam simultaneamente a localização e a classificação a partir dos mapas de características extraídos pela rede, tratando a detecção como um único problema de regressão.

A arquitetura YOLO é uma CNN de um estágio para detecção de objetos, inicialmente proposta por Redmon et al. (2016), e que se desenvolveu posteriormente para diferentes variantes, como a YOLOv3 (REDMON; FARHADI, 2018) e YOLOv8 (JOCHER; CHAURASIA; QIU, 2023b). Seu funcionamento baseia-se em três principais etapas: o redimensionamento da entrada, a detecção e classificação das caixas delimitadoras e o pós-processamento das detecções. Inicialmente, a rede redimensiona a imagem para 448×448. Na segunda etapa, delimita regiões retangulares para identificar os objetos de interesse na imagem. Por fim, é aplicado o algoritmo de supressão não-máxima (*non-maximum supression*) para filtrar a ocorrência de múltiplas caixas delimitadoras da mesma classe na mesma região da imagem. Redmon et al. (2016) observaram que esse mecanismo eleva a precisão do modelo ao manter apenas as detecções mais confiáveis.

Especificamente, o estágio de detecção e classificação da YOLO divide a entrada em uma grade de tamanho $S \times S$ composta por regiões denominadas células. Cada célula torna-se responsável por detectar a presença do objeto de interesse contido nela. No caso da detecção de um objeto, B caixas delimitadoras são delineadas, juntamente com um vetor de probabilidades C para cada classe. A rede seleciona as células com probabilidades

superiores a um limiar predefinido para delimitar a região onde o objeto se localiza na imagem. Este processo é ilustrado na Figura 2.6.

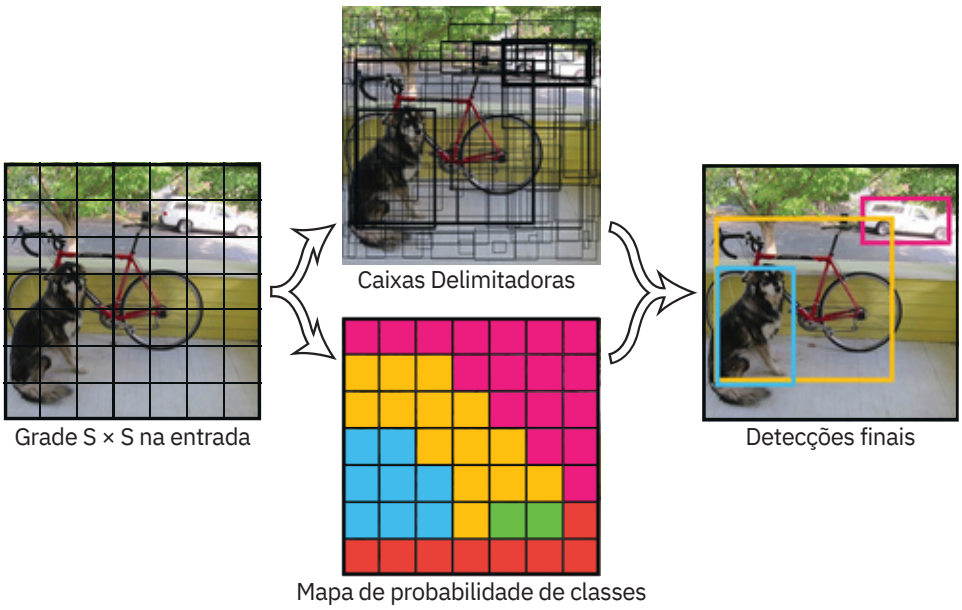

Figura 2.6: Estágio de Detecção da YOLO. Adaptada de Redmon et al. (2016).

De modo geral, a arquitetura original da YOLO é composta por 24 camadas convolucionais e duas camadas totalmente conectadas. Entre algumas das camadas convolucionais principais, são inseridas convoluções com filtros de dimensão 1×1, cuja função é reduzir o número de canais produzidos pelas camadas anteriores. A arquitetura da rede é ilustrada na Figura 2.7.

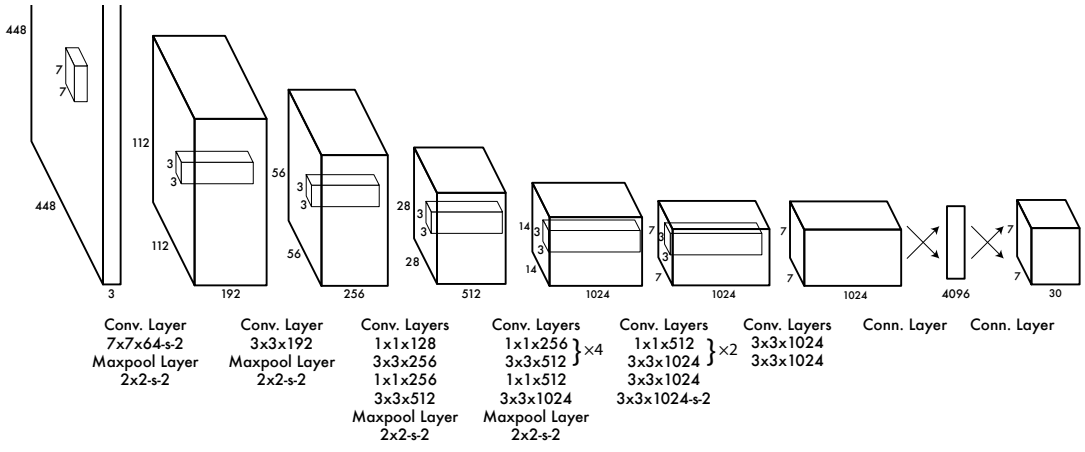

Figura 2.7: Arquitetura da YOLO (REDMON et al., 2016).

3 Trabalhos Relacionados

Este capítulo apresenta uma revisão da literatura referente aos diferentes métodos de aferição objetiva e automatizada do ECC de bovinos através do emprego de técnicas de Visão Computacional. Duas principais classes de abordagens são apresentadas neste capítulo: métodos baseados na extração de características e métodos baseados em CNNs.

3.1 Métodos Baseados na Extração de Características

Sendo um dos estudos pioneiros na área de predição automatizada de ECC por imagens, o trabalho de Bewley et al. (2008) baseou-se na identificação manual de 23 pontos anatômicos em imagens da vista dorsal de vacas localizados especificamente nas regiões da garupa, dos ílios e da inserção da cauda. A partir das coordenadas destes pontos, os autores extraíram 15 ângulos relevantes para a medida do ECC, que serviram como variáveis independentes em modelos de regressão. O estudo utilizou um conjunto de 3.332 imagens de 242 vacas diferentes, uma média de 13,77 imagens por animal. Os autores alcançaram, na escala americana de avaliação (correspondente ao sistema proposto por Wildman et al. (1982)), uma acurácia de 100% dentro de uma margem de erro de 0,5 ponto e 92,79% dentro de uma margem de erro de 0,25 ponto.

Dando sequência ao estudo proposto por Bewley et al. (2008), Azzaro et al. (2011) partiram dos mesmos 23 pontos anatômicos anteriormente propostos. Entretanto, os autores utilizaram a análise de componentes principais (*principal component analysis* – PCA) e sua variação não-linear (*kernel PCA*) para elaborar um modelo estatístico capaz de quantificar as variações do contorno de cada indivíduo em relação a uma forma média estabelecida. O estudo baseou-se em um conjunto de 286 imagens de 29 vacas e utilizou a metodologia de validação cruzada *leave-one-subject-out* para avaliação. Esse trabalho, assim como Bewley et al. (2008), não tratou o problema da estimativa de ECC como um problema de classificação, mas como um problema de regressão e alcançou um erro médio de 0,31 ponto na tarefa na escala de Wildman et al. (1982).

Sob uma perspectiva voltada ao processamento em dispositivos móveis, Arbex et al. (2015) propuseram o aplicativo *e-Score*. Os autores estruturaram um método baseado em recuperação de imagens baseada em conteúdo (*content-based image retrieval* – CBIR). Neste sistema, a inferência do escore ocorre por meio do cálculo de similaridade com imagens de uma base de conhecimento prévia. A partir das características de cor, forma e textura extraídas, o método baseia-se no cálculo da distância entre curvas como medida desse grau de similaridade, de modo que a imagem capturada recebe o escore da representação padrão mais próxima na base de dados. Posteriormente, o método foi validado em um conjunto de 482 imagens de 116 vacas (MATOS et al., 2016), atingindo um índice de acerto de 59,3%.

3.2 Métodos Baseados em Redes Neurais Convolucionais

Em contraste com os métodos clássicos baseados em características identificadas manualmente por pesquisadores, a consolidação das CNNs permitiu que modelos preditivos aprendessem a interpretar informações dos dados brutos das imagens. No contexto da avaliação do ECC, essa mudança viabiliza arquiteturas de detecção e classificação ponta a ponta, mitigando a dependência da interferência humana.

Alvarez et al. (2018) exploraram imagens tridimensionais capturadas por sensores de baixo custo (Microsoft Kinect v2) para abordar o problema utilizando a rede de classificação SqueezeNet, que se caracteriza por apresentar alta acurácia com um número reduzido de parâmetros. Para o treinamento e teste, o estudo constituiu um conjunto de 1.661 imagens da vista dorsal de vacas. Na etapa de pré-processamento, as imagens foram segmentadas para remoção do fundo e convertidas em uma representação de três canais: profundidade, transformada de Fourier e mapa de bordas. Os autores trataram a tarefa como um problema de classificação, e o modelo treinado obteve uma acurácia de 78% para predições com margem de erro de 0,25 ponto e 94% para divergências de até 0,5 ponto.

Huang et al. (2019) elaboraram um método baseado em redes de detecção de uma

etapa, mais especificamente, no modelo *Single Shot Multibox Detector* (SSD), modificando sua estrutura original ao incorporar as arquiteturas DenseNet e Inception-v4. O modelo buscava detectar e classificar, simultaneamente, a região da cauda do animal. O conjunto de dados utilizado consistiu em 8.972 imagens da vista traseira de vacas. A arquitetura SSD aprimorada alcançou uma precisão média (mAP) de 98,46%. Os autores compararam o resultado aos modelos SSD original e YOLOv3, que obtiveram resultados de 99,10% e 88,84% na mesma métrica. Além da alta acurácia, o modelo proposto destacou-se por operar a 115 quadros por segundo (*frames per second* – FPS) e pela compactação (23,1 MB), apresentando desempenho computacional significativamente superior ao do SSD original (39 FPS e 97,1 MB) e ao do YOLOv3 (17 FPS e 246 MB).

Em relação à computação móvel, Çevik (2020) desenvolveu um sistema baseado em imagens RGB e na rede de classificação VGG-19, utilizando transferência de aprendizado (*transfer learning*). O conjunto de dados original foi composto por 505 imagens da porção traseira de bovinos, sendo posteriormente expandido para 2.020 imagens por meio de aumento de dados (*data augmentation*). Ao final do treinamento, a rede alcançou uma acurácia de 94,69% nos testes. O sistema obteve uma acurácia de 78,0% ao ser validado em um conjunto de 50 imagens não contidas no conjunto inicial através de um aplicativo móvel.

4 Conjunto de Dados

O principal desafio enfrentado no desenvolvimento deste trabalho foi a estruturação de um conjunto de dados capaz de atender aos objetivos do projeto. Tendo em vista a especificidade racial e o problema, não foram encontradas bases públicas que contassem com imagens de bovinos Guzerá e seus respectivos rótulos para ECC. Para superar essa limitação, optou-se pela coleta de dados em campo. Essa abordagem possui um alto nível de complexidade logística, porém permite maior proximidade com a realidade do ambiente produtivo, além da possibilidade de captura de imagens de múltiplos ângulos dos animais.

O processo de construção do conjunto de dados para o presente trabalho se deu em quatro principais etapas: aquisição dos dados, seleção dos quadros, anotação das imagens e organização do conjunto para treinamento. As seções a seguir detalham o processo de cada uma destas etapas.

4.1 Aquisição dos Dados

A etapa de aquisição dos dados foi conduzida diretamente nas instalações rurais de quatro sistemas produtivos da raça Guzerá dedicados à produção leiteira. Para isso, foram realizadas visitas técnicas em parceria com a equipe do Programa Nacional de Melhoramento Genético do Guzerá para Leite (PNMGuL).

Em cada visita, foram coletados o ECC o perímetro torácico e o peso de cada animal, quando possível. O processo de aferição do ECC foi conduzido por três avaliadores treinados. Essa escolha se deu para mitigar o viés de um avaliador. Os avaliadores conferiram dois escores para o mesmo animal dentro da escala de 1 a 5 pontos. O primeiro valor era atribuído considerando uma granularidade de 0,5 ponto. O segundo valor atribuído era restrito à granularidade de 1 ponto inteiro. Neste conjunto de dados, o escore de cada animal corresponde à moda das três avaliações. Na ausência de uma moda, optou-se por utilizar a mediana dos valores. O peso dos animais era medido caso houvesse uma balança presente na propriedade. O perímetro torácico, por sua vez, era aferido por

um profissional treinado através de uma fita métrica desenvolvida para esta função.

De modo geral, foram coletadas imagens da vista dorsal e traseira dos animais. As imagens correspondentes à vista traseira de cada um dos animais foram obtidas por meio de uma *webcam* Logitech C930e, em resolução *full HD* (1920×1080) a 30 FPS, mas não foram utilizadas no escopo deste projeto. As imagens da vista dorsal dos animais foram obtidas por meio da câmera de um telefone celular do modelo Apple iPhone13 (câmera grande-angular de 26 mm) em proporção 9:16. Em todas as propriedades, o celular era fixado no teto de um corredor onde os animais poderiam passar, de acordo com a estrutura da propriedade.

A primeira visita técnica foi realizada pela manhã do dia 03/03/2026 em uma propriedade rural localizada no município de Leopoldina-MG. No total, quinze animais foram filmados e tiveram o perímetro torácico aferido. Dois vídeos que englobavam múltiplos animais foram gravados por meio do aparelho celular descrito anteriormente. O aparelho foi fixado no teto a 2,75 metros do chão e configurado para realizar as gravações na resolução 4K (2160×3840) e a 30 FPS. O primeiro vídeo teve duração de dezessete minutos e seis segundos, e o segundo de vinte e dois minutos e trinta segundos. Ao longo da gravação, os animais passavam pela região monitorada pela câmera e tinham suas identificações informadas pela equipe técnica, a fim de garantir a correta correspondência dos dados. Não havia balança para registro do peso dos animais nesta ocasião.

A segunda visita técnica foi realizada no dia 05/03/2026 em uma propriedade rural localizada no município de Carmo-RJ, tendo início pela manhã e sendo finalizada no período da tarde. Neste rebanho, optou-se pela gravação de vídeos individualizados. Ao todo, imagens de dezessete animais foram coletadas. Os vídeos da visão dorsal dos animais foram gravados na resolução *Full HD* (1920×1080) e a 30 FPS. O aparelho celular foi fixado no teto a uma altura de 2,70 metros do chão. Nesta ocasião, o perímetro torácico de cada animal era aferido pelo dono da propriedade. Ao término desta medição, o animal passava por um corredor. A câmera era acionada manualmente antes da entrada de cada animal no corredor. O código de identificação dos animais era informado por voz no microfone do aparelho, para conferência posterior. Nesta ocasião, além do perímetro torácico, o peso dos animais foi aferido em uma balança analógica presente na propriedade.

A terceira visita técnica, também à uma propriedade rural localizada no município do Carmo-RJ, foi realizada no dia 06/03/2026. Nesta ocasião, não havia um corredor coberto para que o celular fosse posicionado. Sendo assim, as condições climáticas não permitiram a gravação adequada dos vídeos. Portanto, as imagens registradas nesta propriedade não foram utilizadas no presente trabalho. Ao todo, 20 animais tiveram seu perímetro torácico e seu peso aferidos.

A última visita técnica foi realizada no dia 20/03/2026 em uma propriedade rural localizada no município de Ibituruna-MG, tendo início pela manhã e sendo finalizada ao fim da tarde. No total, 63 animais deste rebanho foram gravados individualmente na resolução 4K (2160×3840) e a 60 FPS. Neste rebanho, o aparelho celular foi posicionado no teto a uma altura de 3,28 metros do chão. Os animais passavam pelo corredor onde a câmera estava posicionada e aguardavam sua vez para entrar na balança analógica e ter seu peso aferido. Posteriormente, passavam por um tronco, no qual eram presos para aferição do perímetro torácico.

Os vídeos gravados foram segmentados em vídeos menores, descartando trechos onde não havia animais presentes. Utilizando a biblioteca *OpenCV* (BRADSKI, 2000) e a linguagem de programação *Python*, todos os quadros foram extraídos e movidos para pastas cujo nome era o identificador do animal correspondente. No total, 110 vídeos curtos (correspondentes a 85 animais) foram gerados, totalizando 52.945 quadros.

Ao todo, 115 animais passaram pelo processo de avaliação do Escore de Condição Corporal, sendo registradas 6 classes a uma granularidade de 0,5 ponto e 4 classes a uma granularidade de 1 ponto. Dos 115 animais avaliados, este trabalho faz uso de 85 deles. Em grande parte, a exclusão dos animais deste trabalho decorreu de falhas técnicas observadas na aquisição dos dados, como o desligamento da câmera devido à temperatura. Além disso, apenas um animal foi registrado na classe 5,0, impossibilitando a divisão do conjunto em treino e teste. Sendo assim, optou-se por excluir este animal e a classe 5,0 no problema abordado. O conjunto de dados deste trabalho consiste em 82 animais divididos em cinco classes (granularidade 0,5). A Figura 4.1 ilustra a distribuição dos animais por classe.

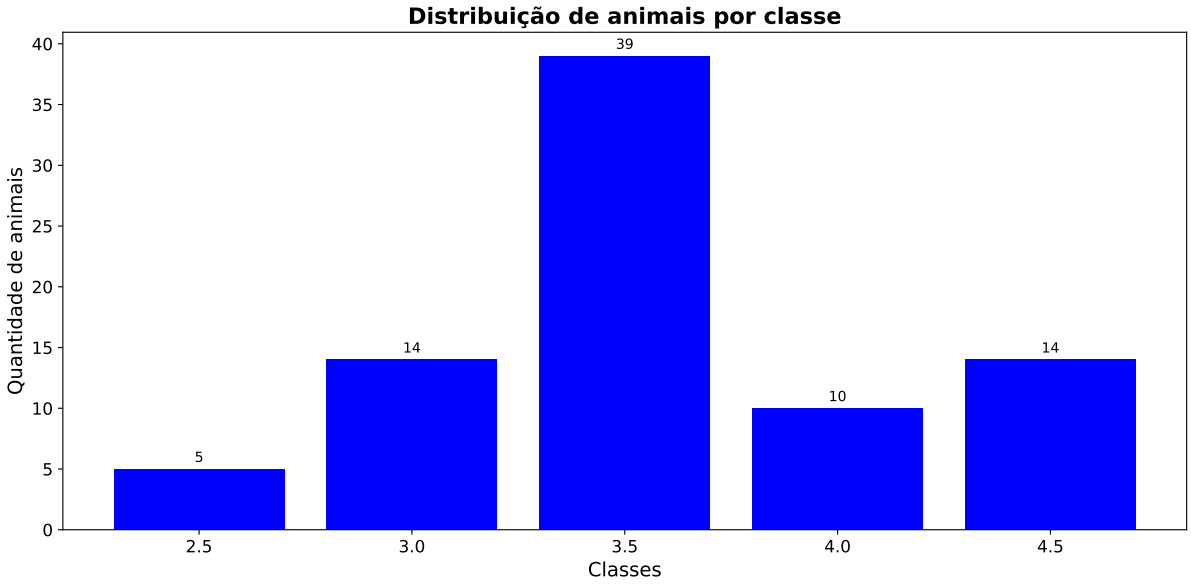

Figura 4.1: Distribuição de animais por classe.

O desbalanceamento entre as classes observado neste conjunto decorre da própria natureza do problema abordado, uma vez que casos de magreza e de gordura extremas são menos frequentes na população. Estas classes minoritárias são predominantemente observadas em casos de manejo inadequado, patologia ou distúrbios reprodutivos (ROCHE et al., 2009).

Após a captura, os quadros dos vídeos foram extraídos por meio de um programa em *Python*, utilizando a biblioteca *OpenCV*.

4.2 Seleção dos Quadros

A seleção dos quadros dos vídeos foi feita manualmente, baseando-se em dois critérios principais: garantir que imagens de um mesmo animal fossem suficientemente diferentes e balancear a quantidade de imagens por animal. O primeiro critério foi adotado para evitar que as redes memorizassem as imagens de entrada, forçando-as a extrair características fenotípicas relevantes para a avaliação do ECC. Por isso, os vídeos foram analisados cuidadosamente, e imagens de um mesmo bovino só eram selecionadas quando havia mudanças claras de posicionamento, postura ou iluminação. Já o segundo critério foi adotado para evitar vieses durante o treinamento. O balanceamento por animal garante

que a rede não foque em detalhes específicos de uma única vaca. A Figura 4.2 ilustra dois quadros, referentes ao mesmo animal, extraídos de um mesmo vídeo, mas considerados suficientemente diferentes entre si.

Figura 4.2: Exemplo de quadros suficientemente diferentes.

Sendo assim, dos 52.945 quadros gerados na etapa de extração, apenas 794 foram selecionados para fazer parte do conjunto de dados deste trabalho. Apesar de ser trivial, o processo de seleção dos quadros demanda uma quantidade considerável de tempo.

4.3 Anotação das Imagens

Para viabilizar a utilização da rede de detecção YOLO e a análise do desempenho das redes de classificação tendo como entrada a imagem correspondente ao recorte da traseira dos animais, as imagens foram anotadas utilizando o programa *LabelStudio* (TKACHENKO et al., 2020). As anotações foram feitas utilizando o padrão de caixas delimitadoras, que são regiões retangulares nas imagens. Duas regiões de interesse foram identificadas com o auxílio da equipe do PNMGuL: a região da traseira e o tronco do animal.

Optou-se pela utilização de dez classes de dois grandes grupos (tronco e traseira) de maneira que cada região possua cinco classes mapeadas segundo o ECC. Como

ilustração, as classes *tronco2_5* e *traseira2_5* identificam, respectivamente, o tronco e a traseira de um animal cujo ECC é 2,5.

A caixa delimitadora do tronco foi feita visando contemplar toda a região entre o fim da giba (popularmente conhecida como cupim) até a região da cauda do animal. Já a caixa delimitadora da região traseira foi feita visando contemplar toda a região da traseira do animal, isto é, do início dos ilíos até o início da cauda. A Figura 4.3 retrata um animal e suas duas caixas delimitadoras correspondentes.

Figura 4.3: Exemplo de anotação por caixas delimitadoras.

4.4 Organização do Conjunto para Treinamento

A fim de melhorar a confiabilidade dos modelos, implementou-se um esquema de validação cruzada estratificada de 5 partições (*5-fold cross-validation*). Esse sistema caracteriza-se pela separação do conjunto de dados em partições (*folds*) de modo que imagens de um mesmo animal nunca estejam presentes simultaneamente em mais de uma partição. Essa separação previne que a rede decore padrões visuais irrelevantes para o problema, como padrões de pelagem ou elementos de fundo. Particularmente, ao dividir o conjunto em cinco partições, quatro eram utilizadas para treinamento e uma era utilizada para teste. As métricas de cada modelo foram avaliadas baseadas na combinação do desempenho no

teste de cada partição.

Para distribuição das imagens do conjunto de dados em cinco partições, utilizou-se um algoritmo guloso que buscava balancear, sobretudo, o número de imagens em cada partição. A cada iteração, o algoritmo selecionava a partição com menor número de imagens e incluía todas as imagens de um animal nela. A Figura 4.4 ilustra a distribuição de quantidade de imagens por partição, assim como a distribuição de classes em uma mesma partição. Adicionalmente, a Figura 4.5 ilustra a distribuição da quantidade de animais por partição.

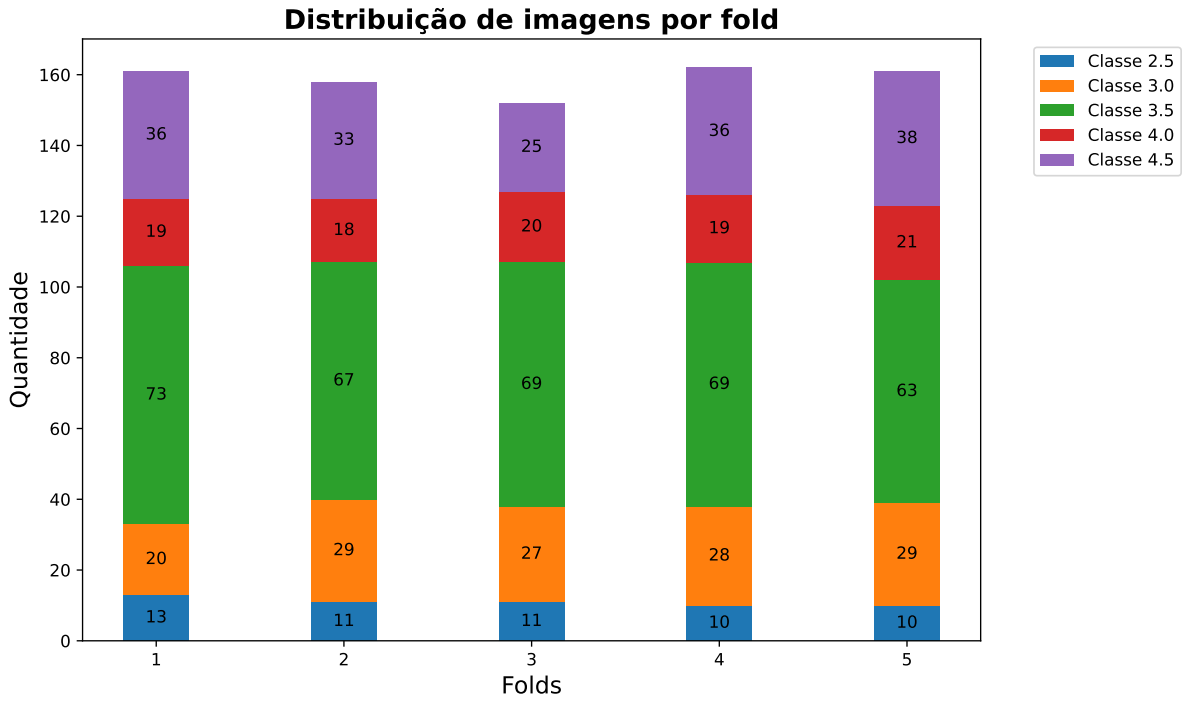

Figura 4.4: Distribuição de imagens por partição.

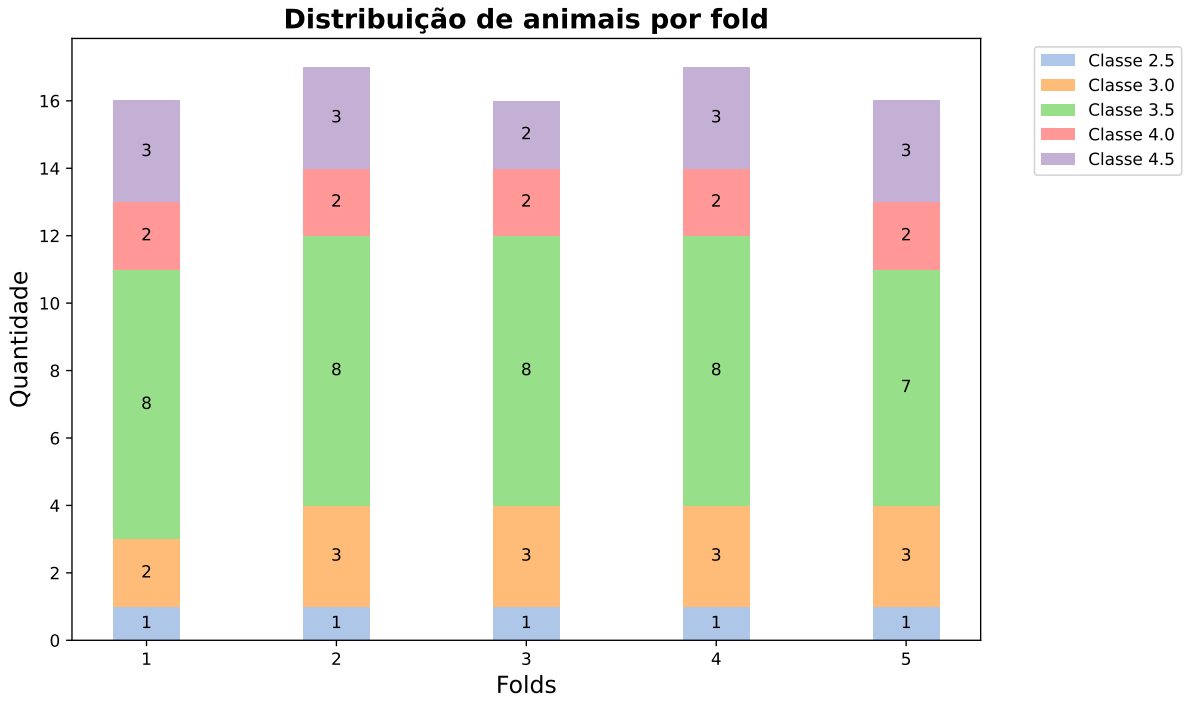

Figura 4.5: Distribuição de animais por partição.

5 Abordagens Propostas

Este trabalho investigou duas principais estratégias de implementação de CNNs para a avaliação objetiva e automatizada do ECC por imagens. A primeira emprega modelos de classificação de imagens, enquanto a segunda utiliza modelos de detecção. As seções a seguir detalham estas estratégias.

5.1 Modelos de Classificação de Imagens

Este trabalho utilizou quatro redes neurais para classificação de imagens: uma variação da arquitetura VGGNet (SIMONYAN; ZISSERMAN, 2014), duas variações da arquitetura ResNet (HE et al., 2016), e uma variação da arquitetura DenseNet (HUANG et al., 2017). Elas foram escolhidas para possibilitar a análise do desempenho de cada uma em relação ao problema abordado, permitindo investigar como diferentes níveis de profundidade e heurísticas de extração de características impactam na acurácia do modelo.

No caso da VGGNet, optou-se por utilizar a configuração D da arquitetura (SIMONYAN; ZISSERMAN, 2014), também conhecida como VGG-16. Esta variação apresenta profundidade superior às configurações A (11 camadas convolucionais) e B (13 camadas convolucionais) e inferior à configuração E (19 camadas convolucionais). A rede apresenta cinco blocos de camadas convolucionais: os dois primeiros são compostos por duas camadas cada, e os últimos três são compostos por três camadas cada. Ao todo, a variação apresenta 16 camadas convolucionais. Em relação à configuração C, a VGG-16 não se diferencia na profundidade, mas sim por não fazer uso dos filtros 1×1, substituindo-os por filtros 3×3. Ademais, a arquitetura mantém os padrões descritos na Seção 2.1.1.

Em relação às ResNets, duas variações foram empregadas: ResNet-18 e ResNet-50. A ResNet-18 é a variação menos profunda proposta por He et al. (2016), totalizando 18 camadas com pesos treináveis. Essa configuração é definida por uma camada convolucional inicial, seguida por 16 camadas convolucionais distribuídas em quatro estágios de blocos residuais, sendo cada estágio estruturado por quatro camadas de filtros con-

volucionais 3×3, organizadas em pares, e uma camada totalmente conectada final. A ResNet-50, por sua vez, é desenvolvida para permitir maior profundidade, mantendo a eficiência computacional. Para isso, a rede substitui os blocos convolucionais por blocos do tipo *bottleneck*. Esses blocos introduzem convoluções 1×1 antes e depois da convolução 3×3, reduzindo e, posteriormente, restaurando a dimensão do mapa de características. Deste modo, a ResNet-50 tem sua profundidade expandida para 50 camadas parametrizadas, mantendo a primeira camada convolucional (como na ResNet-18), seguida por quatro estágios de três, quatro, seis e três blocos *bottleneck*, respectivamente, e uma camada totalmente conectada final.

Quanto às DenseNets, a configuração escolhida foi a DenseNet-121. Essa variação, ainda que apresente profundidade consideravelmente maior em relação às arquiteturas supracitadas, corresponde à versão menos profunda dentre as arquiteturas originalmente propostas por Huang et al. (2017), totalizando 121 camadas. Sua estrutura é definida pelo emprego de quatro blocos densos, compostos por 6, 12, 24 e 16 camadas densas, respectivamente. Em cada bloco, cada camada recebe como entrada os mapas de características gerados por todas as camadas anteriores do mesmo bloco.

Dois formatos de entrada foram utilizados para os modelos de classificação. Inicialmente, os quadros selecionados foram inseridos integralmente, sem nenhum recorte. Em um segundo momento, optou-se por utilizar apenas imagens em formato quadrado da região traseira do animal. Para isso, utilizou-se um programa na linguagem *Python* que recebia as imagens e suas respectivas anotações e extraía um quadrado da traseira tendo como base a maior dimensão do retângulo correspondente à anotação da traseira. Um preenchimento de 100 *pixels* foi acrescido à cada imagem da traseira para preservar parte da informação do fundo. Quando o recorte localizava-se na borda da imagem, acrescia-se o máximo de preenchimento possível, até a borda, e o restante era acrescido no outro lado. Em ambos os casos, as imagens foram redimensionadas para as dimensões originais das redes (224×224).

Para avaliar o impacto do número de classes do problema (granularidade da escala de avaliação), dois formatos de saída foram implementados. Primeiramente, manteve-se as cinco classes descritas previamente na Seção 4.1 de acordo com o escore atribuído a cada

animal pelos avaliadores treinados. Desta forma, a rede buscava atribuir um dos cinco rótulos disponíveis para cada imagem. Depois, reduziu-se o número de classes esperado ao agrupar os rótulos 2,5 e 3,0 no rótulo Magro e os rótulos 4,0 e 4,5 no rótulo Gordo. O rótulo 3,5 foi apenas renomeado para Ideal, reduzindo o problema de cinco para três classes. Essa abordagem favorece o aumento da acurácia dos modelos de classificação.

Adicionalmente, em virtude da baixa disponibilidade de imagens, optou-se pela utilização da técnica de transferência de aprendizado. Tal técnica caracteriza-se pela inicialização da rede com pesos pré-treinados. Camadas congeladas mantêm os pesos pré-treinados e não sofrem alteração nos pesos ao longo do treinamento. A principal vantagem dessa abordagem reside no aproveitamento do aprendizado prévio consolidado pelo treinamento das arquiteturas em conjuntos de dados maiores. Todas as CNNs para classificação de imagens utilizadas neste trabalho foram inicializadas com os pesos pré-treinados no conjunto de imagens *ImageNet*, que possui mais de um milhão de imagens. Tendo em vista que o conjunto *ImageNet* compreende 1000 categorias, a camada de classificação de cada arquitetura foi substituída por uma nova camada totalmente conectada dimensionada especificamente para cinco ou três neurônios de saída, conforme a escala de escores avaliada, para adequar os modelos ao problema abordado. O treinamento das redes baseou-se no modelo de validação cruzada de cinco partições, como mencionado na Seção 4.4.

5.2 Modelos de Detecção de Objetos em Imagens

Além das abordagens baseadas em CNNs para classificação de imagens, foi implementada uma estratégia baseada em detecção de objetos, utilizando a arquitetura YOLO (REDMON et al., 2016) disponibilizada pelo grupo *Ultralytics* através da biblioteca de mesmo nome na linguagem *Python*. As variantes escolhidas foram a YOLO11 (KHANAM; HUSSAIN, 2024) e YOLO26 (JOCHER et al., 2026). Nesse cenário, o problema foi reformulado para incluir não apenas a classificação, mas também a localização espacial de duas regiões anatômicas de interesse (tronco e traseira), associando cada região detectada à classe correspondente ao escore do animal.

No caso da YOLO11, optou-se pela utilização da configuração *nano* (YOLO11n),

correspondente à variante de menor complexidade computacional dentre os modelos disponibilizados (KHANAM; HUSSAIN, 2024). Proposta em 2024, a YOLO11 introduz modificações voltadas ao aumento da eficiência na extração de características em relação a versões anteriores. Destas, destaca-se o emprego do bloco C3k2, que substitui estruturas anteriores baseadas em *Cross Stage Partial* por uma composição de convoluções menores e computacionalmente mais eficientes. Adicionalmente, a arquitetura incorpora o módulo C2PSA (*Cross Stage Partial with Spatial Attention*), responsável por reforçar mecanismos de atenção espacial e favorecer a identificação de regiões relevantes da imagem. Essas modificações permitem reduzir o número de parâmetros e o custo computacional sem comprometer o desempenho preditivo, tornando a YOLO11n adequada para aplicações que demandam inferência em tempo real.

De forma similar a YOLO11, também utilizou-se a configuração *nano* da arquitetura YOLO26 (YOLO26n). Essa arquitetura apresenta-se como uma evolução da YOLO ao introduzir modificações voltadas à simplificação da inferência e ao aumento da eficiência computacional na arquitetura e no treinamento. Em relação à arquitetura, a rede YOLO26 implementa um modelo de cabeça dupla (*dual-head*). A cabeça um-para-muitos (*one-to-many*) mantém a heurística tradicional da YOLO, gerando múltiplas caixas para um único objeto. A cabeça um-para-um (*one-to-one*) força a escolha de apenas uma caixa para cada objeto, eliminando a necessidade do método de supressão não-máxima (*non maximum supression* – NMS). Além disso, a YOLO26 abandona completamente a perda focal de distribuição (*distribution focal loss* – DFL). Em relação ao treinamento, a arquitetura adota o otimizador MUSGD (uma junção do tradicional SGD e do Muon (JORDAN et al., 2024)), a perda progressiva (*progressive loss*) para alinhar dinamicamente o aprendizado entre as cabeças de detecção, e a atribuição de rótulos sensível a pequenos alvos (*small-target-aware label assignment* – STAL), projetada para garantir a supervisão de objetos em pequena escala.

Para a utilização deste modelo, foram consideradas duas diferentes abordagens do problema. Na primeira, o modelo foi treinado para identificar e classificar simultaneamente duas regiões anatômicas de interesse (tronco e traseira), realizando a detecção e classificação de todas as instâncias presentes na imagem. Na segunda, o problema foi

particionado de modo que a rede realizasse a identificação e classificação apenas da região traseira.

Assim como nos modelos de classificação, o emprego da técnica de transferência de aprendizado se manteve. No caso da YOLO, foram utilizados os pesos pré-treinados no conjunto de dados MS COCO (*Microsoft Common Objects in Context*), composto por aproximadamente 330 mil imagens anotadas. A divisão do conjunto de treinamento a partir do modelo de validação cruzada também foi mantida.

6 Experimentos e Resultados

Para avaliar o desempenho dos métodos propostos, conduziu-se uma série de experimentos utilizando o conjunto de dados construído e as arquiteturas previamente mencionadas. Esse capítulo busca detalhar a configuração dos experimentos realizados, bem como as métricas de avaliação adotadas. Além disso, os resultados dos experimentos são discutidos.

6.1 Configuração dos Experimentos

Os experimentos realizados neste trabalho fizeram uso de uma máquina com processador Intel Core i5-10400F, placa de vídeo NVIDIA GTX 1650 4GB, 16GB de memória RAM e sistema operacional Ubuntu 22.04.5 LTS. Os modelos foram treinados utilizando o ecossistema CUDA (*Compute Unified Device Architecture*). Os códigos desenvolvidos fizeram uso da linguagem *Python* na versão 3.11.15. Para a implementação das CNNs de classificação, fez-se uso da biblioteca *Pytorch* na versão 2.10.0. Para a implementação das variações da YOLO, optou-se por utilizar a biblioteca *Ultralytics* na versão 8.4.51. Para o gerenciamento do projeto, utilizou-se o gerenciador de pacotes *UV*, desenvolvido pelo grupo Astral.

A fase de treinamento dos modelos seguiu o protocolo da validação cruzada de cinco partições, conforme detalhado na Seção 4.4.

6.2 Experimentos com as Redes Neurais para Classificação de Imagens

O objetivo dessa seção é descrever o processo de experimentação com as CNNs para classificação de imagens sobre dois conjuntos de entrada. Na primeira abordagem, a rede recebe a imagem completa, sem recorte. Na segunda abordagem, a rede recebe apenas um recorte quadrado da traseira do animal.

6.2.1 Experimentos com as Imagens Inteiras

Inicialmente, optou-se por incluir as imagens do conjunto de dados sem tratamento prévio, com exceção do redimensionamento para a resolução 224×224 para entrada na rede. Além disso, optou-se por utilizar todas as cinco classes apresentadas pelo conjunto. Sob essas condições, as redes VGG-16, ResNet-18, ResNet-50 e DenseNet-121 foram submetidas à experimentação, tendo seus hiperparâmetros como taxa de aprendizado (*learning rate*), tamanho do lote (*batch size*) e número de épocas (*epochs*) variados. Em todos os testes, optou-se por utilizar técnicas de aumento de dados, incluindo leves rotações e espelhamento com probabilidade de 50%. Os otimizadores Adam (*Adaptive Moment Estimation*) (KINGMA; BA, 2015) e AdamW (*Adaptive Moment Estimation Weighted*) (LOSHCHILOV; HUTTER, 2019) foram utilizados durante o processo de treinamento. Também foram realizadas variações no número de camadas treináveis.

Durante a execução destes experimentos, que constituem o *baseline* deste trabalho, observou-se um comportamento de sobreajuste (*overfitting*) em todas as redes, especialmente ao descongelar as camadas convolucionais. Entre todas as arquiteturas, o comportamento de sobreajuste apresentou-se principalmente na VGG-16, que alcançou 90% de acurácia no treino em poucas épocas, apesar de não demonstrar aumento na acurácia nem diminuição de perda durante o teste. A variação na taxa de aprendizado das redes impactava a velocidade do sobreajuste, mas não o evitava. A variação do parâmetro de decaimento dos pesos no otimizador também foi realizada, mas não evitou o sobreajuste mesmo ao utilizar valores elevados. A variação no tamanho do lote e no número de épocas não afetou o desempenho das redes. A Figura 6.1 ilustra o comportamento de sobreajuste da rede VGG-16 (taxa de aprendizado = 10^{-5}). A rede apresentou uma acurácia de 34,9748%.

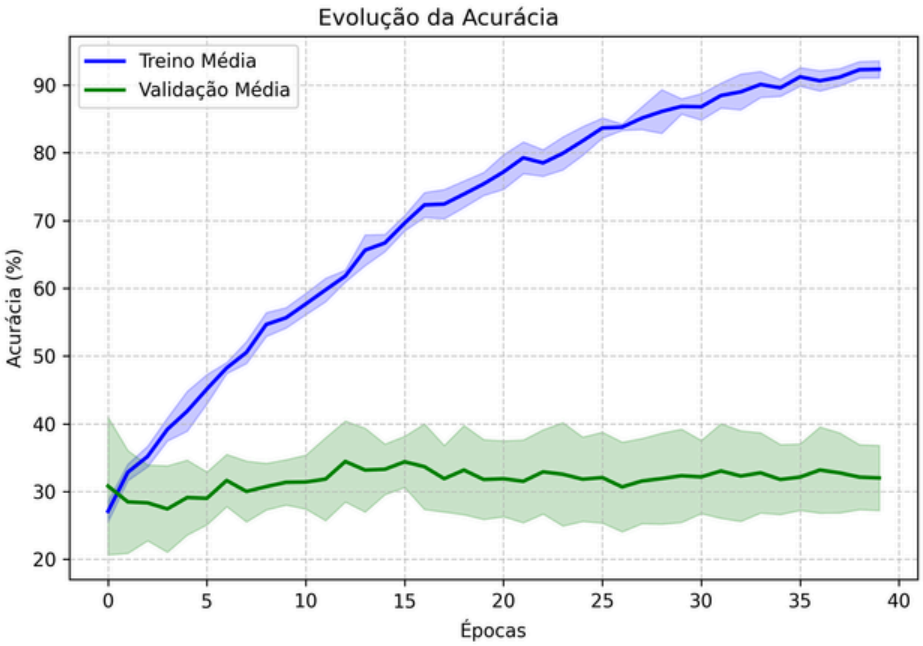

Figura 6.1: Sobreajuste da VGG-16 utilizando a imagem inteira como entrada.

6.2.2 Experimentos com o Recorte da Traseira

Dadas as limitações dos experimentos com as imagens inteiras, prosseguiu-se para a utilização das imagens do recorte da traseira dos animais como entrada da rede, mas ainda mantendo todas as cinco classes. Optou-se por fixar a quantidade de épocas em 45. Esta escolha baseou-se no resultado de experimentos prévios, que indicavam que a utilização de mais épocas não contribuía para o aprendizado das redes.

A experimentação buscou variar o tamanho do lote nos valores 16 e 32, e a taxa de aprendizado nos valores 10^{-4} e 10^{-5}. No caso da rede VGG-16, optou-se por utilizar unicamente a taxa de aprendizado 10^{-5} devido ao comportamento de sobreajuste apresentado em experimentos prévios. O otimizador AdamW foi utilizado, com o decaimento de peso de 10^{-2} para evitar sobreajuste.

Para lidar com o desbalanceamento entre classes durante o treinamento, utilizou-se a função de perda focal (*focal loss*), proposta inicialmente por Lin et al. (2017) e cuja implementação é apresentada no Algoritmo 1. Essa função consiste em uma adaptação da entropia cruzada (*cross-entropy loss*) que introduz um fator modulador $(1 - p_t)^\gamma$, em que p_t representa a probabilidade atribuída à classe correta e γ corresponde ao parâmetro

de focalização, responsável por reduzir progressivamente a contribuição de amostras classificadas com alta confiança e direcionar o aprendizado para exemplos mais difíceis. Além disso, a função incorpora o parâmetro α, utilizado para atribuir pesos distintos entre as classes e compensar situações de desbalanceamento, aumentando a influência de classes menos representadas no cálculo da perda.

Algoritmo 1: Cálculo da função de perda focal

Entrada: Saídas da rede ($inputs$), rótulos reais ($targets$), parâmetro de focalização (γ) e pesos das classes (α).
Saída: Valor da perda focal

início
  $log\_probs \leftarrow log\_softmax(inputs)$;
  $log\_pt \leftarrow$ selecionar as probabilidades correspondentes às classes em $targets$;
  $pt \leftarrow exp(log\_pt)$;
  $focal\_weight \leftarrow (1 - pt)^{\gamma}$;
  $loss \leftarrow -focal\_weight \cdot log\_pt$;
  se $\alpha \neq \varnothing$ **então**
    selecionar pesos α_t correspondentes às classes em $targets$;
    $loss \leftarrow \alpha_t \times loss$;
  fim se
  retornar ($loss$);
fim

A partir de experimentos prévios, definiu-se o parâmetro de focalização como $\gamma = 1,0$. Adicionalmente, utilizou-se o parâmetro α, calculado a partir da distribuição de classes do conjunto de treinamento e posteriormente normalizado de forma que a média dos pesos permanecesse um valor próximo de 1. Dessa forma, classes menos representadas receberam maior contribuição no cálculo da perda, mitigando os efeitos do desbalanceamento sem alterar significativamente a escala global da função de custo. Os resultados obtidos nesses experimentos são detalhados na Tabela 6.1.

Tabela 6.1: Resultados dos experimentos realizados com o recorte traseiro e cinco classes.

Modelo	Taxa de Aprendizado	Tamanho do Lote	Acurácia (%)
VGG-16	10^{-5}	16	38,0769
VGG-16	10^{-5}	32	36,0256
ResNet-18	10^{-4}	16	30,2564
ResNet-18	10^{-4}	32	32,1795
ResNet-18	10^{-5}	16	20,7692
ResNet-18	10^{-5}	32	22,5641
ResNet-50	10^{-4}	16	36,1539
ResNet-50	10^{-4}	32	37,0513
ResNet-50	10^{-5}	16	30,6410
ResNet-50	10^{-5}	32	23,7179
DenseNet-121	10^{-4}	16	35,1282
DenseNet-121	10^{-4}	32	31,2821
DenseNet-121	10^{-5}	16	20,3846
DenseNet-121	10^{-5}	32	20,3633

Elaborada pelo autor (2026).

De modo geral, os resultados não alcançaram uma acurácia semelhante aos trabalhos apresentados na literatura. Através da experimentação, constatou-se que a VGG-16, com taxa de aprendizado 10^{-5} e tamanho do lote 16 se sobressaiu perante as demais, alcançando uma acurácia de 38,7069%. As demais arquiteturas apresentaram seus melhores resultados quando combinadas com a taxa de aprendizado de 10^{-4}. A segunda rede com melhor desempenho foi a ResNet-50, com taxa de aprendizado 10^{-4} e tamanho de lote de 32. A DenseNet-121, assim como a VGG-16, apresenta acurácia levemente superior (35,1282%) quando utilizado o tamanho de lote 16, fato este, que não é observado nas duas variações da ResNet. A melhor configuração da ResNet-18 conseguiu atingir uma acurácia de 32,1795%. Durante os experimentos, observou-se que todas as redes ainda apresentavam um alto nível de sobreajuste, apresentando baixa generalização em todos os casos.

A VGG-16, apesar de apresentar a maior acurácia, se ajustava ao conjunto de treinamento facilmente e, por consequência, não apresentava indícios de aprendizado após

as épocas iniciais, como observado na Figura 6.2. Isso pode ser justificado pela grande quantidade de parâmetros apresentados pela arquitetura. Também observou-se que a função de perda dos testes apresentava comportamento crescente ao passar das épocas. A matriz de confusão para o experimento é ilustrada na Figura 6.3.

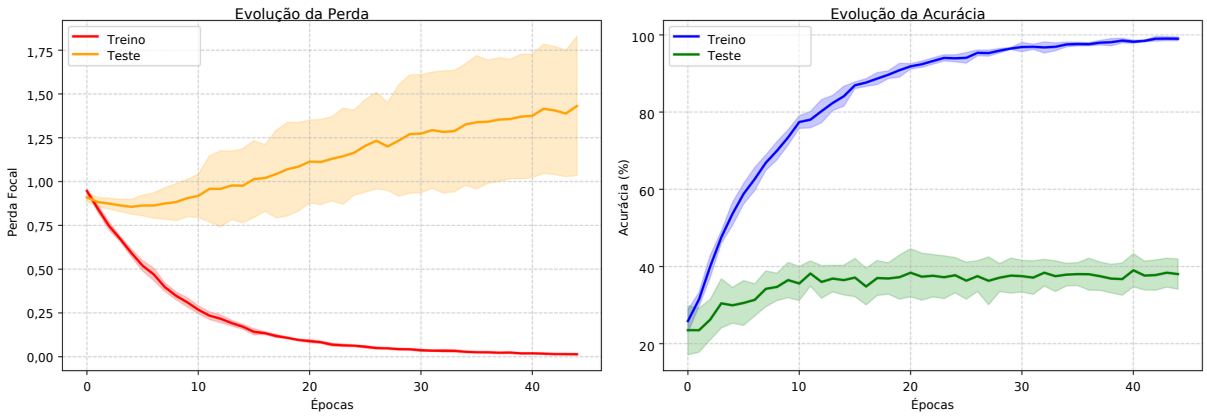

Figura 6.2: Evolução da perda e da acurácia da VGG-16 (tamanho do lote = 16) para cinco classes. O sombreado indica o desvio padrão entre partições.

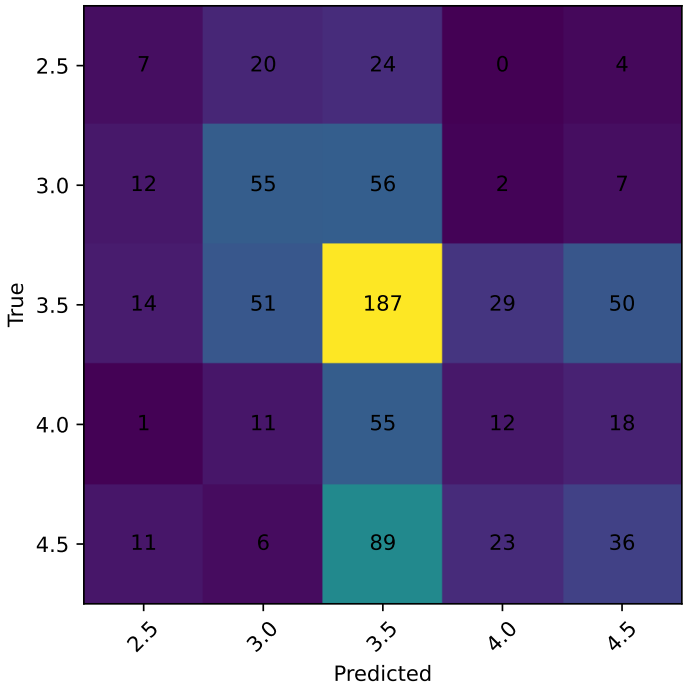

Figura 6.3: Matriz de confusão da VGG-16 no experimento com cinco classes.

A matriz de confusão mostra que a classe 3,5 concentra o maior número de acertos, refletindo também sua maior representatividade no conjunto de dados. Adicionalmente,

observa-se que a confusão das classes se concentra na predição da classe 3,5. Por exemplo, 55 amostras da classe 4,0 e 89 da classe 4,5 foram classificadas como 3,5. Da mesma forma, amostras da classe 2,5 foram classificadas principalmente como 3,0 (20 casos) ou 3,5 (24 casos), evidenciando uma tendência do modelo em favorecer a classe intermediária 3,5. O modelo apresenta confusão entre classes de extremos distintos, porém, estas se mostram mais raras do que confusões entre classes mais próximas. Por exemplo, não houve nenhuma ocorrência de predição da classe 4,0 em imagens da classe 2,5. O caso contrário apresentou uma ocorrência.

Ainda utilizando o recorte da região traseira dos animais, realizou-se a mesma experimentação considerando o sistema de classificação com três classes. Para manter a comparabilidade entre os cenários avaliados, preservaram-se as mesmas configurações de treinamento adotadas nos experimentos anteriores, incluindo arquiteturas, taxas de aprendizado, tamanhos de lote, número de épocas e estratégia de aumento de dados. Entretanto, diferentemente do cenário com cinco classes, optou-se pela utilização da função de perda de entropia cruzada ponderada pelos pesos das classes, isto é, classes mais raras possuem maior peso no cálculo da função de perda. Devido ao menor desbalanceamento observado na configuração com três classes, optou-se por não utilizar a *focal loss*. Assim, os pesos atribuídos às classes foram utilizados apenas para compensar diferenças na distribuição amostral durante o treinamento. Os resultados obtidos para essa configuração são apresentados na Tabela 6.2.

Tabela 6.2: Resultados dos experimentos realizados com o recorte traseiro e três classes.

Modelo	Taxa de Aprendizado	Tamanho do Lote	Acurácia (%)
VGG-16	10^{-5}	16	48,7179
VGG-16	10^{-5}	32	50,7692
ResNet-18	10^{-4}	16	48,2051
ResNet-18	10^{-4}	32	45,1282
ResNet-18	10^{-5}	16	37,1795
ResNet-18	10^{-5}	32	33,3333
ResNet-50	10^{-4}	16	47,5641
ResNet-50	10^{-4}	32	47,5641
ResNet-50	10^{-5}	16	42,4359
ResNet-50	10^{-5}	32	43,2052
DenseNet-121	10^{-4}	16	48,9744
DenseNet-121	10^{-4}	32	45,8974
DenseNet-121	10^{-5}	16	42,9487
DenseNet-121	10^{-5}	32	37,9487

Elaborada pelo autor (2026).

Assim como nos experimentos realizados com o sistema de cinco classes, a VGG-16 apresentou a melhor acurácia entre os modelos. Porém, neste caso, a configuração com maior tamanho de lote se sobressaiu, atingindo uma acurácia de 50,7692%, 2,0513% superior a configuração com tamanho de lote 16. Outro fato observado é que a ResNet-18 também apresentou, em sua melhor configuração, um desempenho superior à ResNet-50, ao atingir 48,2051% de acurácia. A ResNet-50 apresentou uma acurácia de 47,5641% em ambas as configurações que utilizaram uma taxa de aprendizado de 10^{-4}, demonstrando que o tamanho do lote não impactou no desempenho da arquitetura. A DenseNet-121 apresentou o segundo melhor desempenho ao atingir uma acurácia de 48,9744%. A Figura 6.4 ilustra o desempenho da VGG-16 (tamanho do lote = 32) ao longo das épocas. A matriz de confusão para o experimento é ilustrada na Figura 6.5.

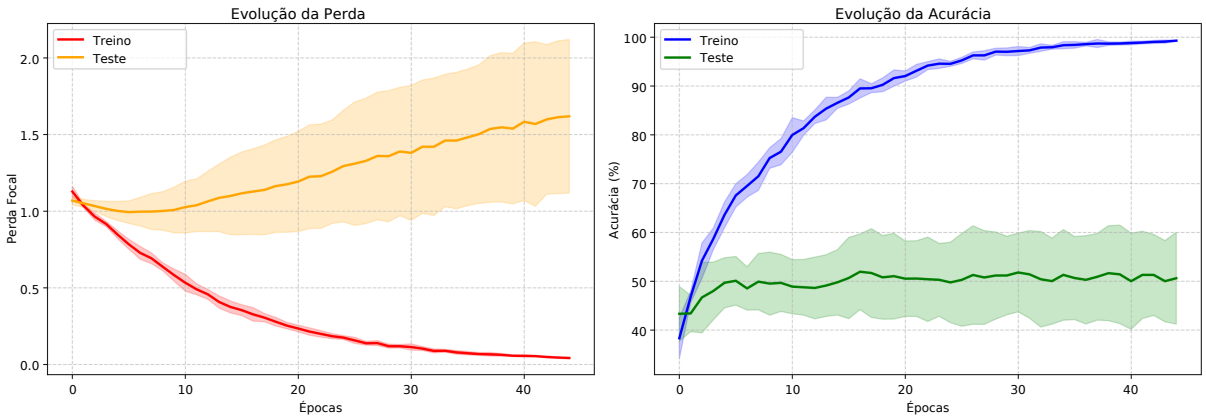

Figura 6.4: Evolução da perda e da acurácia da VGG-16 (tamanho do lote = 32) para três classes.

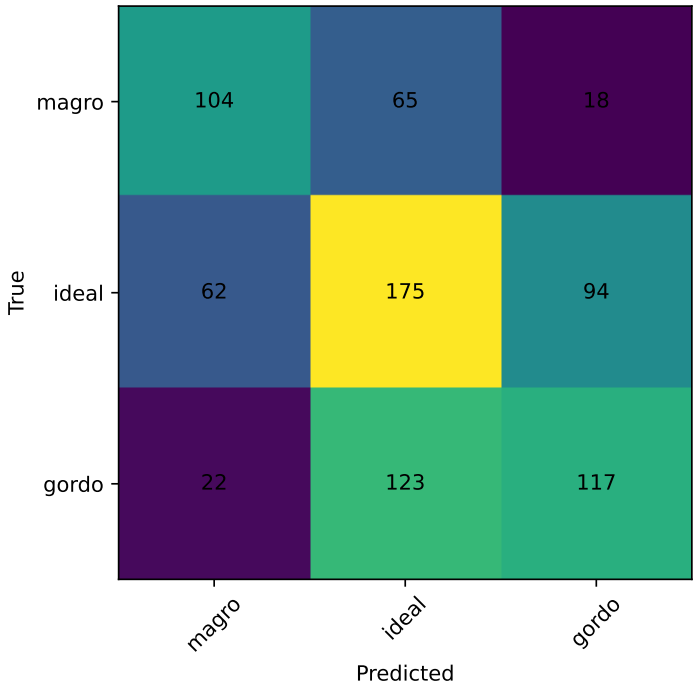

Figura 6.5: Matriz de confusão da VGG-16 no experimento com três classes.

A matriz de confusão mostra que a classe ideal concentra o maior número de acertos, refletindo também sua maior representatividade no conjunto de dados. Observa-se ainda que as confusões ocorrem principalmente entre as classes ideal e as classes adjacentes (magro e gordo). Em contrapartida, são pouco frequentes as confusões entre as classes extremas, com apenas 18 amostras de magro classificadas como gordo e 22 amostras de gordo classificadas como magro.

Através dessa abordagem, observou-se um aumento considerável na acurácia dos

modelos. A redução do número de classes possibilita que confusões entre duas classes magras, ou duas classes gordas, possam ser classificadas como um acerto. Portanto, o aumento na acurácia observado é algo esperado. De modo geral, a ResNet-18, a ResNet-50 e a DenseNet-121 desempenharam pior com uma taxa de aprendizado menor. Esse fato evidencia a necessidade de uma quantidade maior de épocas para estas configurações. A variação no tamanho do lote não apresentou diferenças tão grandes quanto a variação da taxa de aprendizado. Nos experimentos realizados com o sistema de cinco classes, o maior impacto causado pela variação no tamanho de lote foi observado nas configurações de taxa de aprendizado 10^{-5} da arquitetura ResNet-50. Nesse caso, a configuração com menor tamanho de lote (16) obteve acurácia 6,9231% maior que a outra.

6.3 Experimentos com a YOLO

O processo de experimentação prosseguiu com o treinamento das arquiteturas YOLO11n e YOLO26n. De forma análoga ao procedimento adotado para as redes neurais de classificação, foram realizados experimentos com diferentes configurações de hiperparâmetros. Entretanto, pela natureza do problema abordado neste trabalho, optou-se por variar apenas os hiperparâmetros referentes aos componentes de classificação na função de perda (*cls* e *cls_pw*).

O hiperparâmetro *cls* controla a contribuição da perda de classificação para o valor total da função de perda, de modo que valores mais elevados aumentam a influência do desempenho de classificação durante o treinamento. Já o hiperparâmetro *cls_pw* está relacionado à ponderação entre classes e possui papel particularmente relevante em conjuntos de dados desbalanceados, permitindo atribuir maior penalização aos erros cometidos em classes menos representadas e, consequentemente, reduzir o viés em favor das classes majoritárias.

Por padrão, a YOLO incorpora técnicas de aumento de dados durante o treinamento. Neste trabalho, optou-se por manter as configurações padrão da arquitetura para esse processo. Os principais hiperparâmetros de aumento de dados e os respectivos valores adotados são apresentados na Tabela 6.3. As transformações no espaço de cores HSV (*hsv_h*, *hsv_s* e *hsv_v*) foram empregadas para modificar tonalidade, saturação

e intensidade dos *pixels*. Também foram empregadas transformações espaciais, incluindo translação (*translate*) e variação de escala (*scale*), que promovem deslocamentos e alterações no tamanho aparente dos objetos durante o treinamento. Além disso, foram aplicados espelhamento horizontal (*fliplr*) e a técnica de mosaico (*mosaic*), que realiza a combinação de múltiplas imagens de treinamento em uma única amostra.

Tabela 6.3: Hiperparâmetros de aumento de dados utilizados no treinamento da YOLO.

Hiperparâmetro	Descrição	Valor
hsv_h	Alteração da matiz	0,015
hsv_s	Alteração da saturação	0,7
hsv_v	Alteração do brilho	0,4
translate	Deslocamento da imagem	0,1
scale	Escalonamento da imagem	0,5
fliplr	Espelhamento horizontal	0,5
mosaic	Combinação de múltiplas imagens	1,0

Elaborada pelo autor (2026).

Para o treinamento das arquiteturas, definiu-se a utilização de 40 épocas a partir de experimentos prévios. O tamanho de lote foi fixado em 8. As imagens de entrada foram redimensionadas para 640×640. A técnica de mosaico era desativada nas últimas 10 épocas. O otimizador utilizado foi o AdamW, em sua configuração padrão para a YOLO. Os hiperparâmetros *cls* e *cls_pw* foram avaliados em dois cenários: o valor padrão da arquitetura e o maior valor para esses hiperparâmetros permitido por ela.

Para manter o padrão utilizado na literatura para avaliação do desempenho da YOLO no problema abordado neste trabalho, optou-se por utilizar as métricas mAP@50 e mAP@50-95. A métrica mAP@50 corresponde à média dos valores de precisão média (*average precision* – AP) obtidos para cada classe, considerando como corretas apenas as predições cuja sobreposição entre a região prevista e a região de referência atinja um valor mínimo de 0,5, medido pela métrica de interseção sobre união (*intersection over union* – IoU). Já a métrica mAP@50-95 amplia esse critério ao calcular o AP em múltiplos limiares de IoU, variando de 0,50 a 0,95 com incrementos de 0,05, utilizando posteriormente a média dos resultados obtidos. O AP de cada classe é calculado a partir da curva

precisão–revocação (*precision-recall*), permitindo avaliar simultaneamente a capacidade do modelo de identificar corretamente os objetos e reduzir detecções incorretas. Os resultados de mAP apresentados neste trabalho foram obtidos através do cálculo da média ponderada do mAP de cada partição.

Sendo N_i o número de imagens presentes na partição i e mAP_i o respectivo valor de mAP obtido nessa mesma partição, o $\text{mAP}_{\text{final}}$ do modelo foi calculado por meio de uma média ponderada. Para isso, a métrica de desempenho de cada partição foi multiplicada pela sua quantidade de imagens correspondente. O somatório desses produtos, englobando todas as 5 partições avaliadas, foi então dividido pela quantidade total de imagens do conjunto de dados (N_{total}).

$$\text{mAP}_{\text{final}} = \frac{\sum_{i=1}^{5}(N_i \cdot \text{mAP}_i)}{N_{\text{total}}}. \tag{6.1}$$

Adicionalmente, ambas as arquiteturas foram avaliadas utilizando duas configurações de classes. Na primeira, composta por 10 classes, utilizaram-se instâncias tanto do tronco quanto da traseira de cada animal, associadas ao seu respectivo ECC. Na segunda configuração, utilizou-se exclusivamente a região da traseira. Sendo assim, os experimentos foram realizados conforme apresentado na Tabela 6.4.

Tabela 6.4: Resultados dos experimentos realizados com as arquiteturas YOLO.

Modelo	Configuração	*cls*	*cls_pw*	mAP50	mAP50-95
YOLO11n	10 classes (tronco + traseira)	0,5	0,0	0.3681	0.3418
YOLO11n	10 classes (tronco + traseira)	4,0	1,0	0.3138	0.2879
YOLO26n	10 classes (tronco + traseira)	0,5	0,0	0.2998	0.2781
YOLO26n	10 classes (tronco + traseira)	4,0	1,0	0.2959	0.2731
YOLO11n	5 classes (traseira)	0,5	0,0	0.3654	0.3329
YOLO11n	5 classes (traseira)	4,0	1,0	0.3889	0.3456
YOLO26n	5 classes (traseira)	0,5	0,0	0.2829	0.2565
YOLO26n	5 classes (traseira)	4,0	1,0	0.2584	0.2328

Elaborada pelo autor (2026).

Os resultados obtidos indicam que a arquitetura YOLO11n apresentou desempe-

nho superior à YOLO26n em ambas as configurações avaliadas. Na configuração com 10 classes, utilizando imagens do tronco e da traseira, a YOLO11n alcançou os maiores valores de mAP@50 (0,3681) e mAP@50-95 (0,3418) ao utilizar os hiperparâmetros padrão para as perdas de classificação (cls = 0,5 e cls_pw = 0,0). No caso dos experimentos com 10 classes, o aumento desses hiperparâmetros para cls = 4,0 e cls_pw = 1,0 resultou em uma redução do desempenho para ambas as arquiteturas. Já no caso dos experimentos com 5 classes, o aumento dos hiperparâmetros resultou em um mAP@50 maior na YOLO11n.

Ao comparar as arquiteturas, observa-se que a YOLO11n superou consistentemente a YOLO26n, independentemente da configuração utilizada. Na configuração com 10 classes, a diferença de aproximadamente 0,068 em mAP50 evidencia uma vantagem significativa da YOLO11n, enquanto na configuração com 5 classes, essa diferença aumentou para cerca de 0,083.

A utilização apenas da região da traseira, reduzindo o problema para cinco classes, produziu resultados bastante próximos aos obtidos com as dez classes. Para a YOLO11n, o mAP@50 passou de 0,3681 para 0,3654, uma redução inferior a 1%, enquanto o mAP@50-95 diminuiu de 0,3418 para 0,3329. Resultado semelhante foi observado para a YOLO26n. Esses resultados sugerem que, embora a configuração com cinco classes represente um problema de classificação menos complexo, os resultados obtidos foram muito próximos aos observados com dez classes.

Além da análise de precisão média, a análise das matrizes de confusão apresenta elevada relevância por permitir uma avaliação detalhada do comportamento do modelo para cada classe individualmente. Enquanto o mAP fornece uma medida agregada do desempenho geral, a matriz de confusão evidencia padrões específicos de erro ao indicar quais classes são mais frequentemente confundidas entre si. Dessa forma, torna-se possível identificar dificuldades de discriminação entre categorias e orientar ajustes futuros no modelo ou no conjunto de dados.

As matrizes de confusão utilizadas neste trabalho foram geradas pelo *framework Ultralytics*. Para construí-las, o *framework* associa as detecções realizadas pela YOLO às anotações de referência do conjunto de dados por meio do cálculo do valor de IoU entre

cada caixa prevista e sua correspondente caixa real, considerando como válidas apenas as associações que satisfazem o limiar de 0,45 (JOCHER; CHAURASIA; QIU, 2023a). A partir dessas correspondências, são contabilizados os acertos e erros de classificação entre as diferentes categorias, de modo que os elementos da diagonal principal representam detecções corretas, enquanto os elementos fora da diagonal indicam confusões entre classes. Adicionalmente, a matriz permite identificar objetos não detectados e detecções sem correspondência, representados pela classe *background*. Para a apresentação dos resultados, foi construída uma matriz de confusão consolidada por arquitetura a partir da agregação das matrizes obtidas nas cinco partições da validação cruzada. A matriz de confusão da YOLO11n com os hiperparâmetros padrão no problema de 10 classes para a partição 1 é ilustrada na Figura 6.6.

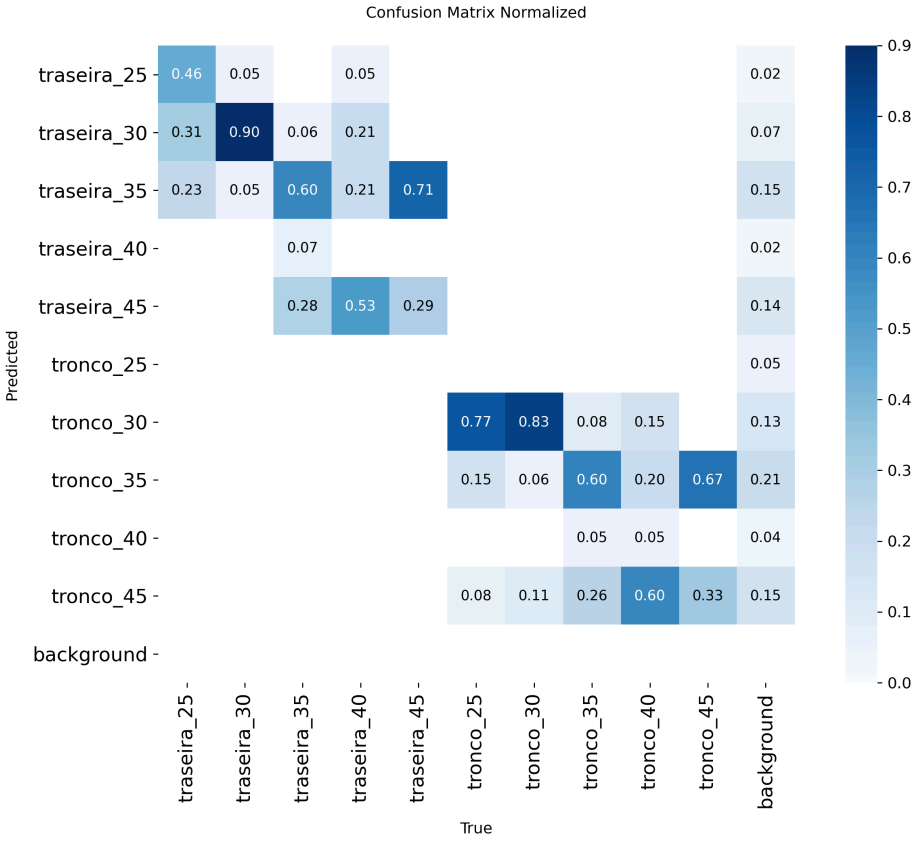

Figura 6.6: Matriz de confusão da YOLO no experimento com dez classes para a partição 1.

Observa-se que, através da traseira, o modelo apresentou maior facilidade na identificação da classe 3,0, apresentando 90% de precisão. A classe 3,5 foi identificada

corretamente em 60% dos casos utilizando a traseira e o tronco. A traseira da classe 2,5 foi identificada corretamente em 46% dos casos, mas não houve acertos na identificação do tronco da classe. A rede também apresentou maior facilidade na identificação da classe 3,0 através do tronco, identificando-a corretamente em 83% dos casos. A rede não foi capaz de identificar a traseira e o tronco da classe 4,0. Além disso, apresentou maior facilidade de identificar as classes mais magras.

Figura 6.7: Matriz de confusão da YOLO no experimento com cinco classes para a partição 1.

A matriz de confusão da YOLO11n com os hiperparâmetros ajustados sob o conjunto de 5 classes para a partição 1 é ilustrada na Figura 6.7. Ao utilizar apenas classes da traseira, a rede ainda tem maior taxa de sucesso na classe 3,0, identificando-a corretamente em 75% dos casos. A identificação da classe 2,5 aumentou para 62%. Sob essa abordagem, o modelo manteve-se sem identificar corretamente a classe 4,0 e identificou a classe 4,5 com sucesso em apenas 29% das vezes.

7 Considerações Finais

Este trabalho teve como objetivo construir um conjunto de dados composto por imagens de bovinos da raça Guzerá e investigar a aplicação de modelos de redes neurais convolucionais para a predição automatizada do Escore de Condição Corporal (ECC). Para isso, foram conduzidas etapas de aquisição e anotação dos dados em campo, pré-processamento das imagens e treinamento de modelos baseados em classificação de imagens e detecção de objetos.

De forma geral, os objetivos propostos foram atingidos. Foi possível estruturar um conjunto de dados original para a raça Guzerá e aplicar modelos de CNNs de detecção e classificação para a predição automatizada do ECC.

Apesar disso, algumas limitações foram observadas durante o desenvolvimento do trabalho. O tamanho reduzido do conjunto de dados e o desbalanceamento entre classes dificultaram o treinamento e limitaram a capacidade de generalização dos modelos. Adicionalmente, as imagens foram coletadas em ambiente não controlado, apresentando variações de iluminação, ângulo, distância e posicionamento dos animais, fatores que aumentam a complexidade do problema.

Os resultados obtidos demonstraram que o problema apresenta elevado grau de complexidade quando aplicado em condições reais de produção. Nos experimentos realizados com redes neurais para classificação, observou-se que a redução da granularidade da escala de escores contribuiu para o aumento da acurácia dos modelos, indicando que a distinção entre categorias mais amplas tende a ser mais consistente do que a classificação em intervalos mais refinados. Também foi possível identificar diferenças de desempenho entre arquiteturas e configurações de treinamento, evidenciando a influência de hiperparâmetros como taxa de aprendizado e tamanho do lote.

Como trabalhos futuros, sugere-se, sobretudo, a ampliação do conjunto de dados e uma distribuição mais equilibrada do número de imagens de cada classe de ECC. Além disso, deve-se investigar abordagens alternativas que não dependam exclusivamente de redes neurais convolucionais, bem como abordagens baseadas em um *pipeline* de múltiplos

estágios, no qual um modelo de detecção de objetos seja empregado para localizar a região de interesse do animal e, posteriormente, uma rede de classificação realize a predição do ECC. Por último, considerando a natureza subjetiva do processo de atribuição do ECC e a existência de diferentes níveis de concordância entre avaliadores, também se mostra promissora a investigação de técnicas de aprendizado com rótulos suaves (*soft labels*), nas quais a saída esperada do modelo represente uma distribuição de probabilidades entre classes em vez de um único rótulo discreto.

Por fim, espera-se que os resultados apresentados contribuam para o avanço de soluções computacionais aplicadas à pecuária de precisão e incentivem novas pesquisas voltadas à automatização da avaliação do Escore de Condição Corporal em bovinos da raça Guzerá.

Bibliografia

ACHOUR, B. et al. Image-based cow breed recognition using deep learning. In: *International Conference on Artificial Intelligence and Information Technology (ICAIIT)*. Ouargla: IEEE, 2020. p. 1–5.

ALVAREZ, J. R. et al. Body condition estimation on cows from depth images using convolutional neural networks. *Computers and Electronics in Agriculture*, v. 155, p. 12–22, 2018.

AMIT, Y.; FELZENSZWALB, P.; GIRSHICK, R. Object detection. In: IKEUCHI, K. (Ed.). *Computer Vision: A Reference Guide*. [S.l.]: Springer, 2020.

ARBEX, W. et al. Computação móvel aplicada à pecuária de precisão para a determinação do escore da condição corporal. In: SBIAGRO. *Anais do X Congresso Brasileiro de Agroinformática (SBIAGRO)*. Ponta Grossa, PR, 2015. ISBN 978-85-69929-00-0.

AZZARO, G. et al. Objective estimation of body condition score by modeling cow body shape from digital images. *Journal of Dairy Science*, v. 94, n. 4, p. 2126–2137, 2011.

BERCKMANS, D. Precision livestock farming technologies for welfare management in intensive livestock systems. *Revista Brasileira de Zootecnia*, v. 43, n. 6, p. 327–334, 2014.

BEWLEY, J. M. et al. Potential for estimation of body condition scores in dairy cattle from digital images. *Journal of Dairy Science*, v. 91, n. 9, p. 3439–3453, 2008.

BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

ÇEVIK, K. K. Deep learning based real-time body condition score classification system. *IEEE Access*, IEEE, v. 8, p. 213950–213957, 2020.

CHEN, C. et al. Recognizing dairy cow behavior based on deep learning and feature fusion. *Computers and Electronics in Agriculture*, v. 182, p. 106020, 2021.

COMINOTTE, A. et al. Automated computer vision system to predict body weight and average daily gain in beef cattle during feedlot finishing. *Computers and Electronics in Agriculture*, v. 169, p. 105196, 2020.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, Springer, v. 2, n. 4, p. 303–314, 1989.

FISCHER, A. et al. Rear shape in 3 dimensions summarized by principal component analysis is a good predictor of body condition score in holstein dairy cows. *Journal of Dairy Science*, v. 98, p. 4465–4476, 2015.

GONZALEZ, R. C.; WOODS, R. E. *Digital image processing*. 4. ed. New York: Pearson, 2018.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. ⟨http://www.deeplearningbook.org⟩.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 770–778.

HUANG, G. et al. Densely connected convolutional networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2017. p. 4700–4708.

HUANG, J. et al. Automatic prediction of dairy cow body condition score using 2d images and deep learning methods. *Computers and Electronics in Agriculture*, v. 166, p. 104982, 2019.

JOCHER, G.; CHAURASIA, A.; QIU, J. *Ultralytics Reference - ConfusionMatrix.process_batch*. 2023. Acessado em: 03 de julho de 2026. Disponível em: ⟨https://docs.ultralytics.com/reference/utils/metrics/#ultralytics.utils.metrics.ConfusionMatrix.process_batch⟩.

JOCHER, G.; CHAURASIA, A.; QIU, J. *Ultralytics YOLO*. 2023. Disponível em: ⟨https://github.com/ultralytics/ultralytics⟩.

JOCHER, G. et al. Ultralytics yolo26: Unified real-time end-to-end vision models. *arXiv preprint arXiv:2606.03748v1*, 2026.

JORDAN, K. et al. *Muon: An optimizer for hidden layers in neural networks*. 2024. Disponível em: ⟨https://kellerjordan.github.io/posts/muon/⟩.

KHANAM, R.; HUSSAIN, M. Yolov11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725v1*, 2024.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. [S.l.: s.n.], 2015.

KOJIMA, T. et al. Body condition score estimation of beef cows using three-dimensional body features derived from 3d camera data. *Livestock Science*, v. 256, p. 104816, 2022.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, ACM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, MIT Press, v. 1, n. 4, p. 541–551, 1989.

LIN, T.-Y. et al. Focal loss for dense object detection. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2017. p. 2980–2988.

LIU, D. et al. Automatic estimation of dairy cattle body condition score from depth image using ensemble model. *Biosystems Engineering*, v. 194, p. 16–27, 2020.

LOSHCHILOV, I.; HUTTER, F. Decoupled weight decay regularization. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. [S.l.: s.n.], 2019.

MAHONY, J. et al. Comparison of two- and three-dimensional imaging approaches for automated body condition scoring in dairy cows. *Animals*, v. 12, n. 23, p. 3302, 2022.

MARTINS, B. M. et al. Estimating body weight, body condition score, and type traits in dairy cows using three dimensional cameras and manual body measurements. *Livestock Science*, v. 236, p. 104054, 2020.

MATOS, J. P. C. d. et al. Validação do software e-score de análise de escore corporal. In: EMBRAPA GADO DE LEITE. Juiz de Fora, MG, 2016. Projeto: Sistema de monitoramento e inteligência para manejo de rebanhos leiteiros e automação em sistemas de produção de leite.

MilkPoint. *MilkPoint: Notícias, produção e mercado de leite no Brasil e no mundo*. 2026. Accessed: 2026-07-01. Disponível em: ⟨https://www.milkpoint.com.br⟩.

QIAO, Y. et al. Bovine individual identification based on deep learning and feature fusion. *Measurement*, v. 148, p. 106956, 2019.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. [S.l.: s.n.], 2016. p. 779–788.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

ROCHE, J. R. et al. Invited review: Body condition score and its association with dairy cow productivity, health, and welfare. *Journal of Dairy Science*, v. 92, n. 12, p. 5769–5801, 2009.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

SONG, X. et al. Automated body condition scoring of dairy cows using 3-dimensional feature extraction from multiple body regions. *Journal of Dairy Science*, v. 102, p. 4294–4308, 2019.

SPOLIANSKY, R. et al. Development of automatic body condition scoring using a low-cost 3-dimensional kinect camera. *Journal of Dairy Science*, v. 99, n. 9, p. 7714–7725, 2016.

SZELISKI, R. *Computer vision: algorithms and applications*. 2. ed. Cham: Springer Nature, 2022.

TKACHENKO, M. et al. *Label Studio: Data labeling software*. 2020. Open source software. Disponível em: ⟨https://github.com/HumanSignal/label-studio⟩.

WILDMAN, E. E. et al. A dairy cow body condition scoring system and its relationship to selected production characteristics. *Journal of Dairy Science*, v. 65, p. 495–501, 1982.

ZHOU, Y.-T.; CHELLAPPA, R. Computation of optical flow using a neural network. In: *IEEE 1988 International Conference on Neural Networks*. [S.l.]: IEEE, 1988. v. 2, p. 71–78.