UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

JOÃO VITOR DE AGUIAR SILVA

APLICATIVO PARA DETECÇÃO DE ANEMIA EM OVINOS

SOBRAL

2025

JOÃO VITOR DE AGUIAR SILVA

APLICATIVO PARA DETECÇÃO DE ANEMIA EM OVINOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Iális Cavalcante de Paula Junior.

SOBRAL

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58a Silva, João Vitor de Aguiar.
APLICATIVO PARA DETECÇÃO DE ANEMIA EM OVINOS / João Vitor de Aguiar Silva. – 2025.
72 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2025.
Orientação: Prof. Dr. Iális Cavalcante de Paula Junior.

1. Haemonchus contortus. 2. FAMACHA. 3. Aprendizado Profundo. 4. Aplicativo móvel. 5. Detecção automática. I. Título.

CDD 621.39

JOÃO VITOR DE AGUIAR SILVA

APLICATIVO PARA DETECÇÃO DE ANEMIA EM OVINOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: 04 de Agosto de 2025

BANCA EXAMINADORA

Prof. Dr. Iális Cavalcante de Paula Junior. (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Jarbas Joaci de Mesquita Sá Júnior
Universidade Federal do Ceará (UFC)

Profa. Dra. Jermana Lopes de Moraes
Universidade Federal do Ceará (UFC)

Dedico este trabalho primeiramente a Deus, à minha mãe Francisca Pereira de Aguiar, ao meu pai Roberto Correia da Silva, à minha tia Maria do Livramento Pereira e à minha avó Maria de Lourdes Pereira.

AGRADECIMENTOS

A Deus, minha eterna gratidão, por ser minha força em todos os instantes e pela sabedoria, paciência e coragem que me sustentaram ao longo de toda esta caminhada.

À minha mãe Francisca Pereira, ao meu pai Roberto Correia, à minha tia Maria do Livramento e à minha avó Maria de Lourdes, pelo amor, pela educação e pelo apoio incondicional, que moldaram quem sou hoje e tornaram realidade esta conquista.

Ao meu irmão Rafael de Aguiar e à minha prima Daniele Pereira, pelo cuidado constante e pela auxilio inestimável ao longo destes anos.

À minha namorada Cintia Carolina, que tive o prazer de conhecer durante esta jornada e cuja presença foi de crucial importância nos momentos difíceis.

Aos amigos que conheci desde o primeiro semestre em 2019, Emanuel Devid, Emanuel Valério, Jeferson Marques, Frank William, Igor Pierre, Vitor Hugo e Vinícius Costa, agradeço por animarem meus dias e ajudarem sempre que possível.

Aos amigos que fiz ao longo da graduação, Wendel Alves, Daniel Meneses, Jhonatas Jorge, Alexandre Gomes, Joaquim Gregório e José Ivo, minha gratidão pelas experiências incríveis dentro e fora do campus da UFC.

Aos amigos da minha cidade natal, Domenico Trévia, Tiago Alves, Isaac Maia e Ítalo Barros, que compartilharam tantas vivências antes e durante o curso.

Ao meu orientador, Prof. Dr. Ialis Cavalcante, sou profundamente grato pela condução, pelo zelo e pela inspiração desde minha a bolsa de pesquisa até a conclusão deste trabalho.

À Universidade Federal do Ceará, em especial ao corpo docente do campus de Sobral, pelo ensino de excelência e pelo estímulo contínuo à pesquisa e à inovação.

"O fim último do universo é o bem do entendimento, que é a verdade."

(São Tomás de Aquino)

RESUMO

A incidência de verminoses causadas pelo *Haemonchus contortus* (HC) tem gerado prejuízos significativos à ovinocultura, afetando diretamente a saúde dos rebanhos e reduzindo a produtividade dos criadores. Um dos métodos mais utilizados no diagnóstico da anemia associada a essa parasitose é o *Faffa Malan Chart* (FAMACHA©), que se baseia na avaliação visual da coloração da mucosa ocular para classificar os animais em cinco níveis. Apesar de sua simplicidade, o método apresenta limitações como a subjetividade da análise e a dependência de experiência prévia por parte do avaliador. Com o avanço das tecnologias para dispositivos móveis e das técnicas de Aprendizado Profundo (do inglês, *Deep Learning* (DL)), surge a possibilidade de automatizar esse processo, tornando-o mais preciso, padronizado e acessível. Neste trabalho, foi desenvolvido um Aplicativo Móvel (APP) para a plataforma Android, utilizando a linguagem de programação Kotlin em conjunto com a biblioteca Jetpack Compose. O sistema proposto realiza a detecção da região ocular, em seguida, sua classificação nos cinco níveis do FAMACHA©, fornecendo ao criador a indicação se o animal está saudável ou doente. A base de dados contém 165 imagens, distribuídas igualmente entre as cinco categorias definidas pelo FAMACHA©. Para a etapa de detecção, empregou-se a arquitetura YOLOv5, que alcançou o desempenho com mAP@0.5 de 0,995 no conjunto de testes com 17 amostras. Para a classificação, o melhor modelo apresentou F1-Score médio de 0,9026 $\pm$ 0,0515 com validação cruzada em cinco *folds*. O tempo de inferência aferido foi de 83 milissegundos para detecção e 20 milissegundos para classificação, validando o uso em dispositivos móveis. Em conclusão, o APP desenvolvido se mostra viável e eficaz como ferramenta de apoio no manejo sanitário de ovinos, contribuindo para a sustentabilidade da produção e para o uso racional de vermífugos.

Palavras-chave: *Haemonchus contortus*. FAMACHA. Aprendizado Profundo. Aplicativo móvel. Detecção automática.

ABSTRACT

Infections caused by *Haemonchus contortus* (HC) have caused significant losses in sheep farming, directly affecting herd health and reducing producer productivity. One of the most widely used methods for diagnosing anemia associated with this parasitosis is the Faffa Malan Chart (FAMACHA©), which is based on the visual assessment of the ocular mucosa coloration to classify animals into five levels. Despite its simplicity, the method presents limitations such as the subjectivity of the analysis and the dependence on prior experience by the evaluator. With advances in mobile device technologies and Deep Learning (DL) techniques, the possibility arises to automate this process, making it more accurate, standardized, and accessible. In this work, a mobile application (APP) was developed for the Android platform, using the Kotlin programming language together with the Jetpack Compose library. The proposed system performs ocular region detection, followed by classification into the five FAMACHA© levels, providing the producer with an indication of whether the animal is healthy or sick. The database contains 165 images, equally distributed among the five categories defined by FAMACHA©. For the detection stage, the YOLOv5 architecture was employed, achieving a mAP@0.5 of 0.995 on the test set with 17 samples. For classification, the best model presented a mean F1-Score of 0.9026 ± 0.0515 with five-fold cross-validation. The inference time measured was 83 milliseconds for detection and 20 milliseconds for classification, validating its use on mobile devices. In conclusion, the developed APP proved to be viable and effective as a support tool in the sanitary management of sheep, contributing to production sustainability and the rational use of anthelmintics.

Keywords: *Haemonchus contortus*. FAMACHA. Deep Learning. Mobile Application. Automated Detection.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AI	*Artificial Intelligence*
AP	*Average Precision*
APP	Aplicativo Móvel
BB	*Bounding Box*
BCE	Binary Cross-Entropy
CCE	Categorical Cross-Entropy
CNN	*Convolutional Neural Network*
CV	*Computer Vision*
DA	*Data Augmentation*
DL	*Deep Learning*
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
FAMACHA©	*Faffa Malan Chart*
HC	*Haemonchus contortus*
Ht	Hematócrito
IoU	*Intersection over Union*
LR	*learning rate*
mAP	*mean Average Precision*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
MSE	*Mean Squared Error*
NAS	*Neural Architecture Search*
NMS	*Non-Maximum Suppression*
PCA	*Principal Component Analysis*
R-CNN	*Region-based Convolutional Neural Network*
RGB	*Red, Green, Blue*
RNN	*Recurrent Neural Networks*
ROI	*Region of Interest*
SGD	*Stochastic Gradient Descent*
SVM	*Support Vector Machine*
TF	*TensorFlow*
TFLite	*TensorFlow Lite*

YOLO *You Only Look Once*

LISTA DE SÍMBOLOS

θ	Parâmetros do modelo
$J(\theta)$	Função de custo
$L(y_i, \hat{y}_i)$	Função de perda
η	Taxa de aprendizado
λ	Coeficiente de regularização
$R(\theta)$	Função de regularização
VP	Verdadeiros Positivos
FP	Falsos Positivos
FN	Falsos Negativos
VN	Verdadeiros Negativos
z	Potencial de ativação
w	Pesos sinápticos
b	Viés

SUMÁRIO

1 INTRODUÇÃO

Um problema recorrente para produtores de pequenos ruminantes, em especial ovinos, é a incidência da verminose provocada por *Haemonchus contortus* (HC). Essa parasitose causa anemia severa nos animais, levando à perda de peso, redução da produção de leite e lã, e, em casos graves, à morte. A hemoncose, doença causada por esse nematóide, gera grandes prejuízos econômicos para os criadores, afetando a rentabilidade das propriedades (BESIER *et al.*, 2016).

Foi constatado uma forte correlação entre a coloração da membrana mucosa ocular e o grau de anemia do animal. Com base nisso, foi desenvolvido o método *Faffa Malan Chart* (FAMACHA©), que utiliza um cartão ilustrativo que categoriza as mucosas em cinco intervalos, que vão de vermelho (indicando boa saúde) a branco (indicando anemia severa). Essa correlação sugere que uma avaliação precisa pode auxiliar na identificação de animais que necessitam de tratamento, permitindo intervenções mais eficazes e reduzindo a seleção de resistência aos antiparasitários (WYK; BATH, 2002).

Conforme observado na Figura 1, o cartão apresenta diferentes níveis de coloração da mucosa. Nos graus A(1) e B(2), a coloração é intensamente vermelha, indicando ausência ou mínima presença de anemia. O tratamento com vermífugo é recomendado a partir do grau C(3). Nos graus D(4) e E(5), a vermifugação torna-se essencial devido à palidez intensa da mucosa, sendo que no grau E(5) é recomendada suplementação alimentar. Essa abordagem reduz a invasividade para o animal e evita a prática indiscriminada de vermifugação do rebanho (CHAGAS *et al.*, 2007).

Figura 1 – Fotografia do cartão FAMACHA©.

Fonte: EMBRAPA (2024).

O método mais preciso para o diagnóstico de verminose em ovinos é o Hematócrito (Ht), que consiste em um exame laboratorial utilizado para medir a proporção de glóbulos vermelhos no sangue, detectando assim a anemia da cria. Embora confiável, este método pode ser demorado e apresentar custo elevado para o criador, especialmente em localidades humildes e de difícil acesso, o que frequentemente impede sua adoção para a avaliação da condição dos animais.

O método FAMACHA© foi desenvolvido para monitorar a anemia causada pelo parasita HC em ovinos e caprinos, por meio da avaliação da coloração da mucosa ocular. Contudo, é importante destacar que o FAMACHA© identifica a anemia de forma geral, independentemente da sua causa, o que implica que outras condições, como doenças ou deficiências nutricionais, também podem influenciar os resultados. Além disso, fatores como a sazonalidade das infecções, a qualidade da gestão da propriedade, incluindo taxas de lotação e a presença de outras espécies de nematóides, podem impactar a eficácia do método. A precisão da avaliação depende ainda do treinamento adequado dos usuários e da assistência técnica disponível, sendo essencial que as análises sejam realizadas com o suporte de profissionais de saúde animal. Por fim, a manutenção adequada do cartão FAMACHA©, incluindo sua proteção contra a luz e substituição anual, é fundamental para garantir a fidelidade das referências de cor (STOREY *et al.*, 2017).

Para mitigar as variáveis humanas no processo de análise da coloração da mucosa e democratizar o acesso ao diagnóstico, propõe-se a implementação de um Aplicativo Móvel (APP) que fornece, de forma instantânea, a indicação da condição clínica do animal (saudável ou doente). O APP captura uma imagem da mucosa ocular e, por meio de algoritmos baseados em técnicas de Visão Computacional (do inglês, *Computer Vision* (CV)) e Inteligência Artificial (do inglês, *Artificial Intelligence* (AI)), detecta e classifica o nível de coloração da mucosa, permitindo um diagnóstico preciso e automatizado.

1.1 Justificativa

A detecção precoce de anemia em ovinos é essencial para garantir a saúde e produtividade desses animais, especialmente entre pequenos produtores rurais que dependem diretamente da criação eficiente. A anemia, se não diagnosticada a tempo, pode comprometer o crescimento, a reprodução e a qualidade da carne e da lã. Tradicionalmente, o método FAMACHA© é utilizado para identificar anemia com base na coloração da mucosa ocular, mas esta técnica apresenta limitações. Como demonstrado por RABELO *et al.* (2023), a natureza subjetiva do

método frequentemente leva a falhas na detecção precoce da anemia e a variações entre diferentes avaliadores, resultando em diagnósticos pouco confiáveis que podem comprometer a eficácia dos tratamentos.

Diante dessas limitações, a utilização de técnicas de Aprendizado de Máquina (do inglês, *Machine Learning* (ML)) torna-se uma inovação promissora. Ao aplicar modelos de ML é possível aumentar a precisão e a eficiência na identificação de condições patológicas. Os modelos podem detectar sinais sutis de anemia que poderiam passar despercebidos no método tradicional, proporcionando uma abordagem mais robusta. Assim, o uso da AI oferece uma solução tecnológica avançada que complementa FAMACHA©, ajudando na tomada de decisão dos proprietários e técnicos responsáveis.

A escolha deste tema torna-se ainda mais relevante devido à parceria com a Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA), que facilita a coleta de dados e conecta o projeto diretamente às necessidades dos pequenos produtores rurais. Com a aplicação de tais tecnologias, espera-se aprimorar o manejo animal e promover a sustentabilidade da produção ovina. A pesquisa também abre portas para futuros estudos e inovações tecnológicas no campo da saúde animal, contribuindo para a modernização da agropecuária. Além disso, o potencial de coleta de dados do APP, como geolocalização, informações sobre vermífugos, datas de aplicação e outros dados relevantes, enriquece a análise e permite um estudo mais aprofundado da anemia em ovinos, sendo de grande valor tanto para os produtores quanto para os órgãos públicos responsáveis pela gestão sanitária.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um APP que automatize o cartão FAMACHA©, utilizando modelos de *Deep Learning* (DL) para detectar e classificar a mucosa de ovinos.

1.2.2 Objetivos Específicos

- Selecionar modelos de DL que sejam adequados para a detecção de objetos e classificação de imagens.
- Implementar e treinar os modelos selecionados utilizando um conjunto de dados.
- Avaliar o desempenho dos modelos de DL com base em métricas como acurácia,

precisão, *recall* e F1-score.

- Comparar os resultados obtidos para determinar qual modelo apresenta o melhor desempenho.
- Desenvolver uma interface amigável no APP que integre os modelos de DL selecionados.

1.3 Trabalhos Relacionados

Foi realizada uma pesquisa na literatura para identificar trabalhos que propõem APPs ou métodos que automatizam o processo de decisão baseado no cartão FAMACHA©. O levantamento priorizou aqueles que utilizam técnicas de Aprendizado Profundo (do inglês, DL), dado que este será o foco central do trabalho.

O trabalho de ALMEIDA (2021) investigou a aplicação de redes neurais profundas para segmentação e classificação de imagens da mucosa ocular de ovinos, com o objetivo de identificar sinais de anemia. Na segmentação, a U-Net apresentou o melhor desempenho, alcançando uma precisão de 97,29% com base no índice de similaridade de Jaccard. Na classificação, os modelos MobileNetV2 e ResNet18 se destacaram, ambos atingindo uma acurácia de 95,23%, com e sem normalização. O estudo utilizou um conjunto de 106 imagens coletadas na EMBRAPA Caprinos e Ovinos, localizada em Sobral-CE, e também resultou no desenvolvimento de um APP que integra os modelos, demonstrando bom desempenho nos testes realizados.

Em de Souza *et al.* (2023), foi desenvolvida uma aplicação Android para imitar o processo de decisão do veterinário no método FAMACHA©. O *software* não possui segmentação ou detecção automática da mucosa conjuntival, exigindo um recorte manual realizado pelo usuário. Foram utilizadas 317 imagens de mucosas coletadas em fazendas do sul do Brasil, que serviram como base de dados para avaliar o desempenho de dois classificadores: uma rede neural sequencial e uma Máquina de Vetores de Suporte (do inglês, *Support Vector Machine* (SVM)). O SVM apresentou o melhor desempenho, com precisão de 87%, demonstrando maior eficácia na classificação dos animais em saudáveis ou anêmicos. Esse resultado evidencia o potencial do sistema para apoiar pequenos produtores no manejo seletivo de parasitas.

ROCHA *et al.* (2025) propôs a automatização do método FAMACHA© por meio de visão computacional, utilizando o YOLOv8n para segmentar a mucosa ocular e o Random Forest para classificação, com base em 450 imagens coletadas em campo. A validação cruzada

revelou métricas como acurácia de 64%, precisão de 66%, *recall* de 77% e F1 Score de 71%, evidenciando o potencial da abordagem, apesar de limitações relacionadas à segmentação e variações de iluminação.

O presente trabalho diferencia-se dos anteriores ao realizar a detecção automática da mucosa ocular por meio de algoritmos de detecção de objetos, eliminando a necessidade de segmentações manuais ou recortes realizados pelo usuário, como observado em de Souza *et al.* (2023). Enquanto trabalhos como ALMEIDA (2021) e ROCHA *et al.* (2025) focam na segmentação da mucosa, esta pesquisa opta pela detecção direta, simplificando o processo. Nesse contexto, propõe-se comparar diversos classificadores baseados em Redes Neurais Convolucionais (do inglês, *Convolutional Neural Network* (CNN)), buscando aprimorar sua precisão e eficiência computacional.

2 FUNDAMENTAÇÃO TEÓRICA

No presente capítulo, apresenta-se o referencial teórico que fundamenta a compreensão dos temas abordados ao longo deste trabalho. São discutidos os principais conceitos técnicos necessários para o entendimento da proposta desenvolvida.

2.1 Sistema FAMACHA©

A Tabela 1 sintetiza os critérios adotados pelo sistema FAMACHA© para a classificação clínica dos animais, associando a coloração da mucosa ocular aos níveis de Ht. Isso permite determinar a gravidade da anemia e orientar a necessidade de intervenção. A gradação varia de 1 a 5, servindo como referência para decisões clínicas de ovinos e caprinos.

Tabela 1 – Classificação FAMACHA segundo valores de Ht

Grau FAMACHA	Coloração da mucosa	Hematócrito (%)	Atitude clínica
1	Vermelho robusto	> 27	Não tratar
2	Vermelho rosado	23 a 27	Não tratar
3	Rosa	18 a 22	Tratar
4	Rosa pálido	13 a 17	Tratar
5	Branco	< 13	Tratar

Fonte: Adaptado de CHAGAS *et al.* (2007).

2.2 Aprendizado de Máquina

O ML é uma subárea da AI que desenvolve algoritmos capazes de aprender e melhorar automaticamente a partir de dados e experiências anteriores, sem programação explícita para tarefas específicas. Este processo é fundamental para criar sistemas inteligentes que se adaptam a novas informações. O ML é amplamente utilizado em aplicações como reconhecimento de padrões, classificação de dados, recomendações personalizadas e diagnósticos médicos, sendo essencial para a análise de grandes volumes de dados e a tomada de decisões eficazes (MONARD; BARANAUSKAS, 2003).

Os algoritmos de ML são classificados em três categorias principais: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço. O aprendizado supervisionado utiliza dados rotulados para prever resultados, com exemplos como regressão linear e SVMs. O aprendizado não supervisionado busca identificar padrões em dados não rotulados, utilizando técnicas como *K-means* e Análise de Componentes Principais (do inglês,

Principal Component Analysis (PCA)). O aprendizado por reforço envolve um agente que aprende a maximizar recompensas através da interação com um ambiente dinâmico, refletindo a diversidade do ML (MONARD; BARANAUSKAS, 2003).

2.2.1 Função de Perda

Na construção de modelos de ML, a função de perda (*loss function*) desempenha um papel crucial, sendo responsável por quantificar o erro entre a saída prevista ($\hat{y}$) pelo modelo e o valor real (y) da variável de interesse (GOODFELLOW *et al.*, 2016). O resultado indica o quão bem o modelo está desempenhando em uma única amostra e orienta o processo de otimização ao fornecer uma direção para o ajuste dos parâmetros.

A partir da função de perda, define-se a *função de custo* (*cost function*), que agrega as perdas individuais ao longo do conjunto de treinamento por meio de uma média:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i), \tag{2.1}$$

em que θ representa os parâmetros do modelo, n é o número de amostras e $L(y_i, \hat{y}_i)$ é a perda associada à i-ésima amostra.

Para problemas de regressão, utiliza-se frequentemente o Erro Quadrático Médio (do inglês, *Mean Squared Error* (MSE)). Ele penaliza erros de forma quadrática, tornando-se sensível a *outliers*.

$$L_{\text{MSE}}(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2. \tag{2.2}$$

Em tarefas de classificação binária, a função de perda comumente utilizada é a Entropia Cruzada Binária (do inglês, Binary Cross-Entropy (BCE)), definida para uma amostra como:

$$L_{\text{BCE}}(y_i, \hat{y}_i) = -\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right], \tag{2.3}$$

em que $\hat{y}_i$ é a probabilidade prevista da classe positiva.

Para classificação multiclasse com codificação *one-hot*, utiliza-se a Entropia Cruzada Categórica (do inglês, Categorical Cross-Entropy (CCE)):

$$L_{\text{CCE}}(y_i, \hat{y}_i) = -\sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}), \tag{2.4}$$

em que C representa o número total de classes, $y_{i,c}$ é o indicador binário (0 ou 1) da classe verdadeira, e $\hat{y}_{i,c}$ é a probabilidade prevista para a classe c (geralmente obtida pela função *softmax*).

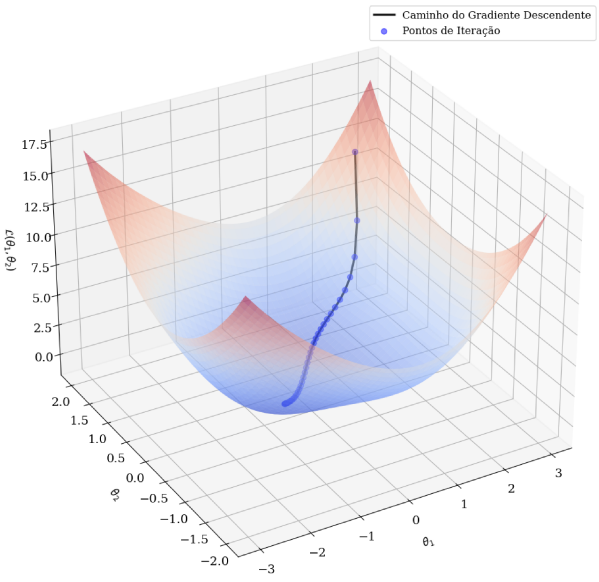
2.2.2 Otimização

A otimização no contexto de ML é responsável por ajustar os parâmetros θ com o objetivo de minimizar a função de custo $J(\theta)$ definida na Subseção 2.2.1. O principal algoritmo para esta minimização é o Gradiente Descendente Estocástico (do inglês, *Stochastic Gradient Descent* (SGD)), que atualiza os parâmetros utilizando o gradiente da função de perda individual:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla_\theta L(y_i, \hat{y}_i), \tag{2.5}$$

em que η é a taxa de aprendizado (do inglês, *learning rate* (LR)), $\nabla_\theta L(y_i, \hat{y}_i)$ é o gradiente da função de perda em relação aos parâmetros, e t denota a iteração atual. A Figura 2 ilustra este processo.

Figura 2 – Visualização do Gradiente Descendente.



Fonte: Próprio autor.

Desenvolvimentos recentes introduziram otimizadores adaptativos que melhoram a eficiência do SGD básico (FEDORENKO, 2019):

- Adam (*Adaptive Moment Estimation*): Adapta individualmente a LR para cada parâmetro usando estimativas de primeiro e segundo momento dos gradientes.
- RMSProp (*Root Mean Square Propagation*): Normaliza a LR pela média móvel exponencial dos quadrados dos gradientes.

2.3 Aprendizado Profundo

O DL é uma subárea do ML que se destaca pelo uso de redes neurais artificiais com múltiplas camadas, conhecidas como redes neurais profundas. Essas redes, inspiradas na arquitetura do cérebro humano, são compostas por camadas hierárquicas de neurônios artificiais, onde cada camada processa e extrai características de níveis crescentes de abstração a partir dos dados de entrada. A capacidade dessas redes de modelar e resolver problemas complexos e não lineares, como reconhecimento de imagens, processamento de linguagem natural e jogos de tabuleiro, posiciona o DL como uma abordagem superior em comparação com métodos tradicionais de ML (TAVANAEI *et al.*, 2019).

A adoção generalizada do DL pode ser atribuída à sua eficiência em lidar com grandes volumes de dados e em aprender representações sofisticadas, especialmente em tarefas que exigem alta precisão. Redes Neurais Convolucionais (do inglês, CNN) e Redes Neurais Recorrentes (do inglês, *Recurrent Neural Networks* (RNN)) são exemplos proeminentes dessa tecnologia, cada uma especializada em diferentes tipos de dados e aplicações (GUPTA *et al.*, 2022).

2.3.1 Perceptron

Proposto por Rosenblatt (1958), o Perceptron é um modelo de classificador linear binário. Ele representa a unidade fundamental das redes neurais artificiais. Nesse modelo, o vetor de entrada $x = [x_1, x_2, \ldots, x_n]$ é combinado linearmente com os pesos sinápticos $w = [w_1, w_2, \ldots, w_n]$ e um viés b, conforme a Equação 2.6. O resultado z, chamado de potencial de ativação, é então passado por uma função de ativação do tipo degrau (*step function*), que determina a saída binária do neurônio, conforme mostrado na Equação 2.7. Esse tipo de resposta binária reflete o funcionamento de um neurônio biológico, que apenas é ativado quando a soma

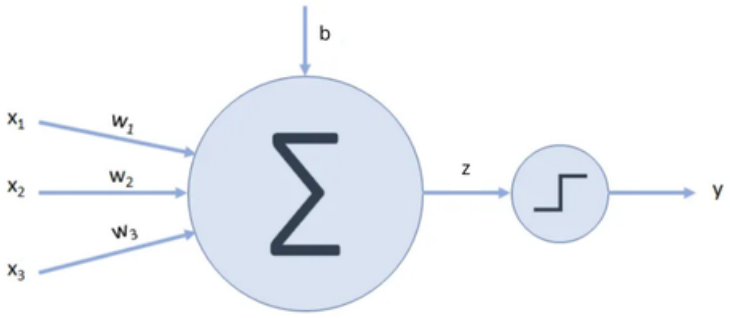
ponderada de seus estímulos de entrada supera um determinado limiar.

$$z = \sum_{i=1}^{n} w_i x_i + b, \tag{2.6}$$

$$y = \begin{cases} 1, & \text{se } z \geq 0 \\ 0, & \text{se } z < 0. \end{cases} \tag{2.7}$$

Um diagrama clássico de Perceptron é apresentado na Figura 3, seguindo a arquitetura *feed-forward*. Isso significa que as entradas são propagadas de forma unidirecional, processadas e transmitidas para a saída. Caracterizando um fluxo de processamento da esquerda para a direita.

Figura 3 – Diagrama de um Perceptron.



Fonte: Adaptado de ARAUJO (2020).

2.3.2 Função de Ativação

As funções de ativação são operadores matemáticos não lineares aplicados à combinação ponderada das entradas de cada neurônio, conferindo à arquitetura capacidade de modelar relações complexas e altamente não lineares presentes em dados reais. Sem essas não linearidades, a composição de múltiplas camadas redundaria numa única transformação afim, restringindo severamente o poder de representação do modelo e inviabilizando o aprendizado de padrões sofisticados (GOODFELLOW *et al.*, 2016).

A seguir, apresentam-se as principais funções de ativação utilizadas em redes neurais:

- Sigmoidal (*Logistic Sigmoid*): mapeia a entrada para o intervalo $(0,1)$, sendo frequentemente utilizada em classificações binárias.

$$\sigma(z) = \frac{1}{1+e^{-z}}. \tag{2.8}$$

- Tangente hiperbólica (*Hyperbolic Tangent*): semelhante à sigmoide, porém centrada na origem, mapeando em $(-1,1)$.

$$\tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}. \tag{2.9}$$

- ReLU (*Rectified Linear Unit*): ativa somente valores positivos, promovendo esparsidade e reduzindo o custo computacional.

$$\text{ReLU}(z) = \max(0,z). \tag{2.10}$$

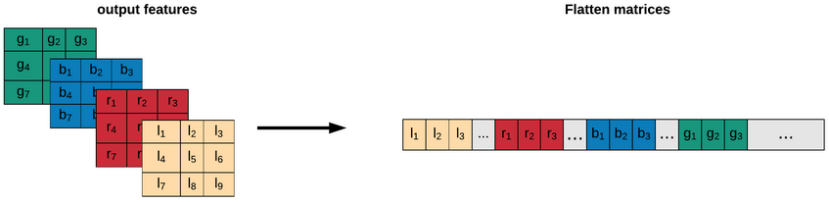
- Softmax: transforma um vetor de valores reais em uma distribuição de probabilidade sobre classes, sendo amplamente empregada na camada de saída de classificadores multiclasse.

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}. \tag{2.11}$$

2.4 Perceptron Multicamadas (MLP)

Um modelo clássico de rede neural é o Perceptron Multicamadas (do inglês, *Multilayer Perceptron* (MLP)), que aceita como entrada um vetor de características unidimensional. No caso de imagens, que possuem canais de cores Vermelho, Verde e Azul (do inglês, *Red, Green, Blue* (RGB)), é necessário transformar a imagem em um único vetor, em um processo conhecido como *flattening* (SILVA, 2024). A Figura 4 ilustra a operação de flattening, onde um vetor multidimensional é convertido em um vetor 1D.

Figura 4 – Operação de *flattening*.

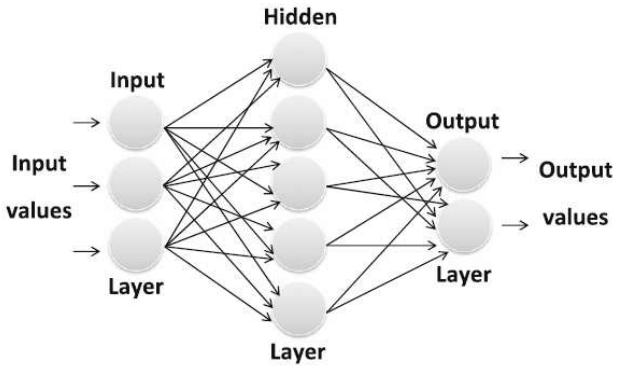


Fonte: Cano *et al.* (2018).

O achatamento da imagem resulta em um número enorme de parâmetros a serem treinados. Por exemplo, uma imagem de 240 × 240 *pixels*, com três canais de cor, geraria

um vetor de 172.800 posições. Dependendo do número de neurônios na camada oculta, a quantidade de parâmetros a serem ajustados pode se tornar extremamente alta. Como se pode observar na Figura 5, as camadas da rede são totalmente conectadas às camadas seguintes. Outra desvantagem do MLP é sua falta de invariância a perturbações, como translações, rotações, escalas e mudanças na posição do objeto de interesse. Devido à sua arquitetura, que aprende a posição exata de cada *pixel*, pequenas variações podem comprometer sua capacidade de avaliação (SILVA, 2024).

Figura 5 – Arquitetura de uma MLP.



Fonte: Al-Naymat *et al.* (2016).

2.5 Redes Neurais Convolucionais (CNN)

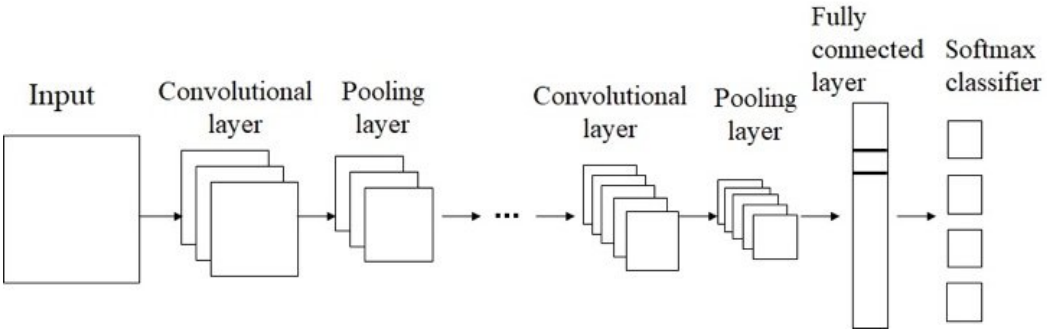
As CNNs são uma classe de redes neurais profundas que se destacam no processamento de dados visuais. Ao aplicar filtros sobre imagens, as CNNs extraem características locais, como bordas, texturas e formas, e constroem representações cada vez mais abstratas em camadas sucessivas. Essa arquitetura permite que as CNNs aprendam hierarquias de características de forma automática, tornando-as ideais para tarefas como classificação de imagens, detecção de objetos e segmentação semântica. A capacidade das CNNs de generalizar para novos dados e sua alta precisão tornaram-nas ferramentas essenciais em diversas áreas, como CV, processamento de linguagem natural e análise de dados (SINGH *et al.*, 2023).

2.5.1 Arquitetura das CNN

As CNNs e MLPs são redes neurais que processam dados de forma distinta. Enquanto as MLPs têm limitações em variações espaciais, as CNNs se destacam pela sua capacidade de extrair características locais da imagem. Essa capacidade torna as CNNs mais robustas a

operações como rotação, escala e translação. A estrutura básica de uma CNN é ilustrada na Figura 6, composta por três camadas principais: convolução, *pooling* e camada totalmente conectada.

Figura 6 – Estrutura básica de uma CNN.



Fonte: You *et al.* (2020).

2.5.1.1 Camadas Convolucionais

As camadas convolucionais aplicam filtros (ou *kernels*) à imagem de entrada com o objetivo de extrair características relevantes, como bordas, texturas e formas. Cada *kernel* é uma pequena matriz de pesos de tamanho $k \times k \times D$, onde $k \times k$ representa a dimensão espacial do filtro e D corresponde à profundidade do canal da imagem (por exemplo, 3 para imagens RGB). Durante a operação de convolução, o *kernel* percorre a imagem realizando multiplicações ponto a ponto com a região correspondente, e os resultados são somados para gerar um valor único que compõe o mapa de ativação. O processo resulta em um valor único que compõe o mapa de ativação. A utilização de múltiplos *kernels* permite que a camada convolucional produza diversos mapas de características, capturando diferentes aspectos da imagem.

As camadas iniciais da rede capturam características de baixo nível, como bordas e texturas, enquanto as camadas mais profundas identificam padrões mais complexos. Os parâmetros dos *kernels* são ajustados durante o processo de treinamento, permitindo à rede otimizar os filtros para tarefas específicas (HIJAZI *et al.*, 2015).

Para ilustrar o funcionamento da operação de convolução, considera-se uma matriz de entrada I com dimensões 3×3 e um *kernel* K com dimensões 2×2, definidos, respectivamente em (2.12) e (2.13).

$$I = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \tag{2.12}$$

$$K = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}. \tag{2.13}$$

Neste exemplo, K percorre a matriz de entrada I de forma sequencial, sendo aplicado sobre todas as submatrizes 2×2 extraídas de I. Conforme iliustrado na Figura 7, em cada aplicação realiza-se o produto elemento a elemento seguido da soma dos resultados, produzindo um valor escalar que compõe o mapa de ativação.

- Primeira aplicação: O kernel é sobreposto aos quatro elementos no canto superior esquerdo da matriz I:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \odot \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} = (1 \cdot 1) + (1 \cdot 0) + (0 \cdot 1) + (1 \cdot 0) = 1 + 0 + 0 + 0 = 1$$

- Segunda aplicação: O kernel é deslocado uma coluna à direita:

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \odot \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} = (1 \cdot 1) + (0 \cdot 0) + (1 \cdot 1) + (1 \cdot 0) = 1 + 0 + 1 + 0 = 2$$

- Terceira aplicação: O kernel é aplicado na linha inferior, reiniciando pela primeira coluna:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \odot \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} = (0 \cdot 1) + (1 \cdot 0) + (1 \cdot 1) + (0 \cdot 0) = 0 + 0 + 1 + 0 = 1$$

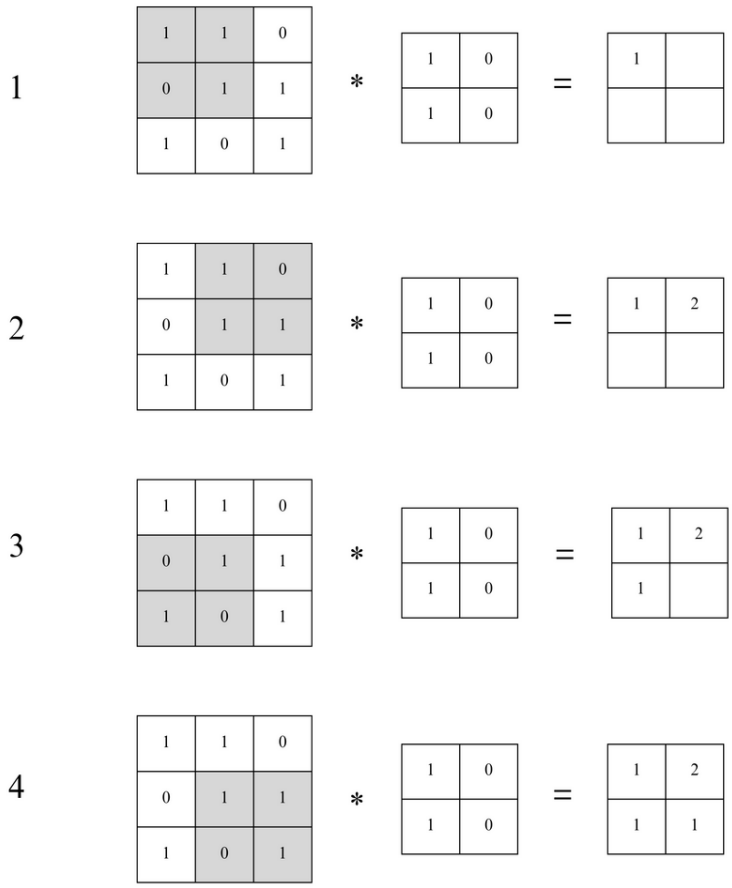
- Quarta aplicação: O kernel se desloca novamente para a coluna à direita:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \odot \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} = (1 \cdot 1) + (1 \cdot 0) + (0 \cdot 1) + (1 \cdot 0) = 1 + 0 + 0 + 0 = 1$$

Com isso, o mapa de ativação resultante é dado por:

$$A = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix} \tag{2.14}$$

Figura 7 – Etapas da operação de convolução sobre a matriz de entrada I com o kernel K.



Fonte: Próprio autor.

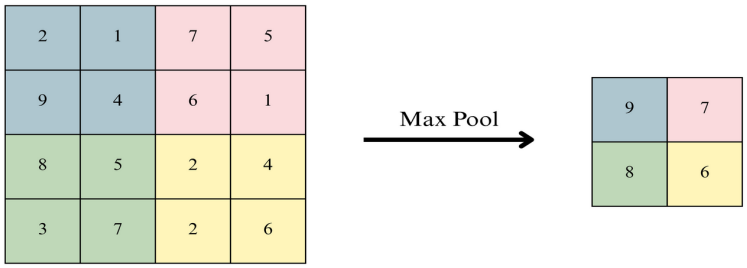
2.5.1.2 Camada de Pooling

Após a extração de características relevantes por meio da aplicação de filtros, a camada de *pooling* desempenha um papel fundamental na redução da dimensionalidade dos mapas de características gerados. Essa camada tem como objetivo diminuir a dimensão espacial das representações, o que resulta em uma redução significativa na quantidade de computação necessária e nos pesos a serem ajustados durante o treinamento. A redução não apenas acelera o processo de treinamento, mas também ajuda a preservar as características mais importantes, tornando a rede mais robusta a variações na entrada (BHATT *et al.*, 2021).

Existem diferentes métodos de *pooling*, sendo os mais comuns o *max pooling* e o *average pooling*. No *max pooling*, um *kernel* é movido sobre o mapa de características, e o valor máximo de cada região é selecionado, o que ajuda a manter as características mais proeminentes. Por outro lado, o *average pooling* calcula a média dos valores na vizinhança, resultando em uma representação mais suavizada.

O filtro de *pooling* atua de forma análoga ao filtro convolucional, deslocando o *kernel* sobre o mapa de características com um deslocamento definido pelo parâmetro de passo (*stride*). Em cada posição da janela, é aplicada uma operação de agregação (*max pooling* ou *average pooling*). O resultado obtido em cada região é então armazenado em uma matriz de saída, cuja resolução espacial é inferior à do mapa original. A Figura 8 ilustra o processo utilizando a técnica de *max pooling* com um *kernel* de 2×2 e *stride* igual a 2.

Figura 8 – Operação de *Max Pooling*.



Fonte: Próprio autor.

Além das técnicas de *pooling* locais, existem estratégias de *pooling* globais, como a camada *Global Average Pooling*. A agregação é aplicada sobre toda a extensão espacial de cada mapa de características, calculando a média de seus valores e reduzindo-o a um único escalar. A camada pode ser empregada como alternativa à utilização de camadas totalmente conectadas, assim como à camada *flatten*. Sua aplicação reduz significativamente o número de parâmetros do modelo e diminui o risco de *overfitting*.

2.5.1.3 Camada Totalmente Conectada

A camada totalmente conectada é uma das etapas finais em uma CNN. Trata-se de uma camada à qual cada neurônio está conectado a todos os neurônios da camada anterior, realizando uma soma ponderada das entradas. Essa operação utiliza todas as características extraídas nas camadas precedentes para gerar a saída final. O principal objetivo da camada totalmente conectada é combinar as informações para produzir uma decisão definitiva sobre a classe do objeto ou padrão reconhecido (HIJAZI *et al.*, 2015). Sua estrutura é similar à de um MLP, sendo responsável pela classificação final.

2.6 Técnicas de Regularização

Um problema comum no ML é desenvolver um algoritmo que apresente bom desempenho tanto nos dados de treinamento quanto nos de teste. A regularização é uma técnica utilizada para reduzir os erros de teste, geralmente às custas de um aumento no erro de treinamento. Seu principal objetivo é reduzir a complexidade excessiva dos modelos e auxiliar a formular uma função mais simples, promovendo, assim, a generalização.

Dentre as técnicas de regularização mais empregadas em DL, destaca-se o *dropout*, que consiste em desativar aleatoriamente uma fração dos neurônios durante o treinamento. Com o intuito de evitar coadaptações excessivas entre unidades de processamento, obrigando cada neurônio a contribuir de forma autônoma e a incorporar informações de todos os seus sinais de entrada. Após o treinamento, todos os neurônios permanecem ativos. Em consequência, obtém-se uma arquitetura mais robusta e com maior capacidade de generalização (GÉRON, 2019).

Considerando a função de custo $J(\theta)$ definida na Subseção 2.2.1 e o processo de otimização discutido posteriormente, técnicas de regularização podem ser incorporadas ao modelo por meio da adição de um termo penalizador à função objetivo. A nova formulação assume a forma:

$$O(\theta) = J(\theta) + \lambda R(\theta), \tag{2.15}$$

em que λ é um hiperparâmetro que regula a intensidade da penalização, e $R(\theta)$ representa a função de regularização. As abordagens mais comuns para essa penalização baseiam-se nas normas ℓ_2 e ℓ_1, conhecidas respectivamente como regularização Ridge e Lasso. A regularização ℓ_2, definida por

$$R_{\mathrm{L2}}(\theta) = \|\theta\|_2^2 = \sum_{i=1}^{n} \theta_i^2, \tag{2.16}$$

impõe uma penalidade quadrática sobre os parâmetros e favorece soluções com pesos menores, embora não necessariamente nulos. Esse tipo de regularização, associado a uma distribuição normal como *prior* sobre os parâmetros, reduz a sensibilidade a valores extremos e melhora a estabilidade numérica do treinamento. Por outro lado, a regularização ℓ_1 é expressa como

$$R_{\mathrm{L1}}(\theta) = \|\theta\|_1 = \sum_{i=1}^{n} |\theta_i|, \tag{2.17}$$

e introduz uma penalização linear que promove esparsidade no vetor de parâmetros, zerando coeficientes menos relevantes. Essa característica torna a regularização L1 especialmente útil quando se deseja realizar seleção de variáveis de forma implícita.

2.7 Métricas de Avaliação

A avaliação do desempenho de modelos de ML constitui um aspecto fundamental para compreender sua capacidade de generalização e eficácia. Diversas métricas clássicas são empregadas na literatura para quantificar o desempenho de modelos, tanto em tarefas de classificação quanto em detecção de objetos.

2.7.1 Acurácia

A acurácia representa a proporção de predições corretas sobre o total de amostras avaliadas, considerando os verdadeiros positivos (VP) e verdadeiros negativos (VN).

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.18}$$

2.7.2 Precisão

A precisão (*precision*) quantifica a proporção de VP em relação ao total de instâncias previstas como positivas, ou seja, a soma entre verdadeiros positivos e falsos positivos (FP):

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{2.19}$$

2.7.3 Recall

A revocação (*recall*) mede a proporção de instâncias positivas corretamente identificadas, isto é, a razão entre VP e a soma entre verdadeiros positivos e falsos negativos (FN):

$$\text{Recall} = \frac{VP}{VP + FN} \tag{2.20}$$

2.7.4 F1-Score

O F1-Score é a média harmônica entre precisão e revocação, sendo especialmente útil em cenários com classes desbalanceadas, pois equilibra ambas as métricas:

$$\text{F1-Score} = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{2.21}$$

2.7.5 Matriz de Confusão

A matriz de confusão é uma representação tabular do desempenho do modelo, relacionando as classes reais com as predições realizadas. Cada linha representa uma classe real e cada coluna uma classe predita:

Tabela 2 – Estrutura de uma Matriz de Confusão para classificação binária.

	Predito: Positivo	Predito: Negativo
Real: Positivo	VP	FN
Real: Negativo	FP	VN

Fonte: Próprio autor.

2.7.6 Interseção sobre União (IoU)

A métrica Interseção sobre União (do inglês, *Intersection over Union* (IoU)) é fundamental para avaliar a qualidade das detecções. Ela quantifica a sobreposição entre a caixas delimitadoras (do inglês, *Bounding Box* (BB)) predita pelo modelo e a caixa real (*ground truth*), sendo expressa pela equação abaixo:

$$\text{IoU} = \frac{\text{Área de Interseção}}{\text{Área de União}} \tag{2.22}$$

O valor da IoU varia de 0 a 1, sendo que o 1 representa sobreposição perfeita entre a predição e a anotação. Em aplicações práticas, é comum estabelecer um limiar, como 0,5, para definir se uma detecção será considerada VP. Quando a IoU é igual ou superior a esse valor, a predição é classificada como VP. Caso contrário, é tratada como FP em relação à anotação correspondente. A definição desse limiar é uma escolha metodológica que deve estar alinhada aos objetivos do experimento. Valores mais elevados exigem maior precisão na localização, enquanto valores menores permitem maior tolerância a variações na posição ou nas dimensões da caixa delimitadora.

2.7.7 Average Precision (AP)

A precisão média (do inglês, *Average Precision* (AP)) é uma métrica que avalia o desempenho de modelos de detecção de objetos em uma única classe. Ela sintetiza a relação entre precisão e *recall*, considerando diferentes limiares de confiança. A AP é calculada com base na curva *Precision-Recall*, a qual evidencia o equilíbrio entre identificar corretamente todos os objetos presentes na imagem e evitar detecções incorretas.

No PASCAL VOC, a AP é obtida por meio de uma interpolação da curva *Precision–Recall* em 11 pontos igualmente espaçados, abrangendo valores de revocação de 0 a 1 com incremento de 0,1 (EVERINGHAM *et al.*, 2010). Essa abordagem suaviza as variações pontuais na curva e facilita a comparação entre diferentes modelos. Em contraste, o benchmark Microsoft COCO adota uma metodologia que calcula a AP integrando continuamente a curva *Precision–Recall* com um limiar fixo de IoU igual a 0,5 (LIN *et al.*, 2014), proporcionando uma avaliação mais detalhada da precisão do modelo.

2.7.8 Mean Average Precision (mAP)

A precisão média média (do inglês, *mean Average Precision* (mAP)) é calculada como a média das AP obtidas para cada classe do conjunto de dados, conforme expressa na Equação 2.23:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i, \tag{2.23}$$

em que N representa o número total de classes e AP_i corresponde à AP da classe i. Valores elevados de mAP indicam que o modelo apresenta desempenho satisfatório na detecção de objetos em diversas categorias. Em contrapartida, valores reduzidos podem revelar limitações específicas, como dificuldades na identificação de objetos pequenos, oclusos ou com baixa diferenciação visual.

Existem estratégias distintas para o cálculo da mAP, destacando-se duas variações principais:

- mAP@0,5: cálculo da média das APs usando um limiar fixo de IoU igual a 0,5, conforme o protocolo do PASCAL VOC. Essa configuração é mais permissiva quanto a erros de localização.

- mAP@[0,5:0,95]: média das APs obtidas em múltiplos limiares de IoU de 0,5 a 0,95, com incrementos de 0,05, metodologia adotada pelo COCO. Essa abordagem é mais rigorosa e exige maior precisão.

2.7.9 Validação Cruzada

A validação cruzada *k-fold* é uma técnica fundamental em ML para a avaliação de modelos. Seu princípio consiste em dividir o conjunto de dados em k subconjuntos (*folds*) de tamanho aproximadamente igual, nos quais $k-1$ *folds* são utilizados para treinamento e um *fold* para teste, repetindo esse processo k vezes com diferentes *folds* de teste. Essa abordagem cíclica permite que todos os dados sejam utilizados tanto para treinamento quanto para teste, proporcionando uma estimativa mais confiável do desempenho do modelo em comparação a métodos como a simples divisão entre treinamento e teste (*hold-out*) (GÓRRIZ *et al.*, 2024).

2.8 Modelos de CNN

A aplicação de CNNs em dispositivos com recursos computacionais limitados, como *smartphones* e microcontroladores, tem impulsionado o desenvolvimento de arquiteturas otimizadas que equilibram precisão e eficiência computacional. Modelos como o MobileNet, EfficientNet e NASNet foram projetados com o objetivo de manter altos níveis de acurácia, ao mesmo tempo em que reduzem significativamente o número de parâmetros e o custo computacional. Elas possibilitam a realização de inferências em tempo real, mesmo em ambientes com restrições de *hardware*, tornando-se adequadas para aplicações embarcadas e móveis.

2.8.1 MobileNetV2

O MobileNetV2 é uma arquitetura de CNN projetada para dispositivos móveis. Como destacado por Sandler *et al.* (2018), esse modelo introduz duas inovações principais: *inverted residuals* (resíduos invertidos) e *linear bottlenecks* (gargalos lineares). Os *inverted residuals* invertem a estrutura tradicional dos blocos residuais, conectando camadas estreitas para reduzir o consumo de memória, enquanto os *linear bottlenecks* removem as funções de ativação em camadas de baixa dimensionalidade, a fim de preservar informações. Combinados com convoluções *depthwise separable*, esses mecanismos permitem que o MobileNetV2 alcance um equilíbrio entre acurácia e eficiência computacional.

A arquitetura do MobileNetV2, conforme apresentada na Tabela 3, é composta por uma sequência de blocos *bottleneck* que seguem o padrão de expansão e projeção. A primeira camada é uma convolução tradicional 3×3 com *stride* 2, seguida por uma série de blocos *inverted residual* com diferentes fatores de expansão. A rede finaliza com uma convolução 1×1, *global pooling* e uma camada de classificação.

Tabela 3 – Arquitetura da MobileNetV2.

Entrada	Operação	t	c	n	s
$224^2 \times 3$	`conv2d`	-	32	1	2
$112^2 \times 32$	`bottleneck`	1	16	1	1
$112^2 \times 16$	`bottleneck`	6	24	2	2
$56^2 \times 24$	`bottleneck`	6	32	3	2
$28^2 \times 32$	`bottleneck`	6	64	4	2
$14^2 \times 64$	`bottleneck`	6	96	3	1
$14^2 \times 96$	`bottleneck`	6	160	3	2
$7^2 \times 160$	`bottleneck`	6	320	1	1
$7^2 \times 320$	`conv2d 1×1`	-	1280	1	1
$7^2 \times 1280$	`avgpool 7×7`	-	-	1	-
$1 \times 1 \times 1280$	`conv2d 1×1`	-	k	-	-

Fonte: Adaptado de Sandler *et al.* (2018).

Na tabela, as colunas representam:

- t: fator de expansão dos canais;
- c: número de canais de saída;
- n: número de repetições do bloco;
- s: *stride* da convolução.

2.8.2 EfficientNet

O EfficientNet, introduzido por Tan e Le (2019), é uma família de arquiteturas de CNN projetada para maximizar a eficiência computacional e a precisão. Seu principal diferencial está no método de *compound scaling* (escalonamento composto), que equilibra de forma conjunta a profundidade, a largura e a resolução da rede. A abordagem parte de uma arquitetura base, o EfficientNet-B0, obtida por meio de uma busca neural por arquiteturas (do inglês, *Neural Architecture Search* (NAS)). A partir do EfficientNet-B0, aplicam-se coeficientes de escalonamento (α, β, γ) para gerar modelos mais complexos, variando de B1 a B7, que mantêm alta eficiência. Internamente, o modelo combina blocos MBConv com módulos *squeeze-and-excitation* (compressão e excitação) e convoluções *depthwise separable* (separáveis em profundidade), reduzindo o custo computacional sem comprometer o desempenho.

A Tabela 4 detalha a arquitetura do EfficientNet-B0, que serve como base para a família. Cada estágio utiliza blocos MBConv com fatores de expansão variados, seguidos por uma camada final de convolução 1×1 e *pooling* global. O escalonamento composto permite que versões posteriores (como o B7) atinjam a acurácia 84,3% no ImageNet.

Tabela 4 – Arquitetura do EfficientNet-B0.

Estágio	Operador	Resolução	Canais	Repetições
1	Conv 3×3	224×224	32	1
2	MBConv1, k3×3	112×112	16	1
3	MBConv6, k3×3	112×112	24	2
4	MBConv6, k5×5	56×56	40	2
5	MBConv6, k3×3	28×28	80	3
6	MBConv6, k5×5	14×14	112	3
7	MBConv6, k5×5	14×14	192	4
8	MBConv6, k3×3	7×7	320	1
9	Conv 1×1 + Pooling	7×7	1280	1

Fonte: Adaptado de Tan e Le (2019).

Na tabela, as variáveis e colunas informam:

- *MBConvN*: bloco *Mobile Inverted Bottleneck* com fator de expansão *N* e ativação Swish;
- *K*: tamanho do lado do *kernel* quadrado da convolução *depthwise*;
- *Resolução*: dimensões espaciais do mapa de características;
- Canais: número de mapas de ativação gerados ao final de cada estágio;
- Repetições: número de vezes que a célula é repetida em cada estágio.

2.8.3 NASNetMobile

O NASNetMobile é uma arquitetura de CNN projetada para dispositivos móveis e sistemas embarcados. Como destacado por Zoph *et al.* (2018), esse modelo é uma versão otimizada do NASNet original, reduzindo o custo computacional enquanto mantém uma precisão competitiva. O NASNetMobile utiliza blocos convolucionais modulares (*cells*) aprendidos automaticamente via busca por reforço, que são repetidos para formar a arquitetura final. Os blocos incorporam operações como convoluções separáveis (*separable convolutions*) e conexões residuais, otimizadas para eficiência.

A arquitetura do NASNetMobile, conforme apresentada na Tabela 5, é composta por dois tipos de células: *Normal Cell* (preserva dimensões espaciais) e *Reduction Cell* (reduz resolução e aumenta canais). Cada célula é construída a partir de blocos básicos que combinam operações convolucionais e não lineares, seguindo uma topologia aprendida durante o processo

de busca. A rede finaliza com uma camada de *pooling* global e classificação.

Tabela 5 – Arquitetura do NASNetMobile.

Estágio	Operador	Resolução	Canais	Repetições
1	`Conv3×3`	224×224	32	1
2	Normal Cell	112×112	44	1
3	Reduction Cell	112×112	44	2
4	Normal Cell	56×56	88	3
5	Reduction Cell	28×28	176	4
6	Normal Cell	14×14	352	3
7	Reduction Cell	7×7	704	3
8	Normal Cell	7×7	1408	1
9	`Conv1×1 + Pooling`	7×7	1056	1

Fonte: Adaptado de Team (2018).

As colunas e variáveis da tabela representam:

- *Normal Cell*: mantém as dimensões espaciais, realizando operações aprendidas (por exemplo, convoluções separáveis e conexões residuais);
- *Reduction Cell*: reduz a resolução por *stride* 2 e dobra o número de canais, conforme heurísticas de escalonamento;
- Canais: número de mapas de ativação gerados ao final de cada estágio;
- Repetições: número de vezes que a célula é repetida em cada estágio.

2.9 Detecção de Objetos

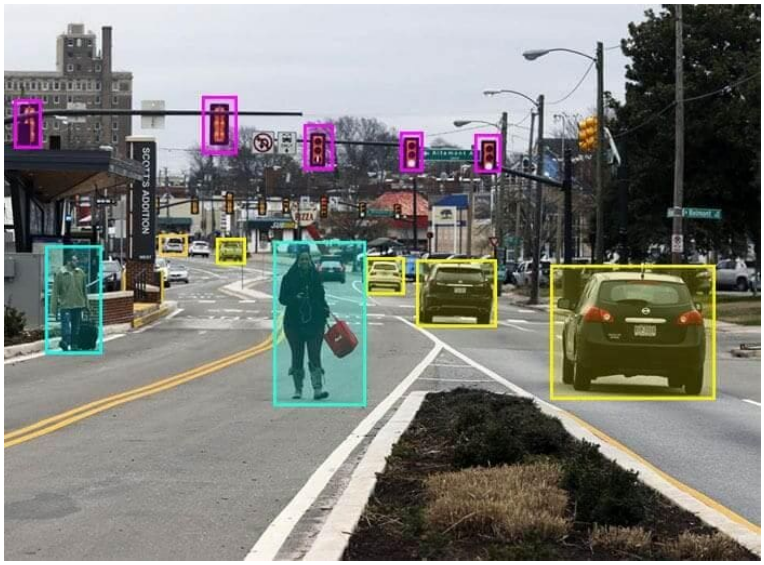
Dentro da visão computacional, uma atividade de extrema importância para problemas atuais é a detecção de objetos. Ela trata da detecção de instâncias de objetos visuais de uma determinada classe em imagens digitais. Em contextos nos quais é necessário não apenas localizar e classificar, mas também reconhecer e determinar suas posições exatas na cena, a detecção de objetos configura-se como uma abordagem eficiente (ZOU *et al.*, 2023).

Para representar a detecção de objetos em uma imagem, utilizam-se as BB. Essas caixas têm como objetivo indicar a localização dos objetos detectados. A BB é representada por um vetor de coordenadas no formato $[x_{\text{center}}, y_{\text{center}}, \mathit{width}, \mathit{height}]$, onde x_{center} e y_{center} correspondem às coordenadas normalizadas do centro da caixa delimitadora, enquanto *width* e *height* representam, respectivamente, a largura e a altura da caixa, também normalizadas. Na Figura 9 ilustra a detecção de varios objetos de classes distintas.

Os algoritmos de detecção de objetos podem ser categorizados em duas abordagens principais: *One-Stage* e *Two-Stage*. Os modelos *Two-Stage* diferenciam-se dos *One-Stage* pela

utilização de propostas de região, enquanto os *One-Stage* realizam a detecção diretamente em um único estágio. Essa distinção implica em diferenças significativas de desempenho. Modelos *One-Stage*, por operarem de forma mais simplificada, tendem a ser menos precisos, porém significativamente mais rápidos em termos computacionais. Em contrapartida, os modelos *Two-Stage*, ao incorporarem a geração e refinamento de regiões de interesse, alcançam maior precisão às custas de uma maior complexidade e latência.

Figura 9 – Exemplo de BBs.



Fonte: Sharma (2022).

2.10 YOLO

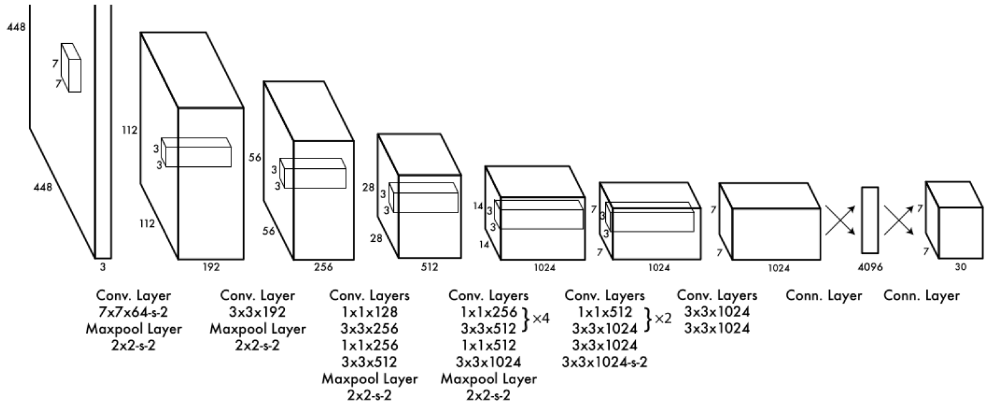
Algoritmos *One-Stage* têm como principal representante modelos que seguem a arquitetura *You Only Look Once* (YOLO). Introduzida por Redmon *et al.* (2016), representou um marco na detecção de objetos ao propor um modelo de inferência unificada, capaz de realizar localização e classificação em uma única etapa. Diferentemente de modelos baseados em múltiplas fases, como o *Region-based Convolutional Neural Network* (R-CNN), o YOLO trata a tarefa como um problema de regressão direta sobre a imagem inteira. A Figura 10 ilustra a arquitetura da primeira versão do YOLO, evidenciando o fluxo da imagem de entrada até a predição final.

A imagem de entrada é redimensionada para 448×448 *pixels* e subdividida em uma grade espacial $S \times S$ (geralmente, 7×7). Cada célula dessa grade é responsável por prever B caixas delimitadoras, juntamente com os respectivos escores de confiança C. O escore, definido

pela Equação (2.24), corresponde ao produto entre a probabilidade de existência de um objeto na caixa e a métrica de IoU.

$$\text{Confiança} = \Pr(\text{objeto}) \times \text{IoU}_{\text{predição,real}} \tag{2.24}$$

Figura 10 – Arquitetura YOLO.



Fonte: Redmon *et al.* (2016).

Adicionalmente, cada célula da grade estima as probabilidades condicionais das classes presentes, representadas por $P(C_i \mid o)$, conforme a Equação (2.25). A expressão define a probabilidade de a classe C_i estar presente, dado que há um objeto (o) na célula em questão. Essas estimativas são fundamentais para a posterior combinação com o escore de confiança de cada predição, resultando na probabilidade final de uma determinada classe estar associada a uma caixa delimitadora.

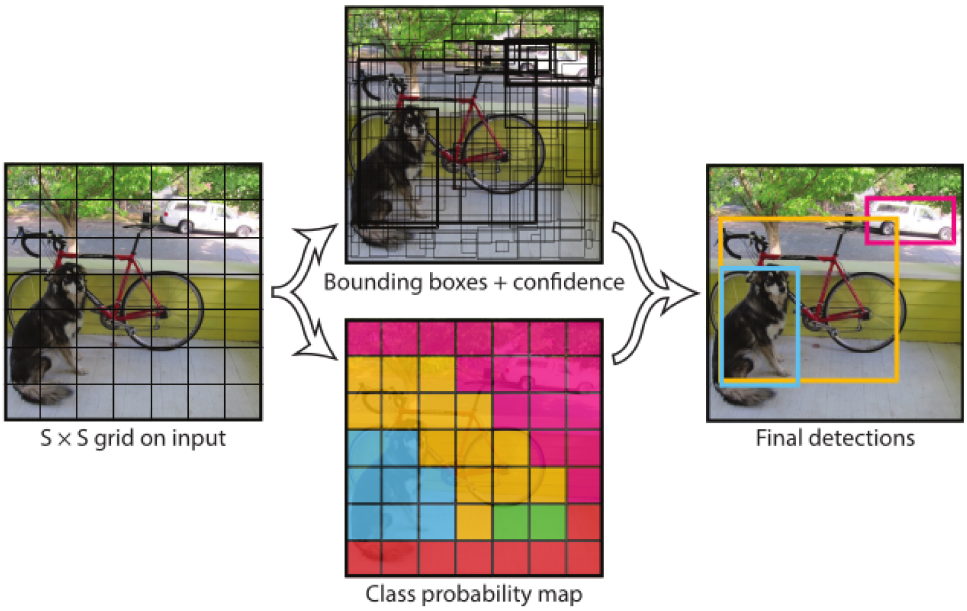
$$P(C_i \mid o) \tag{2.25}$$

Com base nessas informações, a saída da rede é organizada em um tensor tridimensional de dimensão $S \times S \times (B \times 5 + C)$, em que S representa o número de células por eixo da grade, B é o número de caixas delimitadoras previstas por célula, e C corresponde ao número total de classes. O valor 5 refere-se aos parâmetros de cada caixa: coordenadas (x, y), dimensões (w, h) e escore de confiança. A Figura 11 ilustra esse processo, desde a divisão da imagem em células da grade até as predições de caixas delimitadoras.

A primeira versão da arquitetura, o YOLOv1, iniciou uma evolução significativa na abordagem de detecção de objetos em tempo real. Em seu sucessor, o YOLOv2, foram introduzidas melhorias relevantes, como o uso de *anchor boxes*, normalização por lote (*batch*

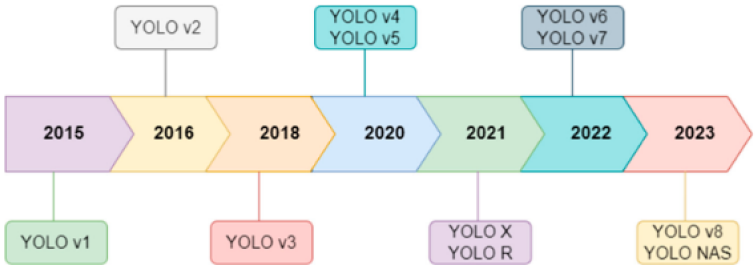
normalization) e treinamento em múltiplas escalas, resultando em ganhos substanciais de acurácia. Na versão YOLOv3, foram incorporadas predições em diferentes resoluções e conexões residuais por meio do *backbone* Darknet-53, otimizando, em especial, a detecção de objetos de pequeno porte (SAPKOTA *et al.*, 2025). A Figura 12 ilustra a evolução das versões do YOLO ao longo do tempo.

Figura 11 – Representação do funcionamento do YOLO.



Fonte: Redmon *et al.* (2016).

Figura 12 – Evolução das arquiteturas YOLO.

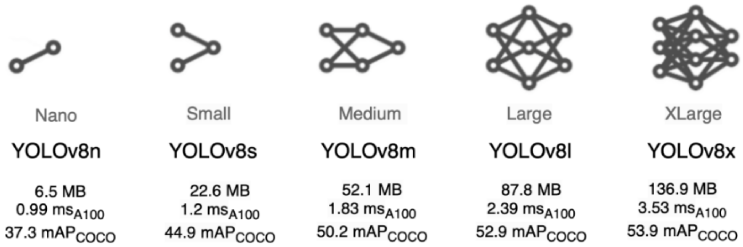


Fonte: Ali e Zhang (2024).

A arquitetura YOLOv4, desenvolvida por pesquisadores independentes, adotou uma abordagem híbrida, combinando componentes como CSPDarknet53, PANet e estratégias avançadas de aumento de dados, como *mosaic* e DropBlock. Essa combinação elevou a acurácia do modelo ao mesmo tempo em que manteve alta eficiência computacional, consolidando-o como referência em aplicações em tempo real. Em seguida, o YOLOv5, desenvolvido pela empresa Ultralytics, representou uma mudança de paradigma ao migrar para o *framework* PyTorch, o que

facilitou a disseminação e personalização do modelo pela comunidade. A versão cinco também introduziu variantes escaláveis como *nano*, *small*, *medium*, *large* e *extra-large* que superaram seus antecessores em termos de precisão e velocidade, especialmente em conjuntos de dados como o PASCAL VOC. A Figura 13 ilustra essas variantes para a versão oito do YOLO

Figura 13 – Comparação entre as variantes do YOLOv8.



Fonte: Sharma (2023).

Nas versões mais recentes, YOLOv6, YOLOv7 e YOLOv8, o foco tem sido o aprimoramento da robustez e da precisão, por meio da introdução de redes reparametrizadas, mecanismos de atenção e arquiteturas *anchor-free*. O YOLOv8, em particular, destaca-se por sua estrutura modular e por sua capacidade de realizar múltiplas tarefas, como detecção, segmentação, rastreamento e classificação (SAPKOTA *et al.*, 2025).

3 METODOLOGIA

Para o desenvolvimento deste trabalho, serão empregadas tecnologias e ferramentas modernas para o treinamento de modelos de ML. Além disso, será implementado um APP com o objetivo de proporcionar uma visualização rápida e eficiente da imagem analisada. Nesta seção, serão discutidos esses aspectos em detalhes.

3.1 Materiais

Para a execução deste projeto, foram empregadas ferramentas e materiais essenciais para o desenvolvimento e análise dos modelos de ML. O Google Colab foi utilizado como ambiente de desenvolvimento, oferecendo uma plataforma baseada em nuvem com suporte a GPUs L4, facilitando o treinamento intensivo de redes neurais. Sua integração com o Google Drive possibilitou o armazenamento e gerenciamento eficiente dos conjuntos de dados e modelos, simplificando a importação e exportação de arquivos.

Além disso, adotou-se a linguagem de programação Python, reconhecida por sua simplicidade sintática e ampla aceitação na comunidade de ML. O desenvolvimento dos modelos foi conduzido por meio do framework *TensorFlow* (TF), em conjunto com sua API de alto nível, o Keras, que proporcionou um ambiente abrangente para criação, treinamento e validação dos algoritmos.

Para validação em dispositivo móvel, utilizou-se um smartphone Samsung Galaxy A52s, equipado com processador octa-core Snapdragon 778G, 6 GB de memória RAM e GPU integrada Adreno 642L. A execução dos modelos diretamente no dispositivo permitiu avaliar com precisão os tempos de inferência tanto para detecção quanto para classificação, garantindo que a solução proposta fosse tecnicamente viável, funcional e responsiva em ambientes externos ao laboratório.

3.2 Base de Dados

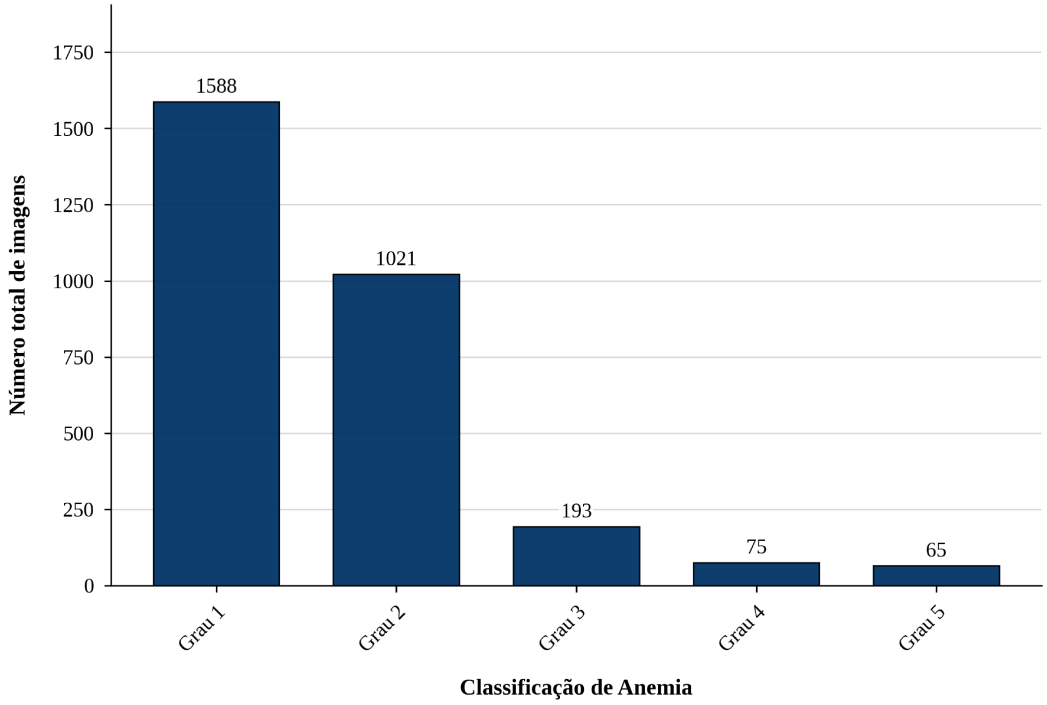
As imagens utilizadas neste trabalho foram cedidas com a devida autorização por ALMEIDA (2021), tendo sido coletadas na EMBRAPA, com o apoio dos colaboradores da instituição. As imagens foram organizadas e classificadas com base no número do brinco do animal, na raça e nos valores de Ht.

As amostras foram inicialmente organizadas conforme a data de captura e o disposi-

tivo utilizado, o que resultou em múltiplas ocorrências do mesmo animal em diferentes aparelhos de coleta. Para eliminar essa redundância, adotou-se o critério de selecionar apenas as imagens provenientes de um único aparelho por coleta.

Posteriormente, as amostras foram categorizadas e agrupadas de acordo com seus respectivos níveis de Ht, seguindo como referência os valores contidos na Tabela 1. Resultando na distribuição observada na Figura 14.

Figura 14 – Distribuição de imagens por grau de anemia.



Fonte: Próprio autor.

Na Figura 14, observa-se uma clara discrepância entre os valores para animais saudáveis (graus 1 e 2) e os doentes. Isso se deve ao fato de que a criação da EMBRAPA possui uma excelente estrutura, incluindo assistência veterinária, suplementação vitamínica e alimentação adequada, o que dificulta a progressão de doenças entre os animais.

Para a montagem da base de dados final, foram selecionadas amostras que apresentavam boas condições de nitidez e ausência de ambiguidade na classificação. Essa curadoria foi necessária devido à frequente ocorrência de casos em que, embora o nível de Ht fosse compatível com a classe atribuída, a coloração da mucosa ocular não refletia claramente essa condição, optando-se pela exclusão dessas amostras.

Esse critério resultou em apenas 33 imagens para o grupo mais reduzido (grau 5). Diante disso, adotou-se o equilíbrio amostral, selecionando a mesma quantidade de imagens

para todos os grupos, totalizando 165 amostras igualmente distribuídas entre as classes para o presente trabalho.

3.3 Aumento de Dados

A técnica de Aumento de Dados (do inglês, *Data Augmentation* (DA)) foi aplicada para ampliar o conjunto de dados e melhorar a performance dos modelos de ML. Segundo Rici *et al.* (2021), essa técnica ajuda a reduzir o *overfitting*, permitindo que os modelos aprendam com uma variedade maior de exemplos. No projeto, foram realizadas as seguintes transformações nas imagens:

- Rotações (*rotation range*): alteração da orientação das imagens para simular variações angulares.
- Espelhamento horizontal (*horizontal flip*): aumento da diversidade por meio de inversão horizontal.
- Espelhamento vertical (*vertical flip*): aumento da diversidade por meio de inversão vertical.
- Ajuste de brilho (*brightness range*): modificação dos níveis de brilho para simular diferentes condições de iluminação.
- Zoom (*zoom range*): alteração da escala das imagens para simular aproximação ou afastamento.
- Cisalhamento (*shear range*): distorção angular para simular mudanças de perspectiva.
- Deslocamento horizontal (*width shift range*): deslocamento lateral para simular variações na posição da imagem.
- Deslocamento vertical (*height shift range*): deslocamento vertical para simular variações na posição da imagem.
- Normalização dos pixels (*rescale*): transformação dos valores dos pixels para o intervalo [0, 1], facilitando o aprendizado.

3.4 Pipeline de Detecção e Classificação

Para viabilizar a utilização dos modelos de DL no APP, foi desenvolvido um *pipeline* de detecção e classificação. O processo inicia-se com a aquisição e decodificação da imagem para um objeto `Bitmap`, seguida do redimensionamento às dimensões exigidas pelo modelo de detecção. Essa etapa garante que os dados estejam no formato e escala adequados à inferência

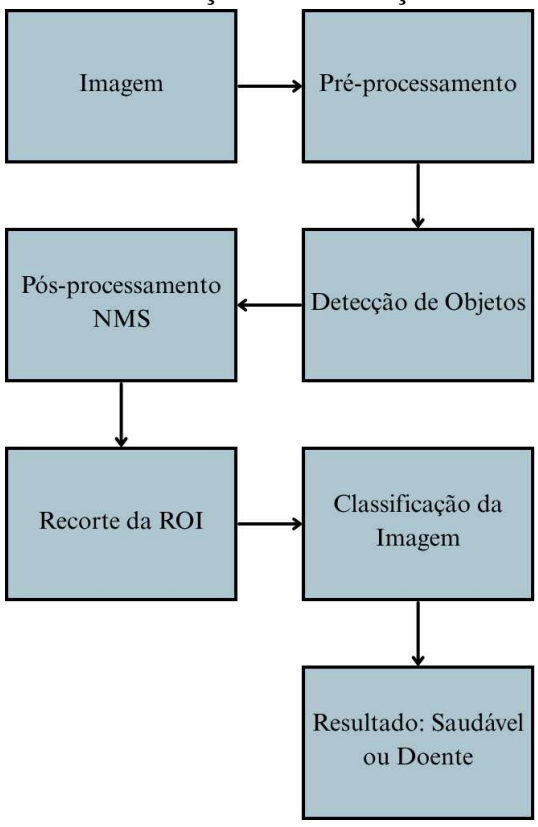
no *TensorFlow Lite* (TFLite), formato adotado para a exportação dos modelos.

Posteriormente, o modelo de detecção de objetos identifica possíveis mucosas na imagem. Cada detecção gera uma BB acompanhada de uma pontuação de confiança. Para eliminar sobreposições e redundâncias, aplica-se a técnica de *Non-Maximum Suppression* (NMS), que seleciona apenas as caixas com maior confiança e IoU aceitável. Caso nenhuma detecção atenda aos critérios definidos, o sistema notifica o usuário para realizar uma nova captura.

Com a BB mais confiável identificada, extrai-se a *Region of Interest* (ROI) da imagem original. Essa região é então submetida a um pré-processamento específico para a etapa de classificação, o qual inclui redimensionamento e normalização. O classificador gera como saída um vetor de probabilidades, a partir do qual se seleciona a classe com o maior valor predito.

São utilizadas cinco classes distintas, posteriormente agrupadas em duas categorias finais: “saudável” ou “doente” de acordo com a Tabela 1. O resultado da classificação é então apresentado ao usuário, juntamente com a imagem analisada. Caso desejado, o diagnóstico pode ser salvo para consultas futuras ou para fins de acompanhamento clínico. O fluxo completo está representado na Figura 15.

Figura 15 – Diagrama do *pipeline* de detecção e classificação.



Fonte: Próprio autor.

3.5 Modelo de Detecção

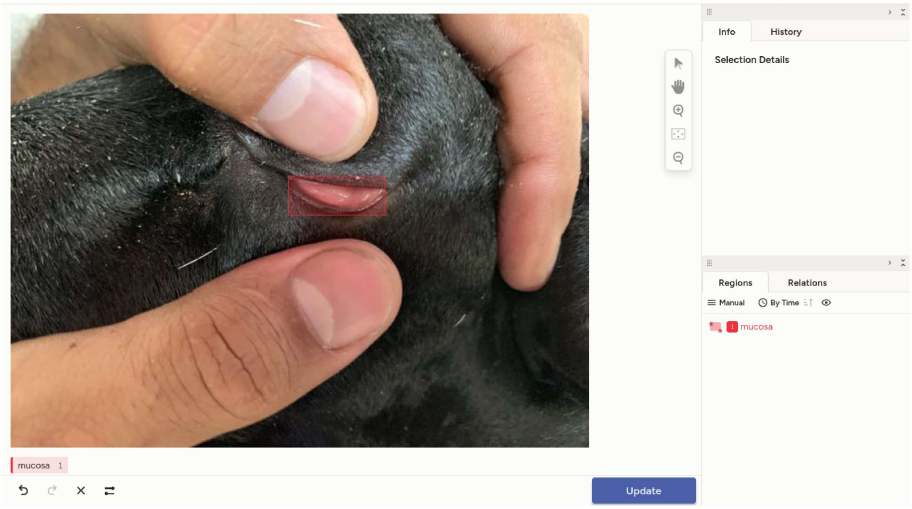
Conforme apresentado na Seção 2.10, o YOLOv5 oferece variantes escaláveis que permitem ajustar o modelo à complexidade e restrições da aplicação. Considerando que a aplicação desenvolvida neste trabalho foi destinada a dispositivos móveis que possuem limitações de processamento e memória, foi necessário optar por uma versão leve do modelo.

A escolha pelo YOLOv5 justifica-se pela ampla documentação disponível, pela maturidade do modelo e pela facilidade de integração, fatores que contribuíram para um desenvolvimento ágil e confiável. Optou-se pela variante YOLOv5n (nano), por sua adequação a ambientes com recursos computacionais reduzidos.

As 165 imagens foram anotadas utilizando a ferramenta LabelImg, conforme evidenciado na Figura 16, configurada para gerar arquivos no formato YOLO (`.txt`), contendo cinco valores normalizados por linha: `class_id`, `x_center`, `y_center`, `width` e `height`. O conjunto de dados foi dividido em 80% para treino, 10% para validação e 10% para teste. Após a anotação, os arquivos foram organizados conforme a estrutura exigida pelo framework YOLOv5, distribuídos nos diretórios *train*, *valid* e *test*.

- `train/images/` – imagens utilizadas para treinamento;
- `train/labels/` – rótulos correspondentes às imagens de treinamento;
- `valid/images/` – imagens utilizadas para validação;
- `valid/labels/` – rótulos correspondentes às imagens de validação;
- `test/images/` – imagens utilizadas para teste;
- `test/labels/` – rótulos correspondentes às imagens de teste.

Figura 16 – Anotação das mucosas com o *software* LabelImg.



Fonte: Próprio autor.

A organização reflete a estrutura típica de datasets otimizados para esse tipo de modelo, garantindo uma integração direta e eficiente com o pipeline de treinamento. A configuração do modelo foi definida em um arquivo `data.yaml`, no qual foram especificados os caminhos dos dados, o número total de classes e seus respectivos nomes.

Para o treinamento do modelo, foi definida uma entrada de 320×320 *pixels*, com duração de 100 épocas e *batch size* igual a 32. Os hiperparâmetros obrigatórios foram configurados manualmente no arquivo `custom_hyp.yaml`, conforme apresentados na Tabela 6. Entre eles, destacam-se: a taxa de aprendizado inicial (`lr0`), o fator de decaimento da taxa de aprendizado (`lrf`), o *momentum*, a regularização via `weight_decay`, o número de épocas de aquecimento (`warmup_epochs`) e os pesos atribuídos às funções de perda associadas à detecção de caixas (`box`), à classificação (`cls`) e à presença de objetos (`obj`). Esses parâmetros são fundamentais para o ajuste fino do processo de aprendizagem.

Tabela 6 – Hiperparâmetros utilizados no treinamento do modelo YOLOv5

Hiperparâmetro	Valor
`lr0`	0,01
`lrf`	0,1
`momentum`	0,937
`weight_decay`	0,0005
`warmup_epochs`	3,0
`warmup_momentum`	0,8
`warmup_bias_lr`	0,1
`box`	0,05
`cls`	0,5
`cls_pw`	1,0
`obj`	1,0
`obj_pw`	1,0
`iou_t`	0,20
`anchor_t`	4,0
`fl_gamma`	0,0

Fonte: Próprio autor.

Além dos hiperparâmetros de treinamento, foram definidos parâmetros específicos para estratégias de DA. Esses hiperparâmetros também foram incluídos no arquivo `custom_hyp.yaml` e envolvem modificações nas propriedades de cor, geometria e composição das imagens. Na Tabela 7 são definidos os parâmetros empregados. Destacam-se alterações no matiz (`hsv_h`), saturação (`hsv_s`), valor (`hsv_v`), rotações (`degrees`), translações (`translate`), escalonamento (`scale`), cisalhamento (`shear`), adição de perspectiva (`perspective`), espelhamento vertical (`flipud`) e horizontal (`fliplr`), além de composições por mosaico (`mosaic`), mistura

de imagens (`mixup`) e colagem de objetos (`copy_paste`).

Tabela 7 – Hiperparâmetros de DA utilizados no treinamento do modelo YOLOv5

Hiperparâmetro	Valor
`hsv_h`	0,015
`hsv_s`	0,7
`hsv_v`	0,4
`degrees`	10,0
`translate`	0,1
`scale`	0,5
`shear`	2,0
`perspective`	0,0
`flipud`	0,0
`fliplr`	0,5
`mosaic`	1,0
`mixup`	0,0
`copy_paste`	0,0

Fonte: Próprio autor.

3.6 Modelos de Classificação

A escolha dos classificadores considerou sua aplicabilidade em dispositivos móveis, especialmente *smartphones*. Além disso, esses modelos possuem implementação nativa no TF, o que facilitou o desenvolvimento e viabilizou o lançamento de versões futuras de forma mais ágil. Com essas premissas, os modelos escolhidos foram MobileNetV2, EfficientNetB0 e NASNetMobile.

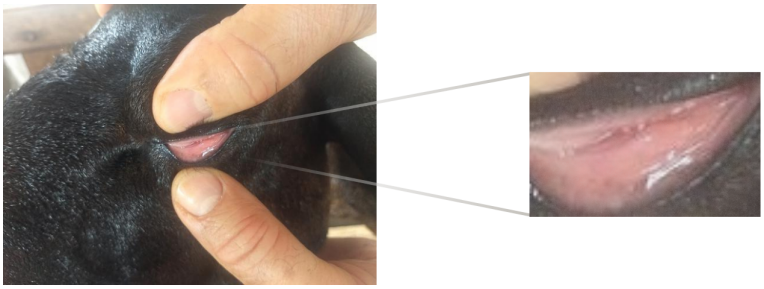
Além da utilização de modelos consolidados e validados na literatura, foi implementada uma arquitetura CNN desenvolvida para este trabalho, com o objetivo de permitir comparações experimentais entre modelos base e a arquitetura proposta.

O processo de recorte da mucosa foi realizado automaticamente pelo modelo de detecção. A partir do *dataset* inicial, foi gerado um novo conjunto contendo apenas as regiões recortadas conforme ilustrado na Figura 17.

Neste trabalho não foi empregada transferência de aprendizado. Todos os classificadores foram treinados a partir de pesos aleatórios. Durante o treinamento, adotaram-se hiperparâmetros padronizados para garantir consistência metodológica nas comparações. A base de dados foi composta por cinco classes com distribuição equilibrada entre elas. As imagens foram redimensionadas para 64×64 *pixels* e normalizadas para o intervalo $[0, 1]$. O processo de treinamento foi conduzido por 700 épocas, com *batch size* de 16, e a avaliação dos resultados

foi realizada por meio de validação cruzada do tipo *k-fold*, com $k = 5$. Em cada iteração dessa validação, as previsões obtidas e os respectivos valores reais foram registrados para possibilitar análises posteriores mais detalhadas do desempenho dos modelos.

Figura 17 – Exemplo de mucosa cortada.



Fonte: Próprio autor.

Nos experimentos iniciais, constatou-se que as classes de grau 2 e grau 3, correspondentes ao limiar entre casos saudáveis e anêmicos, apresentaram desempenho inferior em relação às demais categorias. Com o intuito de atenuar a discrepância, aplicou-se um fator de ponderação igual a 1,5 para essas classes, mantendo-se o peso unitário (1,0) para as demais. Essa medida teve como objetivo elevar a acurácia dessas classes específicas, aproximando seus resultados daqueles observados nas demais categorias.

Para aumentar a capacidade de generalização dos classificadores, aplicaram-se técnicas de DA. Os valores correspondentes podem ser consultados na Tabela 8.

Tabela 8 – Hiperparâmetros de DA utilizados nos classificadores

Hiperparâmetro	Valor
`rescale`	1,0 / 255
`brightness_range`	(0,7 ; 1,3)
`zoom_range`	0,2
`rotation_range`	15°
`shear_range`	0,15
`width_shift_range`	0,1
`height_shift_range`	0,1
`horizontal_flip`	Ativado
`vertical_flip`	Ativado

Fonte: Próprio autor.

3.6.1 MobileNetV2

O classificador foi baseado na arquitetura MobileNetV2 do TF. A agregação espacial foi realizada por meio de `GlobalAveragePooling2D`. Em seguida, para mitigar o *overfitting*,

foram inseridas duas camadas de *dropout* (0,5), intercaladas por uma camada densa de 128 neurônios com ativação ReLU e regularização L2 ($\lambda = 0{,}01$).

A saída final é composta por uma camada densa com cinco unidades e ativação Softmax, gerando distribuições de probabilidade para as cinco classes da base de dados. A compilação foi efetuada com o otimizador Adam com $\eta = 0{,}001$, utilizando *categorical crossentropy* como função de perda e a acurácia como métrica de avaliação. As camadas com hiperparâmetros estão resumidas na Tabela 9.

Tabela 9 – Configuração do MobileNetV2.

Camada	Configuração
MobileNetV2 Base	Sem pesos pré-treinados
`GlobalAveragePooling2D`	—
`Dropout`	0,5
`Dense`	128 neurônios, ReLU, L2 ($\lambda = 0{,}01$)
`Dropout`	0,5
`Dense (saída)`	5 neurônios, softmax

Fonte: Próprio autor.

3.6.2 EfficientNet-B0

Desenvolvido utilizando a arquitetura EfficientNet-B0 seguindo o mesmo padrão de construção do MobileNetV2, com a diferença de que a camada densa intermediária utiliza 256 neurônios e a regularização L2 de $\lambda = 0{,}001$.

Tabela 10 – Configuração do EfficientNet-B0.

Camada	Configuração
EfficientNet-B0 Base	Sem pesos pré-treinados
`GlobalAveragePooling2D`	—
`Dropout`	0,5
`Dense`	256 neurônios, ReLU, L2 ($\lambda = 0{,}001$)
`Dropout`	0,5
`Dense (saída)`	5 neurônios, softmax

Fonte: Próprio autor.

3.6.3 NASNetMobile

Construído a partir da arquitetura NASNetMobile, segue o mesmo padrão das demais, porém com regularização L2 de $\lambda = 0{,}001$ e 128 neurônios na camada densa intermediária.

Tabela 11 – Configuração do NASNetMobile

Camada	Configuração
NASNetMobile Base	Sem pesos pré-treinados
`GlobalAveragePooling2D`	—
`Dropout`	0,5
`Dense`	128 neurônios, ReLU, L2 ($\lambda = 0{,}001$)
`Dropout`	0,5
`Dense (saída)`	5 neurônios, softmax

Fonte: Próprio autor.

3.6.4 Arquitetura Proposta

A arquitetura da CNN proposta foi composta por três camadas convolucionais sequenciais (`Conv2D`) com 32, 64 e 128 filtros, respectivamente. Cada camada convolucional foi seguida por uma camada de `BatchNormalization` e uma camada de `MaxPooling2D`.

Para mitigar o *overfitting*, aplicou-se regularização L2 com $\lambda = 0{,}01$ em todas as camadas convolucionais. Na etapa de classificação, os mapas de características são transformados em vetor por meio de uma camada Flatten, seguida por uma camada densa com 128 neurônios, função de ativação ReLU e regularização L2 ($\lambda = 0{,}01$). Após essa etapa, aplica-se um *dropout* de 0,5.

Por fim, a camada de saída é uma densa com cinco unidades e ativação Softmax, responsável por gerar as distribuições de probabilidade finais para as cinco classes do problema. A compilação do modelo foi realizada com o otimizador Adam, configurado com LR de $\eta = 0{,}0001$, função de perda *categorical crossentropy* e métrica de avaliação a acurácia. A Tabela 12 detalha a arquitetura completa.

Tabela 12 – Configuração da Arquitetura Proposta.

Camada	Configuração
`Conv2D`	32 filtros, (3x3), ReLU, L2 ($\lambda = 0{,}01$)
`BatchNormalization`	—
`MaxPooling2D`	(2x2)
`Conv2D`	64 filtros, (3x3), ReLU, L2 ($\lambda = 0{,}01$)
`BatchNormalization`	—
`MaxPooling2D`	(2x2)
`Conv2D`	128 filtros, (3x3), ReLU, L2 ($\lambda = 0{,}01$)
`BatchNormalization`	—
`MaxPooling2D`	(2x2)
`Flatten`	—
`Dense`	128 neurônios, ReLU, L2 ($\lambda = 0{,}01$)
`Dropout`	0,5
`Dense (saída)`	5 neurônios, softmax

Fonte: Próprio autor.

3.7 Aplicativo

O APP foi desenvolvido utilizando a linguagem de programação Kotlin, em conjunto com o *framework* Jetpack Compose, que oferece uma abordagem moderna e declarativa para a construção de interfaces de usuário no sistema operacional Android. Esta combinação de ferramentas permitiu a criação de uma aplicação responsiva, com um fluxo de navegação intuitivo e de fácil usabilidade para o público-alvo.

Os requisitos do APP foram definidos de forma a atender aos objetivos propostos neste trabalho, proporcionando uma ferramenta prática e acessível para a análise de anemia em ovinos. O *software* de funcionamento simples opera basicamente com duas tarefas principais, a escolha da imagem e posteriormente a analise dessa amostra pelos modelos de DL.

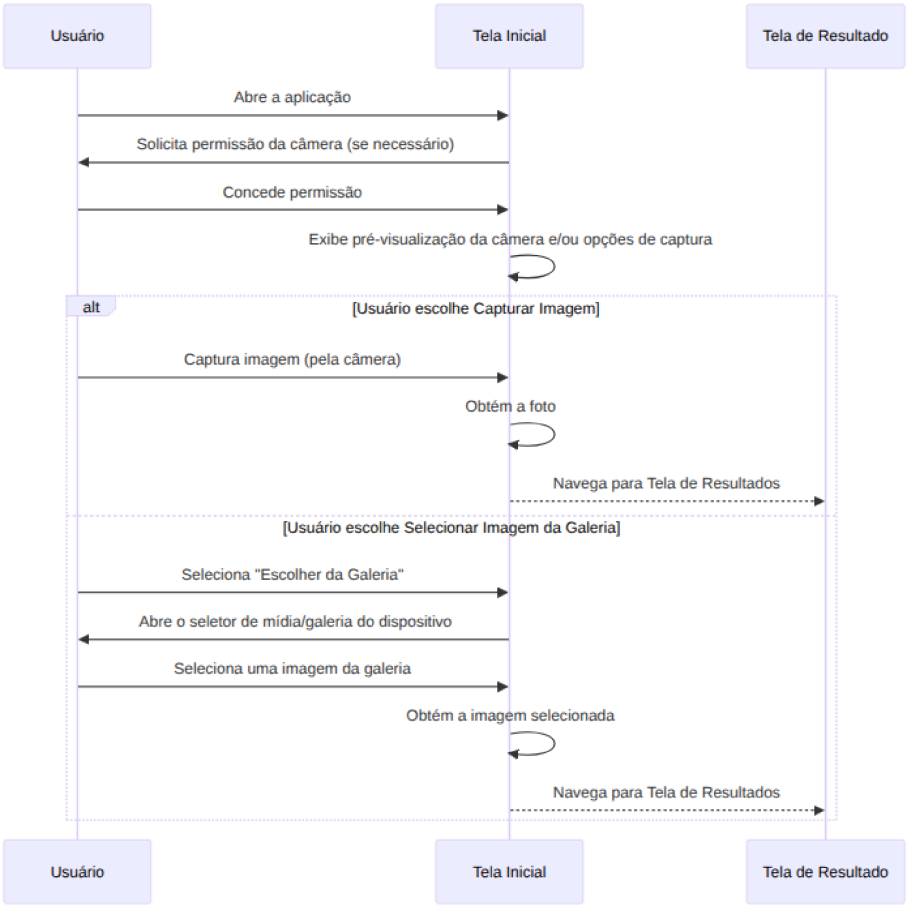
3.7.1 Tela Inicial

Na tela inicial, o APP possibilita ao usuário selecionar uma imagem da galeria ou capturar uma nova utilizando a câmera do dispositivo. A interface apresenta ícones de tamanho ampliado para facilitar a interação. O fluxo de navegação dessa etapa está representado no diagrama de sequência da Figura 18.

3.7.2 Tela de Resultado

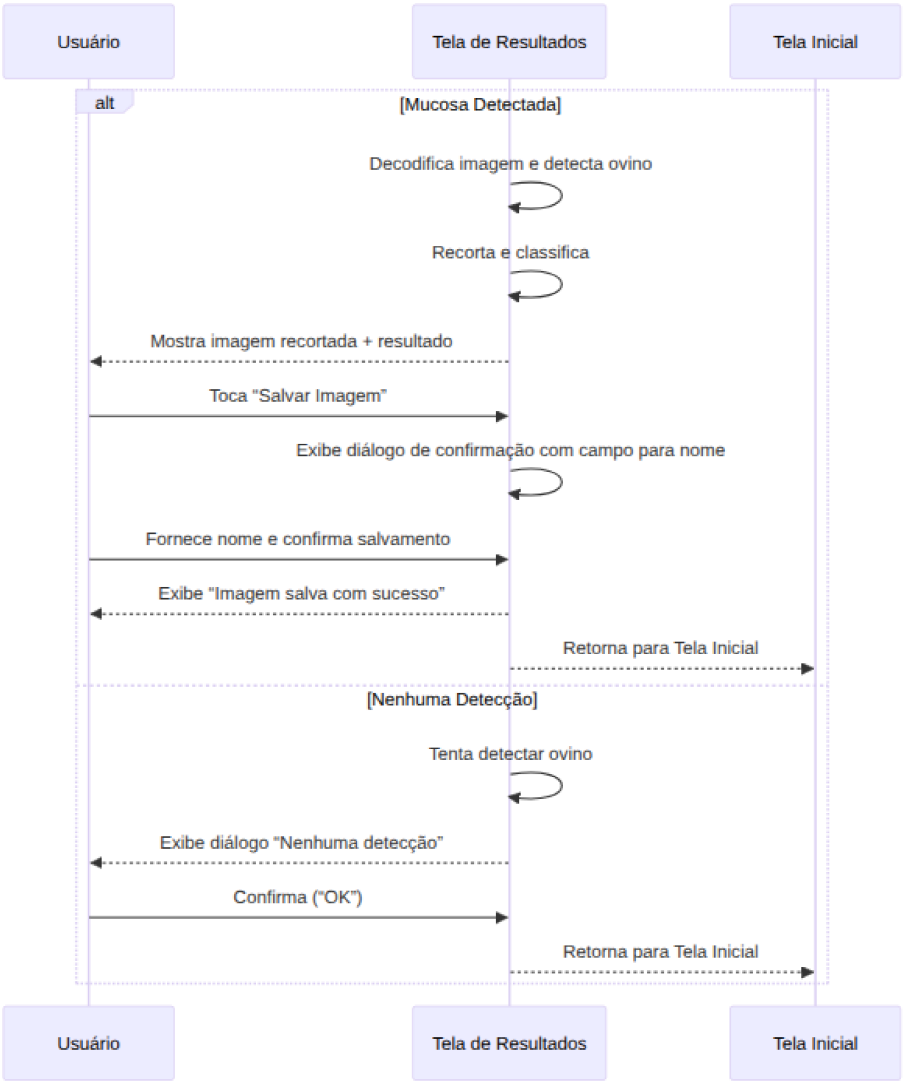
Após a seleção ou captura da imagem na tela inicial, o usuário é direcionado para a tela de resultados. Nesta etapa, o sistema exibe a imagem recortada da ROI (mucosa do animal) com a classificação gerada pelo modelo CNN . A interface oferece a opção de salvar a imagem processada, por meio de um diálogo de confirmação que permite ao usuário atribuir um nome ao arquivo antes do armazenamento. Caso nenhuma mucosa seja detectada na imagem, o sistema apresenta uma mensagem informando a impossibilidade de realizar a análise e, após confirmação, retorna automaticamente à tela inicial. O fluxo desta etapa está ilustrado no diagrama de sequência da Figura 19.

Figura 18 – Diagrama de sequência da tela inicial.



Fonte: Próprio autor.

Figura 19 – Diagrama de sequência da tela de resultados.



Fonte: Próprio autor.

4 RESULTADOS

Este capítulo apresenta os resultados da metodologia do Capítulo 3, detalhando as métricas dos modelos de detecção e classificação, além da análise da usabilidade do APP desenvolvido.

4.1 YOLO

O modelo YOLOv5 foi avaliado utilizando um conjunto de teste composto por 17 amostras. As métricas de desempenho obtidas encontram-se na Tabela 13.

Tabela 13 – Resultados das métricas para detecção da mucosa.

Métrica	**Valor**
Precisão	0.996
Recall	1.000
mAP@0.5	0.995
mAP@0.5:0.95	0.598

Fonte: Próprio autor.

O modelo YOLOv5 apresentou precisão de 0,996, *recall* de 1,000 e mAP@0,5 de 0,995, indicando alta eficácia na detecção e localização das BB. O mAP@0,5:0,95 foi 0,598, refletindo seu desempenho em múltiplos limiares. A Figura 20 ilustra amostras do conjunto de teste com as detecções e seus valores de confiança.

Figura 20 – Exemplos de detecção da mucosa utilizando YOLO.

Fonte: Próprio autor.

4.2 Modelos de classificação

Os modelos de classificação apresentados na Seção 3.6 foram avaliados por meio de validação cruzada *k-fold*. Em cada iteração, as previsões e os valores reais do conjunto de teste foram registrados para análises posteriores. Na Tabela 14 apresenta o desempenho médio e o desvio padrão de cada modelo, considerando acurácia, precisão, *recall* e F1-Score.

Tabela 14 – Desempenho comparativo entre os modelos avaliados.

Modelo	Acurácia	Precisão	Recall	F1-Score
MobileNetV2	$0{,}6061 \pm 0{,}1767$	$0{,}6118 \pm 0{,}2114$	$0{,}6076 \pm 0{,}1702$	$0{,}5589 \pm 0{,}2125$
EfficientNet-B0	$0{,}8303 \pm 0{,}1159$	$0{,}8797 \pm 0{,}0771$	$0{,}8324 \pm 0{,}1169$	$0{,}8326 \pm 0{,}1101$
NASNetMobile	$0{,}3939 \pm 0{,}0691$	$0{,}3182 \pm 0{,}0619$	$0{,}3905 \pm 0{,}0663$	$0{,}2995 \pm 0{,}0459$
Arquitetura Proposta	$0{,}9030 \pm 0{,}0521$	$0{,}9168 \pm 0{,}0432$	$0{,}9057 \pm 0{,}0500$	$0{,}9026 \pm 0{,}0515$

Fonte: Próprio autor.

O MobileNetV2 apresentou limitações de generalização, evidenciadas pela divergência progressiva entre as curvas de perda de treino e validação, caracterizando *overfitting*. A análise por classe indicou padrões recorrentes de confusão entre as categorias grau 2 e grau 3, com erros consistentes observados nas matrizes de confusão, conforme descrito no Apêndice A.

O EfficientNet-B0 evidenciou volatilidade nas curvas de aprendizado, apesar do desempenho superior em relação ao MobileNetV2. Foram identificadas fragilidades nos graus iniciais de anemia, com *recall* reduzido no grau 1 e precisão baixa no grau 2, como apresentado no Apêndice B.

O NASNetMobile apresentou redução na capacidade discriminatória, com tendência a predições concentradas nas classes grau 3 e grau 5. Observou-se degradação gradual da perda de validação, associada a *overfitting*, e menor separabilidade entre classes, evidenciada nas matrizes de confusão por *fold*, descritas no Apêndice C.

A arquitetura proposta apresentou desempenho consistente entre classes, com F1-Score superior a 0,84 em todas as categorias. A convergência estável das curvas de treino e validação, juntamente com a precisão elevada nos graus 4 e 5, indicam boa adaptação ao domínio do problema, com detalhes adicionais no Apêndice D.

Os resultados indicam que a arquitetura proposta apresentou desempenho superior aos demais modelos, evidenciando maior capacidade de discriminação entre os diferentes graus de anemia e consistência nas métricas avaliadas. Essa vantagem pode ser explicada pela topologia arquitetural específica, desenvolvida para captar características essenciais ao problema, sem

a complexidade adicional encontrada em modelos de aplicação mais ampla. Em contraste, MobileNetV2, NASNetMobile e EfficientNet-B0 demonstraram limitações na generalização e na capacidade de classificação, possivelmente decorrentes de sua natureza mais abrangente, que pode representar um excesso de complexidade para a tarefa em questão.

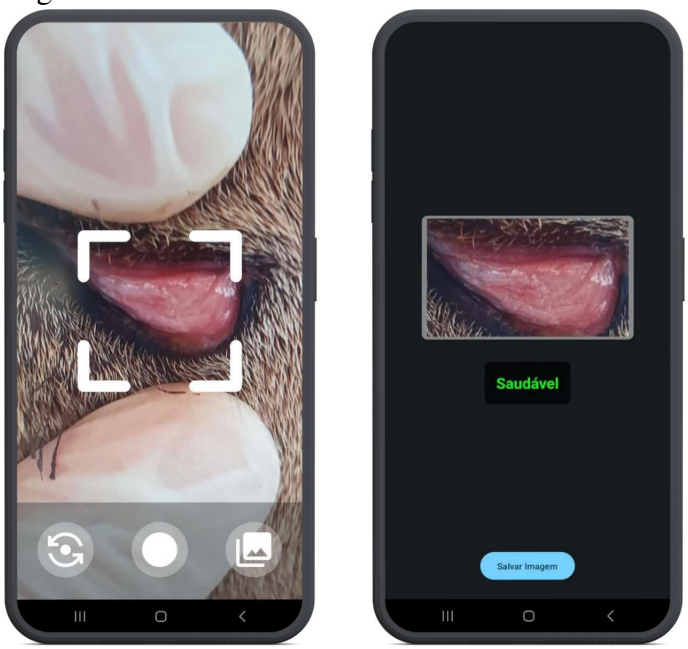
Assim, o modelo desenvolvido mostra-se mais apropriado para implementação no sistema móvel, combinando precisão elevada com flexibilidade para futuras adaptações. Sua estrutura permite a incorporação de novos dados e retreinamento simplificado, facilitando atualizações e aprimoramentos contínuos do sistema ao longo do tempo.

4.3 Aplicativo

A Figura 21 mostra dois *screenshots* da aplicação: a tela inicial para captura ou seleção de imagens, com interface simples e acessível, e a tela de resultado que destaca a área da mucosa detectada e a classificação entre "saudável" ou "doente".

Nos testes, o modelo YOLOv5 foi utilizado para a detecção, apresentando tempo médio de inferência de 83 milissegundos, enquanto a CNN proposta realizou a classificação em 20 milissegundos. O *pipeline* completo consumiu cerca de 103 milissegundos no Samsung Galaxy A52s, garantindo desempenho fluido em tempo quase real, sem necessidade de *hardware* avançado.

Figura 21 – Telas de início e resultados do APP.



Fonte: Próprio autor.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho desenvolveu com sucesso um APP capaz de automatizar a análise do cartão FAMACHA© por meio de técnicas de DL. O sistema demonstrou alta eficácia na detecção e classificação da mucosa ocular de ovinos, cumprindo os objetivos propostos. O modelo YOLOv5 apresentou excelente desempenho na etapa de detecção, com precisão de 99,6%, garantindo a identificação precisa das mucosas. Para a classificação, a arquitetura proposta superou modelos consolidados, como a MobileNetV2 e a EfficientNet-B0, alcançando uma acurácia média de 90,3% e um F1-Score de 90,3%.

A integração dos modelos no APP resultou em uma ferramenta funcional e acessível, com tempos de inferência de 103 milissegundos no total, em um *smartphone* intermediário. A interface, desenvolvida em Kotlin com Jetpack Compose, foi projetada para ser intuitiva, permitindo a captura ou seleção de imagens e exibindo os resultados de forma clara e imediata. A combinação de desempenho e usabilidade torna a solução viável para uso em condições reais de campo, auxiliando produtores, técnicos e médicos-veterinários na identificação da anemia em ovinos.

Para trabalhos futuros, recomenda-se ampliar o banco de imagens, com ênfase nas classes grau 4 e 5 de anemia, atualmente sub-representadas no conjunto de dados. A inclusão de um maior número de amostras desses estágios avançados permitiria o uso de mais imagens no treinamento, considerando a escassez de exemplos para essas classes. Como complemento ao critério baseado no Ht, propõe-se a utilização conjunta do sistema FAMACHA© para seleção das imagens. Essa abordagem híbrida, combinando um método quantitativo com um critério clínico validado, possibilitaria uma triagem mais criteriosa dos casos, minimizando a exclusão de imagens potencialmente úteis devido a variações naturais na coloração da mucosa ocular. A convergência desses dois parâmetros aumentaria a confiabilidade do *dataset* final.

REFERÊNCIAS

AL-NAYMAT, G.; AL-KASASSBEH, M.; ABU-SAMHADANH, N.; SAKR, S. Classification of voip and non-voip traffic using machine learning approaches. **Journal of Theoretical & Applied Information Technology**, 2016.

ALI, M. L.; ZHANG, Z. The yolo framework: A comprehensive review of evolution, applications, and benchmarks in object detection. **Preprints**, 2024. Preprint, <https://www.preprints.org/manuscript/202410.1785/v1>.

ALMEIDA, A. M. A. **Detecção de anemia em ovinos através de aprendizagem profunda em imagens de mucosa ocular**. 111 p. Dissertação (Mestrado) — Universidade Federal do Ceará, Sobral, CE, 2021. Acesso em: 10 ago. 2025. Disponível em: <https://repositorio.ufc.br/handle/riufc/73133>.

ARAUJO, R. M. **Perceptrons: conceitos fundamentais e aplicações**. 2020. <https://ricardomatsumura.medium.com/perceptrons-f18935009a61>. Publicado no Medium. Acesso em: 16 maio 2025.

BESIER, R.; KAHN, L.; SARGISON, N.; Van Wyk, J. Chapter six - diagnosis, treatment and management of haemonchus contortus in small ruminants. In: GASSER, R. B.; SAMSON-HIMMELSTJERNA, G. V. (Ed.). **Haemonchus contortus and Haemonchosis – Past, Present and Future Trends**. Academic Press, 2016, (Advances in Parasitology, v. 93). p. 181–238. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0065308X16300240>.

BHATT, D.; PATEL, C.; TALSANIA, H.; PATEL, J.; VAGHELA, R.; PANDYA, S.; MODI, K.; GHAYVAT, H. Cnn variants for computer vision: History, architecture, application, challenges and future scope. **Electronics**, MDPI, v. 10, n. 20, p. 2470, 2021.

CANO, J.; AMREHN, L.; RUDOLPH, M.; ZHANG, X.; GöTZE, J.; LEUPERS, R. **3D input to 1D array row by row transformation**. 2018. Accessed: Sep. 2, 2024. Disponível em: <https://www.researchgate.net/figure/3D-input-to-1D-array-row-by-row-transformation_fig2_327070011>.

CHAGAS, A. C. d. S.; CARVALHO, C. d. O. d.; MOLENTO, M. B. **Método FAMACHA: um recurso para o controle da verminose em ovinos**. São Carlos, SP, 2007. Acesso em: 10 ago. 2025. Disponível em: <https://www.embrapa.br/pecuaria-sudeste/busca-de-publicacoes/-/publicacao/567536/metodo-famacha-um-recurso-para-o-controle-da-verminose-em-ovinos>.

de Souza, L. F.; COSTA, M. H.; RIET-CORREA, B. Mobile app for targeted selective treatment of haemonchosis in sheep. **Veterinary Parasitology**, v. 316, p. 109902, 2023. ISSN 0304-4017. Disponível em: <https://www.sciencedirect.com/science/article/pii/S030440172300033X>.

EMBRAPA. **Cartão Famacha**. Brasília, DF: [s.n.], 2024. <https://www.embrapa.br/documents/1355090/29069968/Paratec+vermes+foto+famacha/8e443b82-22dc-2449-1d25-34e64707f0d5?t=1509537697389>. Acesso em: 10 ago. 2025.

EVERINGHAM, M.; GOOL, L. V.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. The pascal visual object classes challenge. **International Journal of Computer Vision**, Springer, v. 88, n. 2, p. 303–338, 2010.

FEDORENKO, Y. S. Statistical testing technique for comparison machine learning models performance. **Vestnik komp'iuternykh i informatsionnykh tekhnologii**, Izdatel'skii dom Spektr, LLC, n. 186, p. 10–17, dez. 2019. ISSN 1810-7206. Disponível em: <http://dx.doi.org/10.14489/vkit.2019.12.pp.010-017>.

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

GUPTA, J.; PATHAK, S.; KUMAR, G. Deep learning (cnn) and transfer learning: a review. **Journal of Physics: Conference Series**, IOP Publishing, v. 2273, n. 1, p. 012029, 2022. Disponível em: <https://doi.org/10.1088/1742-6596/2273/1/012029>.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 2. ed. Sebastopol, CA: O'Reilly Media, 2019. ISBN 978-1-492-03264-9.

GÓRRIZ, J. M.; CLEMENTE, R. M.; SEGOVIA, F.; RAMIREZ, J.; ORTIZ, A.; SUCKLING, J. **Is K-fold cross validation the best model selection method for Machine Learning?** 2024. ArXiv preprint arXiv:2401.16407. Acesso em: 10 ago. 2025. Disponível em: <https://arxiv.org/abs/2401.16407>.

HIJAZI, S.; KUMAR, R.; ROWEN, C. *et al.* Using convolutional neural networks for image recognition. **Cadence Design Systems Inc.: San Jose, CA, USA**, v. 9, n. 1, 2015.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. In: FLEET, D.; PAJDLA, T.; SCHIELE, B.; TUYTELAARS, T. (Ed.). **European Conference on Computer Vision (ECCV)**. Cham: Springer, 2014. (Lecture Notes in Computer Science, v. 8693), p. 740–755. Disponível em: <https://doi.org/10.1007/978-3-319-10602-1_48>.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, v. 1, n. 1, p. 32, 2003.

RABELO, M. D.; FERREIRA, A. U. d. C.; SANTOS, I. B. d.; ESTEVES, S. N.; CHAGAS, A. C. d. S. **Uso de NIRS portátil para predição de anemia e diagnóstico de ovinos infectados por nematódeos gastrintestinais visando o tratamento seletivo**. São Carlos, SP, 2023. 40 p. Acesso em: 10 ago. 2025. Disponível em: <http://www.infoteca.cnptia.embrapa.br/infoteca/handle/doc/1153353>.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. **You Only Look Once: Unified, Real-Time Object Detection**. 2016. Disponível em: <https://arxiv.org/abs/1506.02640>.

RICI, P.; SANTOS, S. O. S.; OTTONI, A. L. C. Análise da seleção de hiperparâmetros de data augmentation na detecção de covid-19 em imagens de raio-x com deep learning. **Anais do [Nome Completo do Evento ou Periódico]**, v. 15, p. 1–7, 2021. Disponível em: <https://sbic.org.br/wp-content/uploads/2021/09/pdf/CBIC_2021_paper_22.pdf>. Acesso em: 28 jun. 2025.

ROCHA, M. B. d.; SARMENTO, J. L. R.; SANTOSA, N.; RABELO, R. A. L.; SILVA, R. R. V. e.; FILHO, A. O. d. C.; ARAÚJO, F. H. D. d. Método computacional para automação do FAMACHA em cabras e ovelhas. **Preprints**, 2025. Acesso em: 10 ago. 2025. Disponível em: <https://www.researchgate.net/publication/384273582_Metodo_Computacional_Para_Automacao_do_FAMACHA_em_Cabras_e_Ovelhas>.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, n. 6, p. 386–408, 1958.

SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**. IEEE, 2018. p. 4510–4520. Disponível em: <http://dx.doi.org/10.1109/CVPR.2018.00474>.

SAPKOTA, R.; FLORES-CALERO, M.; QURESHI, R.; BADGUJAR, C.; NEPAL, U.; POULOSE, A.; ZENO, P.; VADDEVOLU, U. B. P.; KHAN, S.; SHOMAN, M.; YAN, H.; KARKEE, M. Yolo advances to its genesis: a decadal and comprehensive review of the you only look once (yolo) series. **Artificial Intelligence Review**, Springer Science and Business Media LLC, v. 58, n. 9, jun. 2025. ISSN 1573-7462. Disponível em: <http://dx.doi.org/10.1007/s10462-025-11253-3>.

SHARMA, A. **Training the YOLOv8 Object Detector for OAK-D**. 2023. <https://pyimagesearch.com/2023/05/01/training-the-yolov8-object-detector-for-oak-d/>. Acesso em: 27 jun. 2025.

SHARMA, G. **Bounding Box Deep Learning: The Future of Video Annotation**. 2022. Acesso em: 25 maio 2025. Disponível em: <https://www.kdnuggets.com/2022/07/bounding-box-deep-learning-future-video-annotation.html>.

SILVA, A. A. N. **Avaliação de Redes Neurais e Proposta de uma Arquitetura de Comitê para Identificação de Pupas e Pragas em Sementes usando Imagens de Raios X**. Tese (Tese (Doutorado em Ciências de Computação e Matemática Computacional)) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024. Acesso em: 13 set. 2024.

SINGH, Y.; FARRELLY, C.; HATHAWAY, Q. A.; CHOUDHARY, A.; CARLSSON, G.; ERICKSON, B.; LEINER, T. The role of geometry in convolutional neural networks for medical imaging. **Mayo Clinic Proceedings: Digital Health**, v. 1, n. 4, p. 519–526, 2023. ISSN 2949-7612. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2949761223000743>.

STOREY, B. E.; WILLIAMSON, L. H.; HOWELL, S. B.; TERRILL, T. H.; BERGHAUS, R.; VIDYASHANKAR, A. N.; KAPLAN, R. M. Validation of the famacha© system in south american camelids. **Veterinary Parasitology**, v. 243, p. 85–91, 2017. ISSN 0304-4017. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0304401717302601>.

TAN, M.; LE, Q. V. **EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks**. 2019.

TAVANAEI, A.; GHODRATI, M.; KHERADPISHEH, S. R.; MASQUELIER, T.; MAIDA, A. Deep learning in spiking neural networks. **Neural networks**, Elsevier, v. 111, p. 47–63, 2019.

TEAM, T. K. **Keras Applications - NASNet Mobile**. 2018. <https://github.com/keras-team/keras/blob/v2.15.0/keras/applications/nasnet.py>. Acessado: 2025-05-03.

WYK, J. A. van; BATH, G. F. The famacha system for managing haemonchosis in sheep and goats by clinically identifying individual animals for treatment. **Veterinary research**, v. 33, n. 5, p. 509—529, 2002. ISSN 0928-4249. Disponível em: <http://www.vetres.org/articles/vetres/pdf/2002/05/08.pdf>.

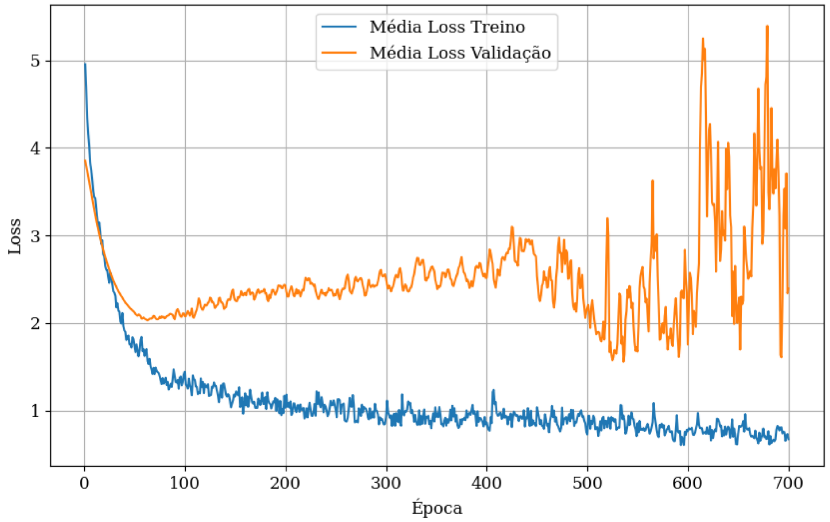
YOU, W.; SHEN, C.; WANG, D.; CHEN, L.; JIANG, X.; ZHU, Z. An intelligent deep feature learning method with improved activation functions for machine fault diagnosis. **IEEE Access**, v. 8, p. 1975–1985, 2020. Scientific Figure on ResearchGate. Available from: <https://www.researchgate.net/figure/Basic-structure-of-a-CNN_fig1_338199242> [accessed 5 Sept 2024].

ZOPH, B.; VASUDEVAN, V.; SHLENS, J.; LE, Q. V. Learning transferable architectures for scalable image recognition. In: **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**. IEEE, 2018. p. 8697–8710. Disponível em: <http://dx.doi.org/10.1109/CVPR.2018.00907>.

ZOU, Z.; CHEN, K.; SHI, Z.; GUO, Y.; YE, J. Object detection in 20 years: A survey. **Proceedings of the IEEE**, Institute of Electrical and Electronics Engineers (IEEE), v. 111, n. 3, p. 257–276, mar. 2023. ISSN 1558-2256. Disponível em: <http://dx.doi.org/10.1109/JPROC.2023.3238524>.

APÊNDICE A – RESULTADOS DO CLASSIFICADOR MOBILENETV2

Figura 22 – Média das curvas de Loss de treinamento e validação do MobileNetV2.



Fonte: Próprio autor.

Tabela 15 – Desempenho consolidado do MobileNetV2.

Métrica	Média
Acurácia	0,6061 ± 0,1767
Precisão	0,6118 ± 0,2114
Recall	0,6076 ± 0,1702
F1-Score	0,5589 ± 0,2125

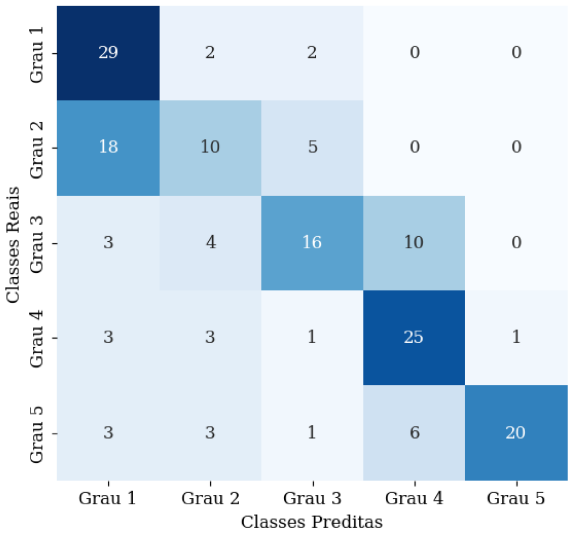
Fonte: Próprio autor.

Tabela 16 – Relatório de classificação por classe do MobileNetV2.

Classe	Precisão	Recall	F1-Score	Suporte
Grau 1	0.5179	0.8788	0.6517	33
Grau 2	0.4545	0.3030	0.3636	33
Grau 3	0.6400	0.4848	0.5517	33
Grau 4	0.6098	0.7576	0.6757	33
Grau 5	0.9524	0.6061	0.7407	33

Fonte: Próprio autor.

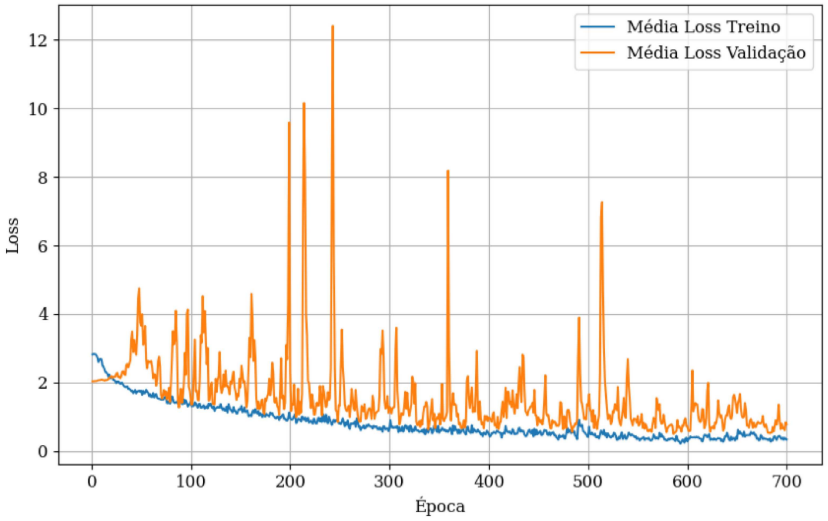
Figura 23 – Matriz de Confusão agregada do MobileNetV2.



Fonte: Próprio autor.

APÊNDICE B – RESULTADOS DO CLASSIFICADOR EFFICIENTNET-B0

Figura 24 – Média das curvas de Loss de treinamento e validação do EfficientNet-B0.



Fonte: Próprio autor.

Tabela 17 – Desempenho consolidado do EfficientNet-B0.

Métrica	Média
Acurácia	0,8303 ± 0,1159
Precisão	0,8797 ± 0,0771
Recall	0,8324 ± 0,1169
F1-Score	0,8326 ± 0,1101

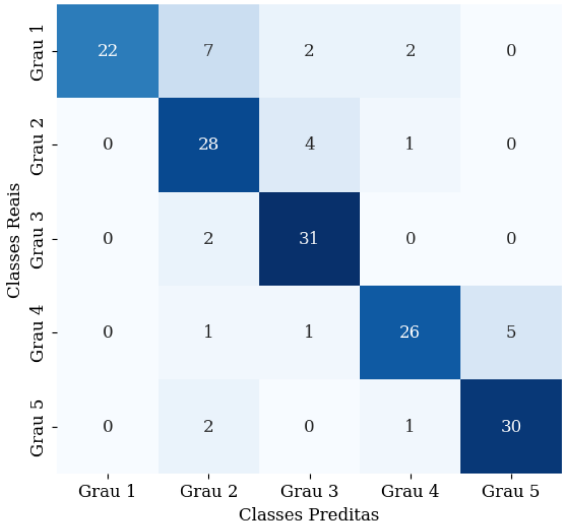
Fonte: Próprio autor.

Tabela 18 – Relatório de classificação por classe do EfficientNet-B0.

Classe	Precisão	Recall	F1-Score	Suporte
Grau 1	1.0000	0.6667	0.8000	33
Grau 2	0.7000	0.8485	0.7671	33
Grau 3	0.8158	0.9394	0.8732	33
Grau 4	0.8667	0.7879	0.8254	33
Grau 5	0.8571	0.9091	0.8824	33

Fonte: Próprio autor.

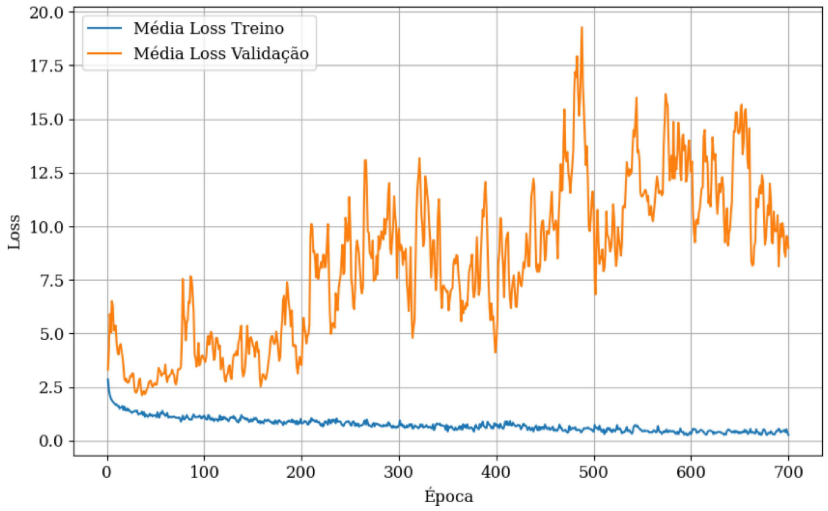
Figura 25 – Matriz de Confusão agregada do EfficientNet-B0.



Fonte: Próprio autor.

APÊNDICE C – RESULTADOS DO CLASSIFICADOR NASNETMOBILE

Figura 26 – Média das curvas de Loss de treinamento e validação do NASNetMobile.



Fonte: Próprio autor.

Tabela 19 – Desempenho consolidado do NASNetMobile.

Métrica	Média
Acurácia	0,3939 ± 0,0691
Precisão	0,3182 ± 0,0619
Recall	0,3905 ± 0,0663
F1-Score	0,2995 ± 0,0459

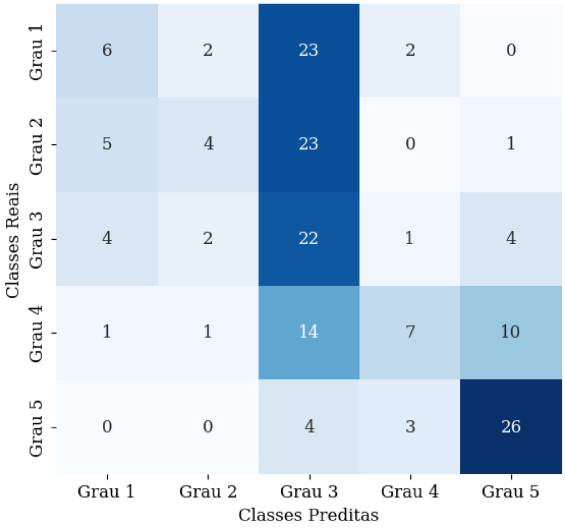
Fonte: Próprio autor.

Tabela 20 – Relatório de classificação por classe do NASNetMobile.

Classe	Precisão	Recall	F1-Score	Suporte
Grau 1	0.3750	0.1818	0.2449	33
Grau 2	0.4444	0.1212	0.1905	33
Grau 3	0.2558	0.6667	0.3697	33
Grau 4	0.5385	0.2121	0.3043	33
Grau 5	0.6341	0.7879	0.7027	33

Fonte: Próprio autor.

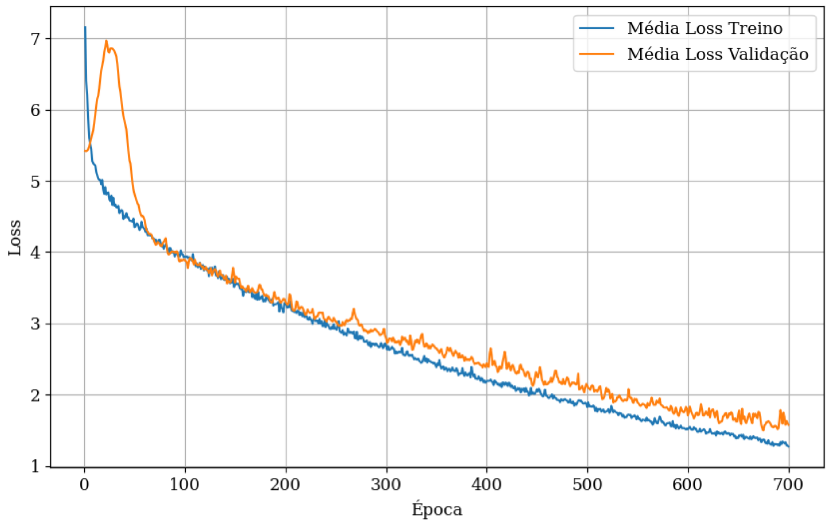
Figura 27 – Matriz de Confusão agregada do NASNetMobile.



Fonte: Próprio autor.

APÊNDICE D – RESULTADOS DA ARQUITETURA PROPOSTA

Figura 28 – Média das curvas de Loss de treinamento e validação da Arquitetura Proposta.



Fonte: Próprio autor.

Tabela 21 – Desempenho consolidado da Arquitetura Proposta.

Métrica	Média
Acurácia	0,9030 ± 0,0521
Precisão	0,9168 ± 0,0432
Recall	0,9057 ± 0,0500
F1-Score	0,9026 ± 0,0515

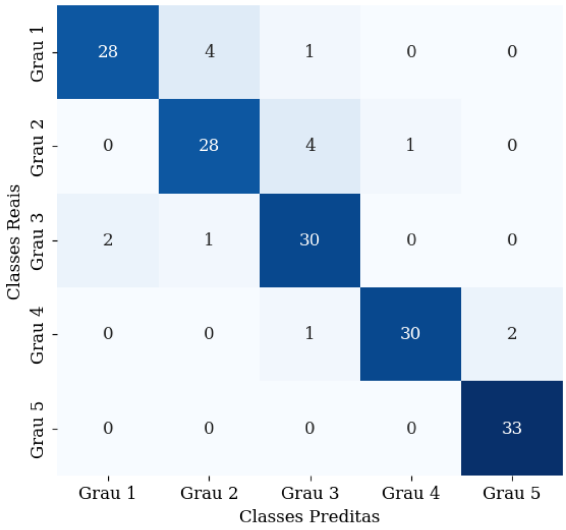
Fonte: Próprio autor.

Tabela 22 – Relatório de classificação por classe da Arquitetura Proposta.

Classe	Precisão	Recall	F1-Score	Suporte
Grau 1	0.9333	0.8485	0.8889	33
Grau 2	0.8485	0.8485	0.8485	33
Grau 3	0.8333	0.9091	0.8696	33
Grau 4	0.9677	0.9091	0.9375	33
Grau 5	0.9429	1.0000	0.9706	33

Fonte: Próprio autor.

Figura 29 – Matriz de Confusão agregada da Arquitetura Proposta.



Fonte: Próprio autor.