UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE SOBRAL

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

SAMUEL PINI MILANEZ

IMPLEMENTAÇÃO DE UM APLICATIVO IOS BASEADO EM PROCESSAMENTO DE IMAGENS COM FRAMEWORKS NATIVAS

SOBRAL

2024

SAMUEL PINI MILANEZ

IMPLEMENTAÇÃO DE UM APLICATIVO IOS BASEADO EM PROCESSAMENTO DE IMAGENS COM FRAMEWORKS NATIVAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus de 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 Júnior.

SOBRAL

2024

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)

M583i Milanez, Samuel.
IMPLEMENTAÇÃO DE UM APLICATIVO IOS BASEADO EM PROCESSAMENTO DE IMAGENS COM FRAMEWORKS NATIVAS / Samuel Milanez. – 2024.
58 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2024.
Orientação: Prof. Dr. Iális Cavalcante de Paula Júnior.

1. iOS. 2. Famacha. 3. Aplicativo. I. Título.

CDD 621.39

SAMUEL PINI MILANEZ

IMPLEMENTAÇÃO DE UM APLICATIVO IOS BASEADO EM PROCESSAMENTO DE IMAGENS COM FRAMEWORKS NATIVAS

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

Aprovada em: 23/09/2024

BANCA EXAMINADORA

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

Prof. Dr. Wendley S. Silva
Universidade Federal do Ceará (UFC)

Stefane Adna dos Santos

À minha família e ao meu amor, Ana, por sempre acreditar nos meus sonhos e ser um pilar fundamental para suas realizações. Mãe, seu cuidado e carinho, mesmo à distância, me deram forças para continuar. Descanse em paz. Pai, você foi e sempre será o alicerce de toda essa caminhada.

AGRADECIMENTOS

Mãe, por você e para você, minha Eterna Rainha. Descanse em paz.

Aos meus pais, que, através do seu incondicional suporte emocional e financeiro ao longo de todos os anos de estudo, viabilizaram minha formação.

Aos meus irmãos, cujas realizações sempre foram minhas inspirações e referências.

Ao meu amor, Ana, que, nos momentos em que me senti sozinho, longe de toda a família, foi minha companheira de vida e apoio constante.

Ao Prof. Dr. Iális Cavalcante de Paula Júnior, pela excelente orientação não apenas neste trabalho, mas em toda a minha trajetória como estudante dentro da instituição.

Aos professores desta universidade, que ao longo de muitos anos, não só contribuíram para minha formação intelectual, mas também pessoal e profissional. Muitos momentos não se limitaram à passagem de conhecimento, mas de experiências e lições de vida.

"A pesquisa é formalizada curiosidade."

(Hurston, 2010. [1942]: 211)

RESUMO

Os avanços tecnológicos têm proporcionado cada vez mais recursos para o setor agropecuário. A aplicação da Inteligência Artificial (IA) na criação de soluções que auxiliam na detecção e diagnóstico de doenças em animais pode ser um recurso valioso para reduzir perdas e minimizar os custos de produção. A IA, especialmente sua subárea de Aprendizagem de Máquina (ML), permite o desenvolvimento de modelos capazes de identificar padrões que podem ser utilizados para diagnosticar automaticamente doenças.

No contexto da criação de ovinos e caprinos, estão sendo empregados métodos para aprimorar a eficiência do tratamento parasitário, visando mitigar o aumento da resistência dos parasitas às drogas anti-helmínticas. O método *Faffa Malan Chart* (FAMACHA©) é amplamente difundido e eficaz na detecção de anemia em ovinos. No entanto, ele depende da interpretação humana, o que pode resultar em erros. Com o objetivo de automatizar o processo de avaliação da saúde desses ovinos, este estudo descreve a criação de um aplicativo móvel, implementado em *Swift* para sistemas *iOS*. O aplicativo utiliza exclusivamente *frameworks* e bibliotecas da *Apple*, que permitem a abstração de conceitos complexos de IA. Isso possibilita que desenvolvedores que não atuam nesse campo específico possam criar aplicativos inteligentes.

Para oferecer um diagnóstico do animal com base na cor dominante extraída de sua mucosa ocular, foram usadas 200 fotografias de animais com suas mucosas expostas. Cerca de dois terços dessas fotografias eram de animais não anêmicos, enquanto um terço era de animais anêmicos. Como resultado, obteve-se um aplicativo com desempenho satisfatório, capaz de detectar a mucosa do animal e classificá-la.

Palavras-chave: FAMACHA; detecção de anemia em ovinos; *Swift*; iOS; inteligência artificial; detecção ; classificação.

ABSTRACT

Technological advances have provided more and more resources for the agricultural sector. The application of Artificial Intelligence (AI) to create solutions that help detect and diagnose animal diseases can be a valuable resource for reducing losses and minimizing production costs. AI, especially its sub-area of ML, enables the development of models capable of identifying patterns that can be used to automatically diagnose diseases.

In the context of sheep and goat farming, methods are being used to improve the efficiency of parasite treatment in order to mitigate the increase in parasite resistance to anthelmintic drugs. The FAMACHA© method is widespread and effective in detecting anemia in sheep. However, it relies on human interpretation, which can result in errors. In order to automate the process of assessing the health of these sheep, this study describes the creation of a mobile application, implemented in Swift for iOS systems. The application uses exclusively *Apple* frameworks and libraries, which allow complex AI concepts to be abstracted. This makes it possible for developers who don't work in this specific field to create intelligent applications.

To offer a diagnosis of the animal based on the dominant color extracted from its ocular mucosa, 200 photographs of animals with their mucous membranes exposed were used. About two-thirds of these photographs were of non-anemic animals, while one-third were of anemic animals. The result was an application with satisfactory performance, capable of detecting the animal's mucosa and classifying it.

Keywords: *FAMACHA; detection of anemia in sheep; Swift; iOS; artificial intelligence; detection; classification.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AI	Artificial Intelligence
CNNs	Redes Neurais Convolucionais
FAMACHA©	*Faffa Malan Chart*
Ht	Hematócrito
IA	Inteligência Artificial
LMKNN	*Local Mean-based KNN*
ML	Aprendizagem de Máquina
MVC	Model-View-Controller
MVP	Produto Viável Mínimo
NT	Não Tratar
RF	Requisitos Funcionais
RN	Regras de Negócio
RNF	Requisitos Não Funcionais
SO	Sistema Operacional
T	Tratar
VC	Visão Computacional
YOLO	You Only Look Once

SUMÁRIO

1 INTRODUÇÃO

A utilização da Inteligência Artificial(IA) agropecuária tem sido uma abordagem cada vez mais comum entre produtores e pesquisadores. Isso ocorre porque a IA pode ser uma ferramenta valiosa para ajudar a melhorar a eficiência do processo produtivo e garantir a qualidade do produto (TOMAZ, 2022).

A caprinocultura e a ovinocultura são atividades difundidas em todo território nacional, mas com uma concentração, em especial do rebanho caprino, na região do Semiárido Brasileiro. No Brasil, cerca de 90% dos rebanhos caprinos e de 60% dos rebanhos ovinos estão localizados na região Nordeste, que abriga 92,5% da área semiárida do país (Embrapa, 2018).

Um dos principais problemas que a ovinocultura e caprinocultura enfrentam são doenças parasitárias gastrointestinais, onde alguns animais são acometidos com doenças em certas épocas do ano causadas por helmintos. Um desses principais parasitas é o *Haemonchus contortus*, que se alimenta do sangue no abomaso de ovinos e pode causar uma forte anemia em curto espaço de tempo (CHAGAS *et al.*, 2007).

O controle de verminose em ovinos e caprinos é realizado quase que exclusivamente com o uso de anti-helmínticos. No entanto, devido à falta de conhecimentos no que se refere à biologia e à epidemiologia dos parasitas, a maioria dos produtores não vermifugam adequadamente seus rebanhos. Na maioria das vezes, a administração de anti-helmínticos é realizada sem base técnica, visando apenas atender a um programa fixo de controle e/ou quando o rebanho é manejado, para adoção de outras práticas de manejo. Consequentemente, tem sido observada uma crescente redução na eficácia dos vermífugos (MOLENTO, 2004), resultando no aparecimento de resistência parasitária a vários grupos químicos, caracterizando o que se conhece como resistência anti-helmíntica múltipla (VIEIRA *et al.*, 1989).

Devido a essa resistência e a necessidade de vermifugação adequada, alternativas de controle têm sido estudadas. O método FAMACHA© é uma destas alternativas. Este método deve ser utilizado quando o principal parasita do rebanho for o *Haemonchus contortus*. O exame se baseia na correlação da conjuntiva ocular de pequenos ruminantes e cinco intervalos de anemia existentes em um cartão guia ilustrativo que auxilia na determinação do grau de anemia dos animais, indicado pelo exame de sangue que mede a porcentagem de células vermelhas (KAPLAN *et al.*, 2004). O uso deste método permite vermifugar apenas uma parte do rebanho, aqueles que apresentam anemia clínica. Observa-se uma diminuição nas dosificações, reduzindo os custos, a contaminação ambiental com os produtos químicos e diminuição da pressão à

resistência aos anti-helmínticos.

Desse modo, com o objetivo de explorar as ferramentas de IA nativas da *Apple* e testar as *frameworks* de desenvolvimento que permitem que profissionais e entusiastas que não são da área de IA desenvolvam aplicações inteligentes ao abstrair ideias mais complexas, bem como oferecer um método prático e acessível para diagnosticar animais que necessitam de desparasitação e prevenir erros humanos no procedimento, este trabalho visa desenvolver um aplicativo para *smartphones iOS* que possibilite o diagnóstico de anemia em ovinos em campo, até mesmo sem conexão com a internet. Esse aplicativo utilizará conceitos de Aprendizagem de Máquina(ML) para analisar a mucosa ocular e identificar se o animal precisa ou não de tratamento.

1.1 Trabalhos relacionados

Na literatura, existem iniciativas semelhantes que se empenharam em construir aplicações para dispositivos móveis que empregam princípios de ML para identificar anemia em caprinos e ovinos.

Um exemplo é o estudo realizado por Praciano (2022), que implementou o algoritmo *Local Mean-based KNN* (LMKNN) (uma variante do KNN) em linguagem *JavaScript*. Essa implementação foi integrada a um aplicativo móvel desenvolvido em *React Native*, com o objetivo de fornecer diagnósticos de animais com base na cor predominante extraída de suas mucosas oculares. Os resultados mostraram um desempenho satisfatório, com o algoritmo de classificação alcançando uma média de precisão de 78% ao diferenciar as classes entre Tratar (T) e Não Tratar (NT).

Outro estudo relevante foi realizado por ALMEIDA (2018), que se concentrou na utilização de Redes Neurais Profundas para a segmentação e classificação da membrana ocular de ovinos anêmicos. Utilizando imagens distintas da membrana ocular de ovinos, o estudo criou um modelo para realizar esse procedimento, alcançando uma precisão de 97,29% na segmentação e uma taxa de precisão de 95,23% na classificação. O modelo foi integrado a um aplicativo móvel para testes, demonstrando um desempenho sólido, embora tenha apresentado limitações práticas quando aplicado em campo.

No estudo realizado por Demoliner e Alves (2017), os pesquisadores desenvolveram um aplicativo móvel com o objetivo de informatizar o método FAMACHA©, tornando o processo de diagnóstico animal mais preciso e ágil, permitindo que até mesmo indivíduos sem experiência

pudessem adotar a técnica. Entre os vários algoritmos empregados no estudo, o *Naive Bayes* apresentou os resultados mais promissores na identificação do nível FAMACHA©, exibindo uma taxa de acertos de 66,3% ao diferenciar as classes de T e NT.

1.2 Objetivos

1.3 Objetivo Geral

Este trabalho tem como objetivo explorar as *frameworks* e bibliotecas nativas da *Apple* voltadas para a construção de aplicações inteligentes. Para tanto, será desenvolvida uma aplicação que utilizará um algoritmo de processamento de imagens para detectar anemia em ovinos por meio de fotografias da mucosa ocular desses animais, analisando a cor dominante da mucosa ocular para realizar essa detecção.

1.3.1 Objetivos Específicos

- Avaliar as potencialidades da *framework CreateML* que através de um alto nível de abstração permite o treinamento de modelos de ML;
- Avaliar as potencialidades das *framework CoreML* que através de um alto nível de abstração permite a implementação aplicações inteligentes (que consomem/-consultam modelos de ML);
- Avaliar o desempenho dos modelos treinados em imagens previamente coletadas e classificadas;
- Estudar a integração da linguagem de programação *Swift* com as *frameworks* nativas da *Apple* (*CreateML, CoreML*) para o desenvolvimento de um aplicativo *mobile* baseado em processamento de imagens;
- Propor uma alternativa de controle seletivo de verminoses em caprinos e ovinos que seja automatizada e não dependente da interpretação humana;
- Avaliar o aplicativo *mobile* desenvolvido que é capaz de auxiliar no processo de decisão do produtor rural;

2 FUNDAMENTAÇÃO TEÓRICA

Este Capítulo é dedicado à Fundamentação Teórica, onde são apresentados e discutidos os conceitos e teorias que alicerçam a solução proposta neste trabalho. A Fundamentação Teórica é uma seção essencial que permite situar o estudo dentro de um contexto acadêmico mais amplo, oferecendo uma análise detalhada das bases teóricas e tecnológicas que fundamentam a pesquisa.

Nesta Seção, serão explorados de maneira aprofundada os conceitos e tecnologias aplicados ao longo do desenvolvimento do trabalho. Os tópicos abordados englobam desde os conceitos relacionados ao campo da veterinária, como o método *Faffa Malan Chart* (FAMACHA©), que desempenha um papel crucial na avaliação clínica de animais, até os conceitos tecnológicos, como os princípios de Inteligência Artificial (IA) e as *frameworks* utilizadas para o desenvolvimento do aplicativo proposto. Ao integrar conhecimentos de diferentes áreas, esta Seção visa construir um entendimento robusto que sustente as escolhas metodológicas e tecnológicas adotadas, evidenciando a relevância e a coerência da solução desenvolvida.

2.1 Hematócrito

O Hematócrito (Ht) é um exame laboratorial que mede o volume de glóbulos vermelhos no sangue em relação ao volume total de sangue. Ele é expresso como uma porcentagem do volume total. Em termos mais simples, é uma medida da densidade dos glóbulos vermelhos no sangue. Segundo o artigo de Matthew R. Pincus e outros, "Evaluation of Anemia in Children", publicado na revista *American Family Physician* em 2010, define o hematócrito da seguinte forma: "O hematócrito é a porcentagem de volume ocupado pelos eritrócitos no sangue total e é uma medida indireta do número de glóbulos vermelhos."

O Ht, em tratamentos veterinários, é frequentemente solicitado como parte de uma análise de sangue mais ampla, sendo comum para o diagnóstico e acompanhamento de condições adversas em animais, como anemia, policitemia e outros distúrbios. Neste estudo, fotografias da membrana mucosa ocular de caprinos foram usadas em conjunto com as medições de Ht fornecidas por (ALMEIDA, 2021). Fotografias com valores de Ht iguais ou inferiores a 23 foram consideradas como indicativas de animais anêmicos, identificados como casos de tratamento necessário - Tratar (T), enquanto medições de Ht iguais ou superiores a 24 foram associadas a animais saudáveis, não necessitando de vermifugação, e foram classificadas como não tratamento

necessário - Não Tratar (NT).

2.2 FAMACHA©

Haemonchus contortus é um helminto que pertence à superfamília *Trichostrongyloidea* e é considerado o principal parasita de pequenos ruminantes em todas as regiões brasileiras (AROSEMENA *et al.*, 1999; AMARANTE *et al.*, 2004; RAMOS *et al.*, 2004; GIGLIOTI *et al.*, 2006). Ele é um parasita hematófago, ou seja, alimenta-se de sangue, e localiza-se no abomaso dos ovinos, onde se desenvolve e se reproduz.

Atualmente não se recomenda mais a aplicação de vermífugos em todo rebanho, pois já foi comprovado que somente uma parte dos animais necessita realmente de vermifugação, aproximadamente 17% das fêmeas secas, 29% das fêmeas gestantes e 55% das fêmeas lactantes (MALAN *et al.*, 2001).

Sendo assim, através de um controle seletivo é possível reduzir o número de tratamentos (50-80%) e o gasto com vermífugos na mesma proporção, o que auxilia na diminuição do desenvolvimento da resistência aos anti-helmínticos (EMBRAPA CAPRINOS E OVINOS, 2024).

O FAMACHA© é o método mais indicado para o controle seletivo em regiões onde o verme predominante é *Haemonchus contortus*, pois ao se alimentar do sangue dos animais, causa diferentes graus de anemia que podem ser classificados com auxílio de um cartão colorido.

Figura 1 – Utilização do cartão do método FAMACHA©

Fonte: Retirado de (EMBRAPA CAPRINOS E OVINOS, 2024)

Na Figura 1 temos a demonstração do uso do cartão FAMACHA©, em que os cinco

graus de coloração no cartão direcionam a vermifugação dos animais, onde os graus 1 e 2 são de animais com coloração vermelho vivo, ou seja, sem traços de anemia. A partir do grau 3, já é indicada a vermifugação, que é imprescindível nos graus 4 e 5. Porém, quando a mucosa apresenta palidez intensa (grau 5), é recomendável que o animal tenha suporte com alimentação reforçada em proteínas (volumosa ou concentrada), ferro oral ou injetável e suplementos a base de aminoácidos, conforme a indicação da bula. Havendo condições (infraestrutura/veterinários) a transfusão de sangue é recomendada. Animais com grau de anemia 5, só devem ser vermifugados quando o quadro de anemia estiver amenizado (EMBRAPA CAPRINOS E OVINOS, 2024).

2.3 Inteligência Artificial

A IA, surgida na década de 1950, tem sua origem praticamente confundida com a própria origem do computador. Mais precisamente, em 1956, ocorreu a *Darthmouth College Conference*, que é considerada o marco inicial da IA. Os pesquisadores reconhecidos como pais da área, como John MacCarthy, Marvin Minsky, Alan Newell e Herbert Simon, entre outros, participaram desse evento e tiveram trajetórias científicas que estabeleceram marcos nesse fascinante domínio da Computação (SICHMAN, 2021).

Uma das primeiras tentativas de definição dos objetivos da IA foi proposta por Rich e Knight (1991), em que foi dito o seguinte: o objetivo da IA é desenvolver sistemas para realizar tarefas que, no momento, são mais bem realizadas por seres humanos que por máquinas, ou não possuem solução algorítmica viável pela computação convencional (RICH; KNIGHT, 1991).

Desde então, essa área da Ciência da Computação tem evoluído e hoje permeia nosso cotidiano na automatização dos carros e das indústrias, nos computadores, na medicina e em vários outros aspectos presentes em nossa vida.

A IA possui áreas que neste trabalho serão utilizadas como ferramentas para atingir o objetivo proposto. Dentre esses áreas, entraremos no campo da Visão Computacional (VC), que é um campo da IA que permite que computadores e sistemas obtenham informações significativas de imagens digitais, vídeos e outras entradas visuais – e tomem ações ou façam recomendações com base nessas informações. Se a IA permite que os computadores pensem, a visão computacional permite que eles vejam IBM (2024).

Também dentro dessa áreas, temos a Aprendizagem de Máquina (ML),que é uma subárea da IA focada no desenvolvimento de algoritmos e técnicas que permitem aos computadores aprender a partir de dados. Em vez de serem explicitamente programados para realizar uma

tarefa, os sistemas de ML utilizam dados para treinar modelos matemáticos que podem fazer previsões ou tomar decisões. Exemplos incluem redes neurais, árvores de decisão e algoritmos de clustering.

2.4 Aprendizagem de Máquina

A ML é o subconjunto da IA que se concentra na construção de sistemas que aprendem, ou melhoram o desempenho, com base nos dados que consomem Oracle (2024). Em 1959, Arthur Lee Samuel, engenheiro do MIT definiu o aprendizado de máquina como "um campo de estudo que dá aos computadores a habilidade de aprender sem terem sido programados para tal" (SAMUEL, 1959).

Em uma definição mais completa, o ML é uma técnica de ciência de dados que permite que os computadores usem os dados existentes para prever futuros comportamentos, resultados e tendências. Usando o ML, os computadores aprendem sem serem explicitamente programados. As ferramentas de ML usam sistemas de IA que podem identificar padrões e criar associações a partir da experiência com os dados (Microsoft, 2024).

Existem três tipos principais de algoritmos de ML: Supervisionado, Não Supervisionado e por Reforço, que de acordo com (LUDERMIR, 2024) pode ser definidos da seguinte maneira:

- No **Aprendizado Supervisionado**, para cada exemplo apresentado ao algoritmo de aprendizado é necessário apresentar a resposta desejada (ou seja, um rótulo informando a que classe o exemplo pertence, no caso de um problema de classificação de imagens, por exemplo, como distinguir imagens de gatos e de cachorros). Cada exemplo é descrito por um vetor de valores (atributos) e pelo rótulo da classe associada. O objetivo do algoritmo é construir um classificador que possa determinar corretamente a classe de novos exemplos ainda não rotulados. Para rótulos de classe discretos, esse problema é chamado de classificação e para valores contínuos como regressão. Esse método de aprendizado é o mais utilizado.
- No **Aprendizado Não Supervisionado**, os exemplos são fornecidos ao algoritmo sem rótulos. O algoritmo agrupa os exemplos pelas similaridades dos seus atributos. O algoritmo analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou clusters. Após a determinação dos agrupamentos, em geral, é necessária uma análise para determinar o que cada agrupamento

significa no contexto problema sendo analisado.

- No **Aprendizado por Reforço**, o algoritmo não recebe a resposta correta mas recebe um sinal de reforço, de recompensa ou punição. O algoritmo faz uma hipótese baseado nos exemplos e determina se essa hipótese foi boa ou ruim. Aprendizado por Reforço é bastante utilizado em jogos e robótica, e foi a técnica utilizada no AlphaGo.

2.5 Visão Computacional

VC, uma subárea da IA, trata de como os computadores percebem e interpretam o ambiente ao seu redor, extraindo informações úteis de imagens capturadas por câmeras, sensores, scanners, entre outros dispositivos. Essas informações possibilitam o reconhecimento, a manipulação e a análise dos objetos presentes em uma imagem (BALLARD, 1982).

Podemos considerar a VC como uma ciência relativamente nova. Uma das primeiras menções a essa área data de 1955, quando Selfridge referiu-se a "...olhos e ouvidos para o computador" (DAVID; SELFRIDGE, 1962). Nos anos 70, começaram os primeiros trabalhos combinando VC e IA. Naquela época, havia a expectativa de que, em breve, seria possível replicar completamente a visão humana em máquinas. Entre os estudos notáveis dessa época estão The Psychology of Computer Vision (WINSTON, 1975) e A Framework for Representing Knowledge (MINSKY, 1974), ambos de 1975.

No entanto, com o avanço das pesquisas nas décadas seguintes, percebeu-se que a complexidade da VC era muito maior do que inicialmente imaginado, principalmente devido à falta de modelos e informações que representassem a interpretação das imagens pelo cérebro humano. O olho humano é capaz de perceber e interpretar objetos em uma imagem rapidamente, graças ao córtex visual do cérebro, uma das áreas mais complexas do sistema de processamento cerebral. Alguns cientistas focam seus estudos em entender o funcionamento dessa parte do cérebro para aplicar esses conceitos à visão computacional. É o que pesquisadores do MIT descrevem como "ensinar computadores a enxergarem como humanos"(VISIONARY. . . ,).

Assim, a VC oferece aos computadores uma vasta quantidade de informações precisas a partir de imagens e vídeos, permitindo que realizem tarefas inteligentes e se aproximem da capacidade de percepção da inteligência humana.

2.5.1 Algoritmos de detecção

Os algoritmos de detecção desempenham um papel fundamental em uma ampla gama de aplicações em visão computacional. Esses algoritmos são projetados para identificar e localizar objetos específicos em imagens ou vídeos, fornecendo informações essenciais para a tomada de decisões automatizada em diversas áreas.

De acordo com uma revisão realizada por (GERÓNIMO *et al.*, 2010), os algoritmos de detecção são cruciais para sistemas de assistência ao motorista, como detecção de pedestres, detecção de veículos e reconhecimento de sinalização de trânsito. Esses sistemas dependem da capacidade precisa de identificar objetos na cena para garantir a segurança dos ocupantes do veículo e de outros usuários da estrada.

Além disso, conforme discutido por (ZHAO *et al.*, 2019), os algoritmos de detecção desempenham um papel vital em aplicações de visão computacional baseadas em aprendizado profundo. Esses algoritmos utilizam redes neurais convolucionais profundas para localizar e identificar objetos em imagens ou vídeos, permitindo avanços significativos em áreas como vigilância por vídeo, diagnóstico médico e veículos autônomos.

Segundo (WU *et al.*, 2020), os algoritmos de detecção continuam a evoluir com o avanço da tecnologia de aprendizado profundo, proporcionando melhorias significativas em precisão e desempenho. Esses avanços têm implicações importantes em uma variedade de domínios, incluindo segurança, medicina, robótica e transporte.

Em resumo, os algoritmos de detecção são ferramentas essenciais em visão computacional, capacitando sistemas automatizados a identificar e localizar objetos com precisão em imagens ou vídeos. Com o contínuo progresso na área de aprendizado profundo, espera-se que esses algoritmos desempenhem um papel cada vez mais crucial em uma ampla gama de aplicações futuras.

2.5.2 Algoritmos de classificação

Algoritmos de classificação são algoritmos de aprendizagem supervisionada onde o objetivo é prever uma classe ou rótulo associado com uma variável de entrada contendo determinados atributos. Inicialmente, o algoritmo é treinado com um conjunto de dados com classes conhecidas, podendo estes dados estar divididos em somente duas (classificação binária) ou em várias classes (classificação multiclasse) (FONTANA, 2020).

Os algoritmos de classificação de duas classes (binária) dividem os dados em duas categorias. Eles são úteis para perguntas que tenham apenas duas respostas possíveis que sejam mutuamente exclusivas, incluindo perguntas do tipo sim/não. Os algoritmos de classificação multiclasse (multinomial) dividem os dados em três ou mais categorias. Eles são úteis para perguntas que tenham três ou mais respostas possíveis que sejam mutuamente exclusivas (MICROSOFT, 2023).

2.6 Redes Neurais Convolucionais

Redes Neurais Convolucionais (CNNs) são uma arquitetura de redes neurais que se destacam em tarefas de processamento de dados com estrutura de *grid*, como imagens e vídeos. Estas redes são projetadas para reconhecer padrões espaciais e temporais, tornando-as extremamente eficazes em visão computacional e reconhecimento de padrões. De acordo com LeCun *et al.* (2015), "redes neurais convolucionais são uma classe de redes neurais profundas que são muito eficazes para reconhecer padrões em dados de imagens". Essa eficácia é atribuída à sua capacidade de extrair características hierárquicas dos dados, processando informações através de camadas convolucionais que capturam características locais e globais da entrada.

Os componentes fundamentais das CNNs incluem camadas convolucionais, camadas de *pooling* e camadas totalmente conectadas. As camadas convolucionais aplicam filtros para detectar características específicas, como bordas e texturas, enquanto as camadas de *pooling* reduzem a dimensionalidade dos dados e a complexidade computacional. Como observado por Goodfellow *et al.* (2016), "CNNs são projetadas para explorar a estrutura espacial dos dados de entrada através de operações de convolução, o que permite capturar as dependências locais e reduzir a dimensionalidade dos dados". A combinação dessas camadas permite que as CNNs aprendam representações de características em múltiplos níveis, tornando-as extremamente eficazes em tarefas como reconhecimento de objetos, detecção de bordas e segmentação de imagens.

Além disso, as CNNs são notáveis por sua capacidade de generalização e escalabilidade, o que as torna aplicáveis em uma variedade de domínios, desde o processamento de linguagem natural até a análise de imagens médicas. De acordo com Krizhevsky *et al.* (2012), "as redes neurais convolucionais têm mostrado um desempenho superior em muitas tarefas de visão computacional, especialmente quando treinadas com grandes conjuntos de dados".

2.6.1 YOLO

You Only Look Once (YOLO) é um algoritmo de detecção de objetos em tempo real que utiliza CNNs para identificar e localizar objetos em imagens. De acordo com Redmon *et al.* (2016), "YOLO é uma abordagem revolucionária para a detecção de objetos, pois trata a detecção como um problema de regressão único, ao invés de uma combinação de tarefas de reconhecimento e localização". Esse método permite que YOLO processe uma imagem em uma única passagem pela rede, o que resulta em uma detecção de objetos rápida e eficiente. Como destacado por Redmon e Farhadi (2018), "a principal inovação do YOLO é sua capacidade de realizar detecção de objetos em tempo real, utilizando uma única rede neural convolucional para prever caixas delimitadoras e probabilidades de classe simultaneamente".

Desde sua introdução, YOLO passou por várias iterações e melhorias (como *YOLOv2, YOLOv3, e YOLOv4*), cada uma incorporando avanços em arquiteturas de CNNs e técnicas de treinamento para melhorar a precisão e a velocidade da detecção. No contexto desse trabalho, foi utilizado a *YOLOv2*.

De maneira sucinta, pode-se definir que CNNs são a base para a arquitetura do YOLO, fornecendo a estrutura para extração de características e processamento de imagens.

2.7 Desenvolvimento de aplicativos

2.7.0.1 Arquitetura MVC

Model-View-Controller (MVC) é um padrão arquitetural usado para implementar interfaces de usuário. O padrão MVC divide uma aplicação em três componentes interconectados: o Modelo, que é responsável pelos dados e pela lógica de negócios da aplicação; a Visão, que é responsável por exibir os dados e os elementos da interface do usuário; e o Controlador, que lida com a entrada do usuário e atualiza o Modelo. A separação de preocupações proporcionada pelo MVC ajuda a gerenciar a complexidade ao permitir que os desenvolvedores trabalhem em diferentes componentes de forma independente. O Modelo representa a funcionalidade central da aplicação e encapsula os dados, enquanto a Visão fornece uma representação visual desses dados. O Controlador media entre o Modelo e a Visão, garantindo que as entradas do usuário sejam processadas e o estado da aplicação seja atualizado de acordo. Esse padrão promove a reutilização de código e a escalabilidade, tornando-o uma escolha popular no desenvolvimento

moderno de software (FOOTE; YODER, 1997).

A escolha da arquitetura MVC para o desenvolvimento deste aplicativo se justifica por sua simplicidade e eficácia em cenários de desenvolvimento de Produto Viável Mínimo (MVP). A arquitetura MVC é amplamente utilizada devido à sua capacidade de modularizar o código e facilitar tanto o desenvolvimento quanto a manutenção de aplicações.

2.7.0.2 Swift

Swift é uma linguagem de programação desenvolvida pela *Apple Inc*, compilada, multiparadigma de propósito geral que foi criada para *iOS*, *OS X*, *watchOS*, *tvOS* e *Linux*. É uma linguagem construída usando uma abordagem moderna para segurança, desempenho e padrões de design de software. O objetivo do projeto sempre foi construir uma linguagem de fácil escrita e manutenção (TEAM, 2024).

O desenvolvimento da linguagem *Swift* foi iniciado em julho de 2010 por Chris Lattner, com a eventual colaboração de muitos outros programadores da *Apple*. *Swift* pegou ideias de linguagem como *Objective-C*, *Rust*, *Haskell*, *Ruby*, *Python*, *C#*, *CLU* e muitos outros. Em 2 de junho de 2014, o aplicativo Worldwide Developers Conference (WWDC) tornou-se um dos primeiros aplicativos lançado publicamente escrito em Swift (TEAM, 2024).

Atualmente, a linguagem é um projeto open-source, o que significa que qualquer pessoa pode baixar o código e as compilações em desenvolvimento para ver e entender detalhes de projeto da linguagem. Os desenvolvedores interessados também podem contribuir com o projeto, registrar bugs, participar da comunidade e efetuar suas próprias correções e aprimoramentos. Entretanto, é importante ressaltar que, para o desenvolvimento de produção da *App Store*, deve ser sempre utilizado as versões estáveis do *Swift* que estão incluídas no *Xcode* (IDE própria da *Apple*). Isso é um requisito para o envio do aplicativo para a loja (TEAM, 2024).

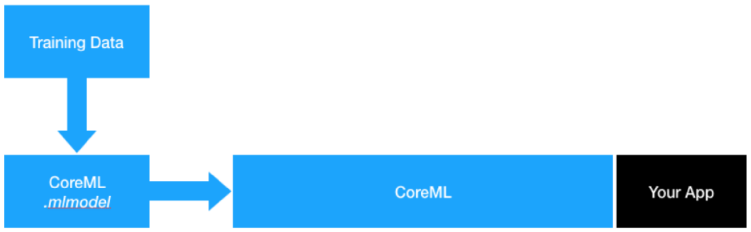
O *Swift* destinou-se a substituir as linguagens baseadas em *C (C, C++ e Objective-C)* e, como tal, é comparável a essas linguagens em desempenho para a maioria das tarefas. O desempenho também é previsível e consistente, não apenas rápido.

2.7.0.3 CreateML

O *CreateML* é uma ferramenta da *Apple* baseado em *Swift* para criar e treinar modelos de ML. Tem dois componentes principais: uma *framework* e uma aplicação desktop. Inicialmente surgiu como aplicação no *macOS* e foi concebido para se tornar uma *framework* em

seguida(GELDARD *et al.*, 2019). Na Figura 2, podemos observar que o treinamento de modelos via *CreateML* nos retorna um modelo no formato *.mlmodel*, que, através de outra ferramenta, que será discutida posteriormente, a *CoreML*, é injetado na aplicação.

Figura 2 – Ilustração da atuação do CreateML no treinamento de modelos e sua atuação com CoreML.



Fonte: Retirado de (GELDARD *et al.*, 2019)

A aplicação permite criar e treinar modelos de aprendizagem automática num ambiente gráfico e utiliza o mesmo subsistema de ML subjacente ao qual a *framework CreateML* dá acesso, integrado numa interface de utilização(GELDARD *et al.*, 2019).

A framework CreateML foi a forma original do CreateML e foi anunciada pela Apple na sua conferência WWDC em 2018. Foi originalmente concebido para permitir a criação de modelos de aprendizagem automática no Xcode Playgrounds. Desde então, evoluiu para ser uma framework de aprendizado de máquina de uso mais geral que não foi projetada apenas para construir modelos, mas também para trabalhar e manipular modelos em geral(GELDARD *et al.*, 2019).

Quase tudo o que você pode fazer com o aplicativo *CreateML* também pode ser feito usando o *framework CreateML* diretamente, com código *Swift*.

A aplicação *CreateML* suporta o treino de modelos a partir dos mais variados tipos de dados. Na Figura 3, temos esse cenário de maneira visual, onde, a partir de imagens, tabelas e textos, temos a geração de um *.mlmodel*. A aplicação *CreateML* suporta o treino dos seguintes tipos de modelos:

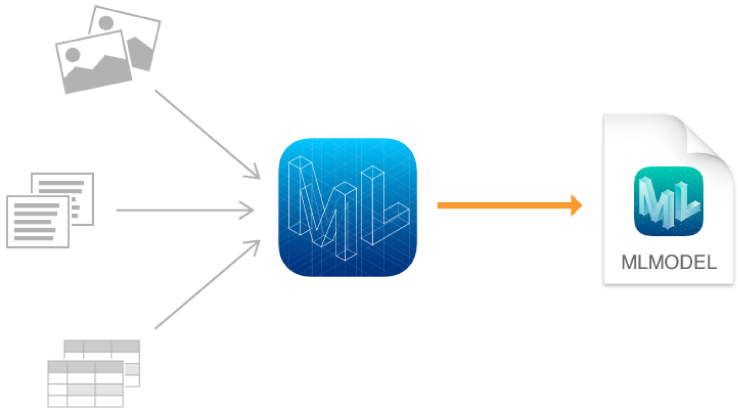
- Classificador de imagens
- Classificador de sons
- Classificador de objectos
- Detector de atividades
- Classificador de textos
- Classificador de palavras

- Etiquetador tabular
- Regressor tabular
- Classificador tabular

A framework *CreateML* tem as seguintes abordagens gerais:

- Classificação de imagens
- Deteção de objetos nas imagens
- Classificação de texto
- Modelos de etiquetagem de palavras
- Incorporação de palavras, para comparação de cadeias de caracteres
- Classificação de sons
- Classificação de atividades
- Sistemas de recomendação
- Estruturas de dados para armazenar dados classificados, valores contínuos e tabelas de dados
- Métricas de precisão do modelo
- Tratamento de erros para aprendizagem automática
- Armazenamento de metadados para aprendizagem automática

Figura 3 – Ilustração da atuação do CreateML no treinamento de modelos a partir de imagens, tabelas e textos.



Fonte: Retirado de (APPLE., 2024)

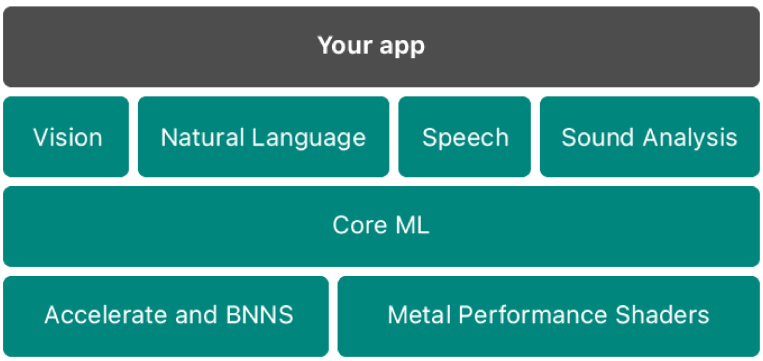
No âmbito deste trabalho, que abrange tanto um modelo de detecção quanto um de classificação, o *CreateML* oferece suporte a uma variedade de modelos de detecção de objetos, incluindo:

- YOLO (You Only Look Once): Suporte a *YOLO* para detecção de objetos. *YOLO* é uma família de modelos de rede neural profunda projetada para a detecção em tempo real de objetos em imagens e vídeos
- SSD (Single Shot MultiBox Detector): Arquitetura popular para detecção de objetos em tempo real, que é conhecida por sua velocidade e precisão.
- Faster R-CNN (Region-based Convolutional Neural Network): Uma arquitetura clássica para detecção de objetos que opera em duas etapas: propõe regiões de interesse e, em seguida, classifica essas regiões.
- RetinaNet: Uma arquitetura de detecção de objetos que resolve o problema de desequilíbrio de classes em conjuntos de dados de detecção de objetos.
- Mask R-CNN: Uma extensão do *Faster R-CNN* que também gera máscaras segmentadas para objetos detectados, útil para tarefas de segmentação semântica.

2.7.0.4 CoreML

CoreML é a ferramenta utilizada para rodar modelos treinados de ML nos dispositivos, o que essencialmente significa que ela tem a função de ficar perguntando ao modelo sobre predições. Além de lidar com o carregamento do modelo, ela também possibilita um modelo padrão de dar entradas e saídas enquanto está sendo utilizado (GELDARD *et al.*, 2019).

Figura 4 – Ilustração da atuação do CoreML entre o modelo e a aplicação.



Fonte: Retirado de (APPLE., 2024)

Através dessa ferramenta também é possível que seja feito correções e treinamentos dos modelos no próprio device, sem a necessidade de alterá-lo em uma ferramente como à CreateML, por exemplo.

Como pode ser observado na Figura 4, *CoreML* é base de outras *frameworks* da

Apple relacionadas à IA, como a *Vision, Natural Language, Speech e Sound Analysis*. Também é baseada em *frameworks* matemáticas e gráficas, como a *Accelerate, BNNS e Metal*. Essas *frameworks* no qual ela se baseia podem ser usadas diretamente para obter o mesmo resultado, porém esse seria o caminho mais trabalhoso. Assim como é possível utilizar a *CoreML* para fazer o trabalho realizado por *frameworks* específicas como a *Vision*, por exemplo, mas este também seria um caminho mais trabalhoso (GELDARD *et al.*, 2019).

Existem vários domínios específicos de IA no qual a *Apple* não fornece um *framework* direta. Nesses casos utilizaremos a *CoreML* diretamente. Porém é importante saber que, sempre que utilizamos a *Vision* ou outras *frameworks* de IA da *Apple*, internamente estamos usando a *CoreML* (GELDARD *et al.*, 2019).

CoreML possui uma maneira padrão de lidar com os modelos de ML na plataforma da *Apple*. Isso significa que, após o entendimento do seu funcionamento, é possível inserir um grande número de funcionalidades de IA em um app seguindo os mesmos passos, independentemente do modelo. Todo o processamento realizado pela *CoreML* é realizado localmente no dispositivo, sem necessidade de internet (GELDARD *et al.*, 2019).

Principais vantagens:

- Está sempre disponível. Todo a aprendizagem e os dados necessários estão nos dispositivos dos usuários. Desse modo, ainda que esteja em uma área sem sinal, ainda assim ele poderá performar aprendizagem.
- Economia de dados móveis. Não ocorre o *download* e *upload* de dados.
- Não precisa de infraestrutura de rede.
- Preserva a privacidade. Uma vez que tudo é executado localmente, ao menos que o dispositivo esteja "contaminado", não existe chance alguma de alguém extrair qualquer informação.

Principais desvantagens:

- É mais difícil atualizar os modelos. Se o modelo da aplicação for atualizado ou trocado, é necessário que esse novo modelo seja enviado aos usuários, seja fazendo uma nova *build*, seja utilizando os recursos de personalização local de modelos (que possui diversas lmitações).
- Aumenta o tamanho do aplicativo. Como os modelos são armazenados localmente, consequentemente isso aumenta o tamanho das aplicações.

2.7.0.5 CoreData

O *Core Data* é utilizado para guardar dados permanentes de uma aplicação para uso *offline*, colocar dados temporários em cache e adicionar funcionalidade de desfazer em um único dispositivo. Para sincronizar dados em vários dispositivos de uma única conta *iCloud*, o *Core Data* espelha automaticamente o esquema de dados para um *container CloudKit* (INC., 2024).

Através do editor do modelo de dados do *Core Data*, define-se os tipos e as relações dos dados, gerando as respectivas definições de classe. O *Core Data* pode então gerenciar instâncias de objetos em tempo de execução, proporcionando os seguintes recursos(INC., 2024):

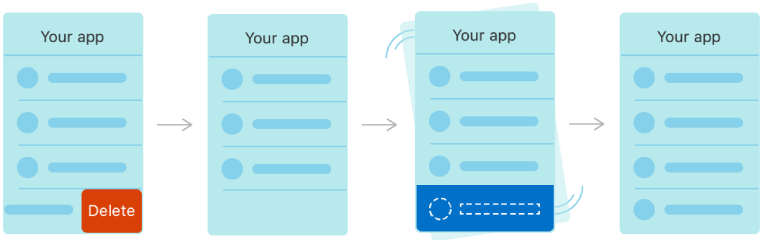
- **Persistência**:

  O *Core Data* abstrai os detalhes do mapeamento de objetos para um armazenamento, facilitando o salvamento de dados em *Swift e Objective-C* sem a necessidade de administrar diretamente um banco de dados.

- **Desfazer e refazer alterações individuais e em lote**:

  O gerenciador de desfazer do *Core Data* rastreia alterações e pode revertê-las individualmente, em grupos ou todas de uma vez, simplificando a adição de suporte a desfazer e refazer no aplicativo. Na Figura 5, temos esse processo ilustrado, onde um dado que foi previamente excluído foi restaurado.

Figura 5 – Ilustração da atuação do *Core Data* em desfazer e fazer alterações.

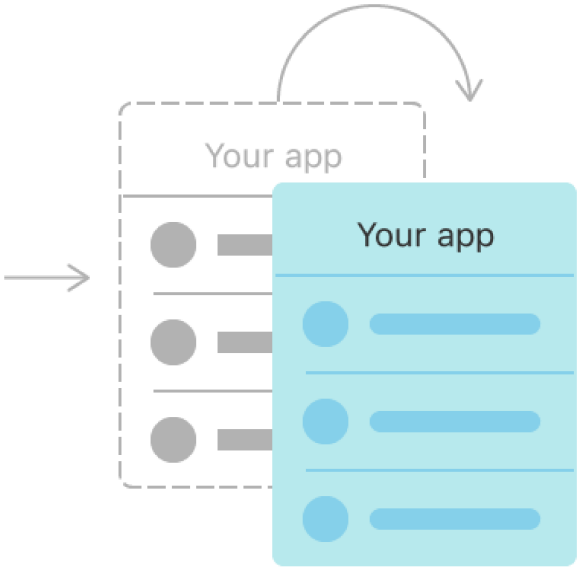


Fonte: Retirado de (INC., 2024)

- **Tarefas de dados em segundo plano**:

  O *Core Data* permite a execução de tarefas de dados que potencialmente bloqueiam a interface do usuário, como a análise de *JSON* em objetos, em segundo plano, na Figura 6 temos uma ilustração desse cenário. Posteriormente, é possível armazenar em cache ou guardar os resultados para reduzir as viagens de ida e volta ao servidor.

Figura 6 – Ilustração da atuação do *Core Data* em segundo plano.



Fonte: Retirado de (INC., 2024)

- **Sincronização de visualizações**:

  O *Core Data* ajuda a manter as exibições e os dados sincronizados, fornecendo fontes de dados para exibições de tabelas e coleções.

- **Controle de versão e migração**:

  O *Core Data* inclui mecanismos para criar versões do modelo de dados e migrar os dados do usuário à medida que a aplicação evolui.

O *Core Data* normalmente diminui em 50% a 70% a quantidade de código que você escreve, para suportar a camada de modelo. Isso se deve principalmente aos seguintes recursos integrados que não é necessário implementar, testar ou otimizar(INC., 2024):

- Controle de alterações e gerenciamento integrado de desfazer e refazer, além da edição básica de texto.
- Manutenção da propagação da mudança, incluindo a manutenção da consistência dos relacionamentos entre os objetos.
- Carregamento preguiçoso de objetos, futuros parcialmente materializados (falha) e compartilhamento de dados *copy-on-write* para reduzir a sobrecarga.
- Validação automática de valores de propriedade. Os objetos gerenciados estendem os métodos de validação de codificação de valor-chave padrão para garantir que os valores individuais estejam dentro de intervalos aceitáveis, de modo que as combinações de valores façam sentido.

- Ferramentas de migração de esquema, que simplificam as alterações de esquema e permitem que você execute uma migração de esquema local eficiente.
- Integração opcional com a camada do controlador do aplicativo para oferecer suporte à sincronização da interface do usuário.
- Agrupamento, filtragem e organização de dados na memória e na interface do usuário.
- Suporte automático para armazenar objetos em repositórios de dados externos.
- Compilação de consulta sofisticada. Em vez de escrever *SQL*, você pode criar consultas complexas associando um objeto *NSPredicate* a uma solicitação de busca.
- Rastreamento de versão e bloqueio otimista, para oferecer suporte à resolução automática de conflitos de vários gravadores.
- Integração efetiva com as cadeias de ferramentas *macOS e iOS*.

2.7.1 Levantamentos de requisitos

Em todo projeto de desenvolvimento de software, a identificação e compreensão dos requisitos são elementos essenciais para garantir o progresso eficiente e a conclusão bem-sucedida do projeto.

Conforme descrito no documento sobre Aquisição de Soluções em Tecnologia da Informação do CONSELHO NACIONAL DE SECRETÁRIOS DE SAÚDE – CONASS (2020), requisitos de *software* referem-se a solicitações, desejos e/ou necessidades específicas do *software* para garantir seu funcionamento conforme esperado, atendendo às demandas e resolvendo problemas reais. O mesmo texto também estabelece a estrutura fundamental para o processo de levantamento de requisitos de *software*, dividindo-a em três tópicos principais, que são delineados a seguir:

- **Requisitos Funcionais (RF)** representam a materialização de uma necessidade ou pedido que o software deve atender, englobando as funções e serviços que o sistema pode oferecer ao seu usuário. É crucial que sejam formulados de maneira clara e direta para facilitar a compreensão.
- **Requisitos Não Funcionais (RNF)** delineiam a abordagem do sistema para alcançar o que é proposto pelos RF, não se vinculando diretamente às funcionalidades do sistema, mas, sim, impondo restrições técnicas mensuráveis ao projeto.
- **Regras de Negócio (RN)** consistem em restrições essenciais para viabilizar

o funcionamento do negócio, tão vitais que organizações ou empresas podem operar sem o *software* em si, mas não sem essas regras. Durante o processo de levantamento de requisitos, as RN devem sempre estar associadas a um RF.

3 METODOLOGIA

Este trabalho tem como meta principal criar um aplicativo móvel com capacidade identificar a anemia por meio da coloração da mucosa ocular de caprinos e ovinos, classificando os animais anêmicos como T e os não anêmicos como NT. Este capítulo apresenta uma descrição detalhada do processo de desenvolvimento, com a metodologia do projeto abrangendo quatro etapas essenciais:

- Implementação do modelo de detecção de mucosa ocular de caprinos e ovinos;
- Implementação do modelo de classificação de mucosa ocular de caprinos e ovinos;
- Incorporação dos modelos de detecção e classificação em um aplicativo *iOS*;
- Implementação da interface do aplicativo que apresentará os resultados da detecção/classificação.

3.1 Modelo de Detecção da Mucosa

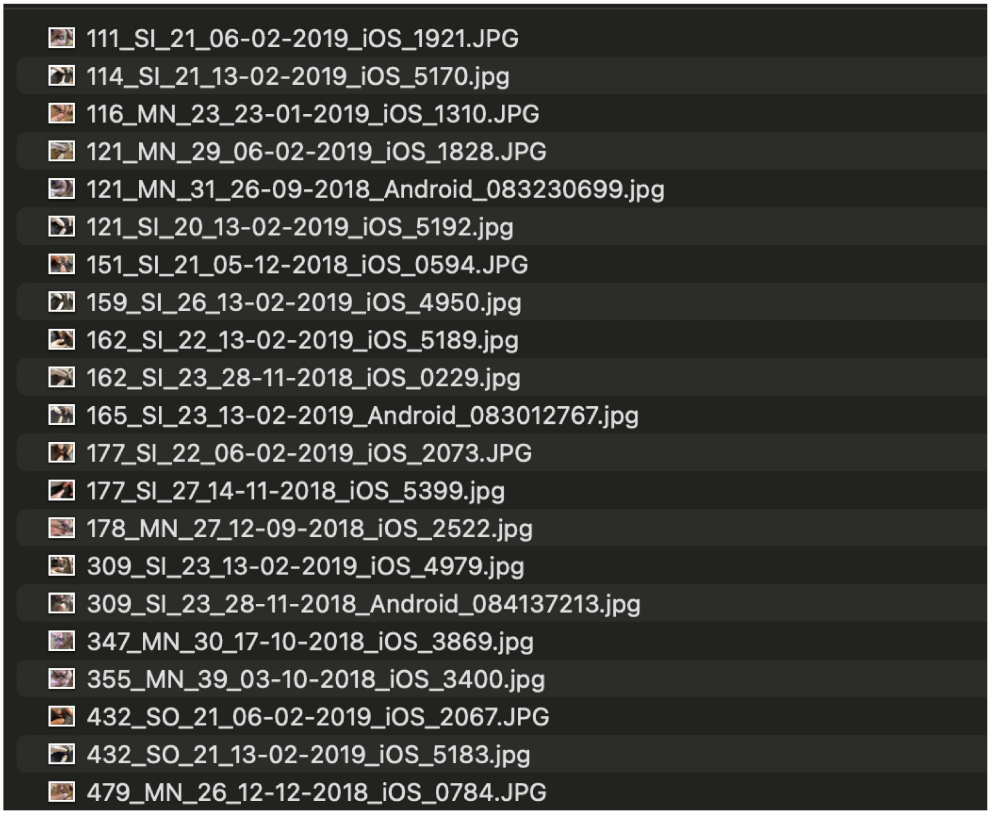
Para desenvolver o modelo de detecção da mucosa ocular, foram utilizadas imagens cedidas por ALMEIDA (2021), as quais foram adquiridas e classificadas pelo próprio autor. Os arquivos de imagem foram nomeados seguindo o padrão: *NUMERO_FOTO-RAÇA-MEDIDA_HEMATÓCRITO-DATA-DISPOSITIVO*, observe a Figura 7 onde temos parte da listagem de imagens. Dessa forma, a partir das imagens fornecidas, foi possível separar o conjunto em duas subclasses utilizando um algoritmo em *Python*, tendo como critério a medida do Ht presente na nomenclatura de cada arquivo.

O banco de dados completo consistia em 200 fotografias de mucosas, sendo que 73 delas apresentavam Ht menor ou igual a 23 (designadas neste estudo como "T"ou doentes), enquanto 127 apresentavam Ht maior ou igual a 24 (classificadas como "NT"ou saudáveis).

Para garantir maior precisão na classificação, foi necessário extrair somente a parte de interesse de cada imagem, ou seja, a mucosa. Para isso, o banco de dados foi rotulado utilizando a ferramenta *RectLabel*, na Figura 8 temos um exemplo desse processo, destacando as mucosas nas imagens para permitir o treinamento de um modelo de detecção.

Com base no banco de dados rotulado, procedeu-se à divisão em dois conjuntos distintos para treinamento e teste. Devido à disparidade na contagem de elementos entre as classes, visando mitigar possíveis vieses no processo de detecção, as imagens foram distribuídas

Figura 7 – Imagens exibindo o padrão de nomeclatura mencionado



Fonte: O próprio autor.

Figura 8 – Imagen rotulada na ferramenta *RectLabel*

Fonte: O próprio autor.

em grupos de treinamento e teste por meio do "método de retenção"ou "*holdout method*". Nessa abordagem, 70% de cada classe foram alocados para o conjunto de treinamento, enquanto 30% foram reservados para o conjunto de teste. Assim, o conjunto de treinamento compreende 90 objetos da classe "Tratar"e 51 da classe "Não Tratar", enquanto o conjunto de teste inclui 38 objetos da classe "Tratar"e 22 da classe "Não Tratar"; esses dados podem ser observados na Tabela 1.

Tabela 1 – Divisão dos dados para treino e teste do modelo de detecção

Classe	**Conjunto de Treinamento**	**Conjunto de Teste**
Tratar	90	38
Não Tratar	51	22

Fonte: O próprio autor.

3.1.1 Treinamento do modelo

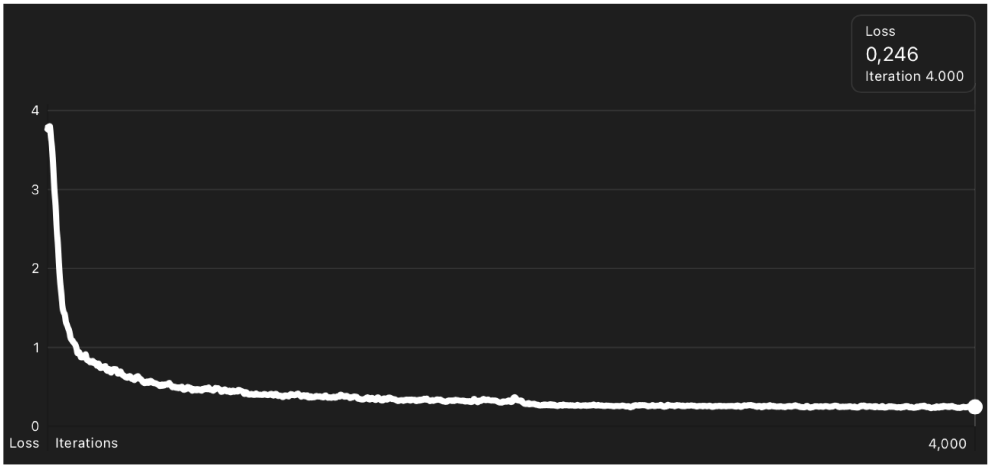
O modelo de detecção foi treinado utilizando a ferramenta *CreateML*. Os dados de validação foram selecionados automaticamente pela ferramenta a partir dos dados de treinamento. O *CreateML* oferece suporte a diversas arquiteturas e técnicas de detecção de objetos, dependendo da versão e das bibliotecas subjacentes disponíveis.

Para o contexto deste trabalho, foi utilizada a versão 5.0 (121.4) da ferramenta. Optou-se por treinar o modelo do zero, embora exista a opção de iniciar a partir de um modelo pré-treinado. A arquitetura escolhida para o treinamento foi a YOLOv2.

O número de iterações foi mantido no padrão da ferramenta, que é 4000; o tamanho do *batch* foi definido como automático; e o tamanho do *grid* foi mantido em 13x13.

As etapas de treinamento incluíram a preparação dos dados, a configuração dos parâmetros de treinamento e a validação do modelo, garantindo que todas as configurações estivessem alinhadas com os objetivos do projeto. A utilização do *CreateML* permitiu uma abordagem eficiente e eficaz no desenvolvimento do modelo de detecção, assegurando resultados precisos e confiáveis. Na Figura 9 temos um gráfico de *Loss/Iteração*, onde "loss"se refere à função de perda (ou função de custo), que é uma medida da disparidade entre a saída predita pelo modelo e a saída real (rótulo) durante o treinamento.

Figura 9 – Log de acompanhamento do treinamento de detecção



Fonte: O próprio autor.

3.2 Modelo de classificação da mucosa

Para garantir maior precisão no treinamento do modelo de classificação, foi utilizado o modelo de detecção previamente treinado. Um *script* em *Swift* foi desenvolvido para percorrer toda a base de dados e extrair a região de interesse das imagens, especificamente a mucosa dos caprinos, resultando em sub-imagens; esse processo pode ser evidenciado na Figura 10, onde é exibido uma imagem original e sua sub-imagem que fora extraída. Com isso, uma nova base de dados foi gerada para o treinamento do modelo de classificação.

Figura 10 – Exemplo de imagem original e sub-imagem extraída através do modelo de detecção

(a) Imagem original (b) Sub-imagem que foi extraída

Fonte: O próprio autor.

Considerando a predominância de imagens de animais saudáveis, e visando evitar a introdução de viés no modelo de classificação, estabeleceu-se como parâmetro limitante o número de itens da classe de animais doentes (classe T), totalizando 73 amostras. Assim, procedeu-se à seleção aleatória de 51 amostras de animais doentes (classe T) e 51 amostras de

animais saudáveis (classe NT) para o grupo de treinamento, enquanto 22 amostras de cada classe foram destinadas ao grupo de teste; esses dados podem ser observados na Tabela 2.

Tabela 2 – Divisão dos dados para treino e teste do modelo de classificação

Classe	**Conjunto de Treinamento**	**Conjunto de Teste**
Tratar	51	22
Não Tratar	51	22

Fonte: O próprio autor.

Dada a limitação no número de amostras disponíveis para treinamento e teste, foram empregados recursos de aumento de dados (*data augmentation*) para explorar possíveis melhorias nos resultados. O aumento de imagens é um processo que envolve a aplicação de transformações, como inversão, corte, redimensionamento, ajuste de brilho, adição de ruído, entre outros. Essas transformações não substituem um conjunto de dados de imagens de qualidade, mas ajudam a maximizar a eficácia do conjunto de dados. Cada aumento tem o potencial de multiplicar o tamanho dos seus dados de treinamento, o que é particularmente útil quando o tamanho da amostra de treinamento é limitado.

Nesse mesmo sentido, foram variadas as iterações até alcançar um resultado satisfatório e observar a convergência do treinamento. O objetivo foi incluir iterações suficientes para obter um modelo preciso, pois interromper o treinamento previamente poderia resultar em um modelo menos preciso.

Além disso, foram utilizados os aumentos de imagens disponíveis no *CreateML*, incluindo inversão e rotação da plataforma.

Como um caminho alternativo à aumentação de dados oferecida pelo *CreateML*, neste trabalho também foi realizada uma aumentação de dados utilizando a biblioteca *ImgAug* da linguagem *Python*. Através dessa biblioteca, foi possível ajustar os seguintes parâmetros:

- Rotação: as imagens foram giradas em um ângulo específico;
- Zoom: as imagens passaram por um processo de aumento de tamanho;
- Brilho: durante essa transformação, houve um ajuste sutil na iluminação de algumas fotografias.
- Deslocamento: nesta etapa, as partes centrais das imagens que continham a mucosa foram deslocadas horizontal e/ou verticalmente após a aplicação de um *zoom*.

No contexto do *CreateML*, é comum que a plataforma automatize muitos dos detalhes

do treinamento, como a escolha de uma arquitetura de rede neural adequada, o ajuste de hiperparâmetros básicos e até mesmo o processo de validação interna. Isso é projetado para simplificar o processo para desenvolvedores, focando mais na facilidade de uso e menos na configuração detalhada de parâmetros. Nesse cenário, apenas estabeleceu-se um período de treinamento para capacitar o modelo de classificação de objetos na ferramenta.

3.3 Desenvolvimento do aplicativo

No processo de desenvolvimento do aplicativo, realizou-se uma análise preliminar do público-alvo, ou seja, dos usuários que interagirão diretamente com a aplicação. Contudo, compreendendo que o objetivo principal deste trabalho é avaliar os modelos de detecção e classificação desenvolvidos exclusivamente com as ferramentas da *Apple*, bem como a interação desses modelos com a aplicação, o foco desta etapa foi direcionado para o desenvolvimento de um MVP, contendo uma arquitetura e uma única interface simples.

Apesar de se tratar de uma aplicação simples, após o processo de definição do que seria o protótipo, foram identificados os Requisitos Funcionais RF, os Requisitos Não Funcionais RNF e as Restrições de Negócio RN do projeto. Cada uma dessas fases é detalhadamente abordada a seguir.

3.3.1 Público alvo

A análise preliminar do público-alvo foi fundamental para garantir que a aplicação atendesse às necessidades e expectativas dos usuários finais. Este estudo incluiu a identificação de perfis demográficos, comportamentos de uso e preferências tecnológicas dos usuários. Com base nessa análise, foi possível delinear características essenciais que a aplicação deveria incorporar para proporcionar uma experiência de usuário satisfatória. Nesse cenário, o aplicativo visa atender predominantemente fazendeiros e pequenos produtores que enfrentam dificuldades no acesso conveniente a exames laboratoriais ou à aplicação do método FAMACHA© e que tenham acesso à *smartphone* com Sistema Operacional (SO) *iOS*.

3.3.2 Ícone da aplicação

No processo de desenvolvimento do aplicativo, um ícone foi criado pelo autor utilizando uma ferramenta de IA chamada *Bing Image Creator*. Essa ferramenta é alimentada

pelo *DALL-E*, desenvolvido pela *OpenAI*, e permite aos usuários criar imagens a partir de descrições em texto. Inicialmente, o ícone foi gerado por meio dessa ferramenta de criação de imagens. Posteriormente, a imagem gerada foi vetorizada utilizando outra ferramenta, chamada *Inkscape*.

A escolha do *Inkscape* para a vetorização do ícone foi estratégica, dado que essa ferramenta oferece flexibilidade e controle preciso na elaboração de elementos gráficos vetoriais. Essas características são cruciais para garantir a qualidade e escalabilidade do ícone em diferentes tamanhos e resoluções. O ícone finalizado está representado na Figura 11.

Figura 11 – Ícone desenhado para o aplicativo

Fonte: O próprio autor.

A criação do ícone seguiu um processo iterativo, começando pela definição dos conceitos visuais desejados, seguidos pela geração de imagens através do *Inkscape*. Este método permitiu a exploração de diversas alternativas visuais, facilitando a seleção de uma imagem base que melhor representasse a identidade do aplicativo. Após a escolha da imagem base, a vetorização no *Inkscape* permitiu refinamentos detalhados e ajustes necessários para garantir a clareza e o impacto visual do ícone em diversas plataformas e dispositivos.

Ademais, a utilização de ferramentas de código aberto não apenas proporcionou um ambiente de desenvolvimento econômico e acessível, mas também garantiu a possibilidade de ajustes futuros e colaboração aberta com outros desenvolvedores e designers. A combinação dessas ferramentas demonstrou ser eficaz na criação de um ícone de alta qualidade, alinhado com os objetivos de design e usabilidade do aplicativo.

Em resumo, a abordagem adotada para a criação do ícone do aplicativo exemplifica

a integração eficiente de ferramentas de IA e *software* de código aberto no processo de *design* gráfico, resultando em um elemento visual que é ao mesmo tempo esteticamente agradável e funcionalmente robusto.

3.3.3 Prototipação das telas

Considerando a delimitação do público-alvo, procedeu-se à elaboração de um protótipo da interface com o intuito de otimizar a adequação às necessidades deste segmento específico. Nesse contexto, buscou-se conferir às telas características de praticidade e simplicidade em sua utilização, visando facilitar a interação por parte dos usuários identificados como alvo.

A aplicação possui uma tela que exibe um *placeholder* que será substituído pela imagem selecionada ou capturada pelo usuário (Figura 16) e uma tela de resultados, onde é possível visualizar a mucosa detectada pelo aplicativo, bem como o resultado obtido após o processamento e classificação da mucosa, que será identificada como saudável T ou não saudável NT.

Dada a simplicidade da interface, as telas foram desenhadas diretamente no *Interface Builder* do *Xcode/UIKit*. Esta ferramenta de construção de *layout*, embora não seja atualmente a mais recomendada para *layouts* complexos, é uma excelente opção para pequenos projetos, pois oferece uma abordagem integrada e eficiente na prototipação, permitindo uma visualização dinâmica e interativa do *design* proposto. A Figura 12 mostra a tela inicial (*Launchscreen*) do aplicativo, que introduz os usuários à interface simplificada.

A escolha do *Interface Builder* do *Xcode/UIKit* se deu por sua capacidade de facilitar o desenvolvimento rápido e a iteração constante durante a fase de prototipagem. Mesmo com suas limitações em projetos de maior complexidade, a ferramenta mostrou-se adequada para atender às demandas de um protótipo voltado para a usabilidade e funcionalidade essenciais. Por exemplo, ao selecionar uma imagem da galeria (Figura 13) ou capturar uma foto usando a câmera (Figura 17), a interface se mantém intuitiva e fácil de usar.

Este *design* final permite uma interação intuitiva e direta, alinhada com os objetivos de simplicidade e eficácia estabelecidos para o público-alvo do aplicativo. A tela de resultados, por exemplo, exibe claramente quando um animal foi identificado como doente (Figura 19) ou saudável (Figura 18).

Em resumo, a abordagem adotada para o desenvolvimento da interface do aplicativo destacou-se pela utilização de ferramentas que promovem a eficiência na prototipação, garantindo

Figura 12 – Launchscreen do aplicativo

CapriScan®

Fonte: O próprio autor.

uma experiência de usuário otimizada e satisfatória para o público-alvo identificado.

3.3.4 Primeira Tela

3.3.4.1 Requisitos Funcionais

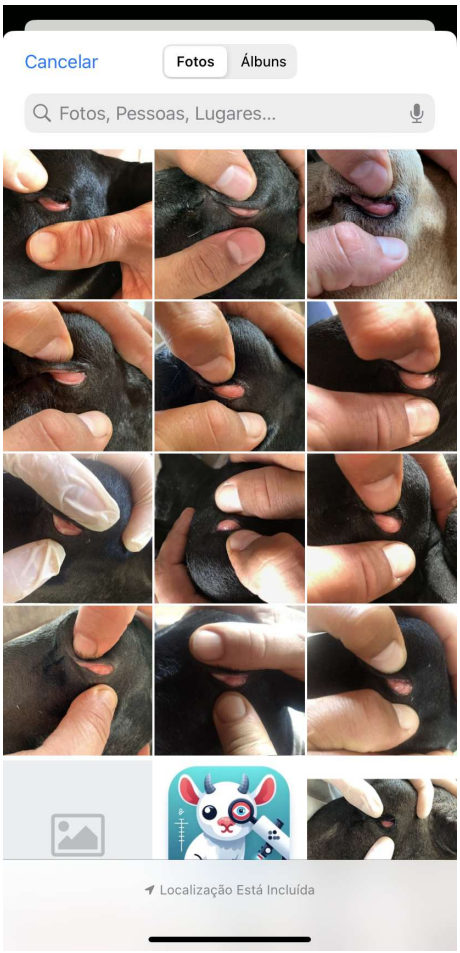
A aplicação deve garantir ao usuário os seguintes processos:

- Tirar fotografias à partir da câmera do celular;
- Alternar a utilização da câmera do celular entre a frontal e a traseira;
- Selecionar uma fotografia previamente salva na galeria do celular;

3.3.4.2 Requisitos Não Funcionais

No desenvolvimento deve ser utilizado os seguintes recursos:

Figura 13 – Galeria para seleção de fotos



Fonte: O próprio autor.

- Utilizar *Swift* como linguagem de desenvolvimento;
- Utilizar a *framework CreateML* para treinamento do modelo de detecção;
- Utilizar a *framework CreateML* para treinamento do modelo de classificação;
- Utilizar a *framework CoreML* para integrar os modelos de classificação e detecção à aplicação;
- Utilizar a *framework UIKit* para construção do layout das telas.
- Utilizar a arquitetura *MVC* para construção da aplicação.

3.3.4.3 Regras de Negócio

Esta tela deve funcionar da seguinte forma:

- Solicitar permissão para leitura e escrita de arquivos no SO *iOS*, caso o *app* não possua;

- Solicitar permissão de acesso à câmera no SO *iOS*, caso o *app* não possua;
- Selecionar uma imagem do celular para análise ou capturar uma a partir da câmera;
- Enviar como parâmetro a imagem a ser analisada e exibir o resultado.

3.3.5 Tela de Resultados

3.3.5.1 Requisitos Funcionais

A aplicação deve garantir ao usuário os seguintes processos na tela de resultados:

- Exibir a imagem analisada com um retângulo ao redor da mucosa ocular detectada;
- Exibir um recorte da mucosa detectada de forma ampliada;
- Exibir a classificação da mucosa como “Tratar” ou “Não tratar”;

3.3.5.2 Requisitos Não Funcionais

No desenvolvimento da tela de resultados deve ser utilizado os seguintes recursos:

- Utilizar *Swift* como linguagem de desenvolvimento;
- Utilizar a *framework CoreML* para integrar o modelo de classificação e detecção à aplicação;
- Utilizar a *framework UIKit* para construção do *layout* das telas;
- Utilizar a arquitetura *MVC* para construção da aplicação;
- Garantir que a interface seja responsiva e se adapte a diferentes tamanhos de tela de dispositivos *iOS*.

3.3.5.3 Regras de Negócio

A tela de resultados deve funcionar da seguinte forma:

- Exibir automaticamente os resultados da análise após a captura ou seleção da imagem;
- Permitir ao usuário visualizar o resultado da análise em formato de texto e imagem;
- Atualizar a exibição dos resultados sempre que uma nova imagem for capturada ou selecionada para análise;

- Garantir que os resultados exibidos sejam claros e precisos para o entendimento do usuário.

Estas especificações são fundamentais para garantir a funcionalidade, a eficiência e a usabilidade da aplicação, assegurando que todos os requisitos técnicos e de negócio sejam devidamente atendidos durante o desenvolvimento.

4 RESULTADOS

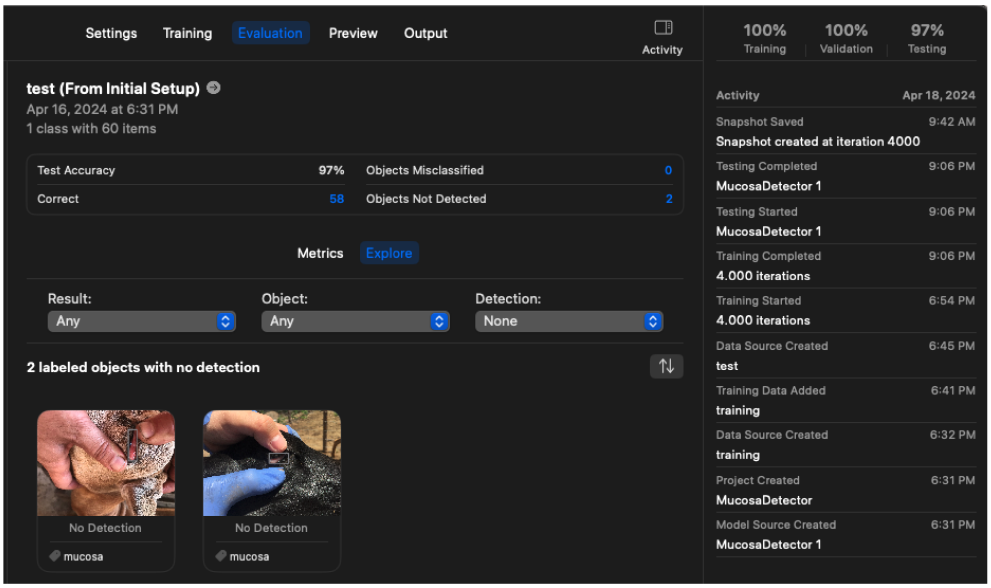
Neste capítulo, serão apresentados os resultados alcançados ao longo do desenvolvimento deste trabalho, abrangendo as métricas de desempenho dos modelos de detecção e classificação, a descrição detalhada do aplicativo desenvolvido, e os resultados obtidos com a integração dos modelos de inteligência artificial na aplicação. Serão discutidos, ainda, os aspectos técnicos e funcionais da implementação, ressaltando as contribuições e desafios enfrentados durante o processo.

4.1 Resultados dos modelos

4.1.1 Resultados do modelo de detecção

Após o treinamento, o modelo de detecção alcançou uma acurácia de 100% tanto na fase de treinamento quanto na fase de validação, e 97% na fase de teste. A Figura 14 apresenta o painel de resultados do *CreateML* e a Tabela 3 exibe um resumo desse resultado.

Figura 14 – Resultado do treinamento do modelo de detecção



Fonte: O próprio autor.

Os resultados obtidos pelo modelo de detecção são extremamente positivos, especialmente considerando a base de dados restrita utilizada. Com as acurácias atingidas em todas as fases, o modelo demonstra uma alta capacidade de generalização e precisão, características essenciais para aplicações práticas em visão computacional. Esses resultados evidenciam a

Tabela 3 – Acurácia do modelo de detecção nas diferentes fases de treinamento

Fase	**Acurácia (%)**
Treinamento	100
Validação	100
Teste	97

Fonte: O próprio autor.

eficiência da arquitetura *YOLOv2*, conhecida por sua habilidade de realizar detecções rápidas e precisas, mesmo com conjuntos de dados limitados.

Além disso, a utilização da ferramenta *CreateML* se mostrou fundamental para alcançar esses resultados. A *CreateML* permitiu, de fato, que, mesmo sem uma formação especializada em IA, fosse possível treinar modelos precisos sem grandes dificuldades. Isso é viabilizado graças à sua interface intuitiva e à abstração dos detalhes de implementação, facilitando o desenvolvimento de soluções eficazes sem a necessidade de um conhecimento profundo dos conceitos de baixo nível da área. Assim, o trabalho demonstra não apenas a qualidade do modelo de detecção, mas também a acessibilidade e o poder das ferramentas modernas de aprendizado de máquina, como o *CreateML*, em democratizar o uso de IA.

4.1.2 Resultados do modelo de classificação

Após o treinamento, o modelo de classificação apresentou uma acurácia de 70% na fase de validação, 84% na fase de teste, e 99% durante o treinamento, conforme realizado pelo *CreateML*. A Figura 15 ilustra a série de treinamentos executados e os resultados obtidos, e a Tabela 4 exibe um resumo desse resultado. Para garantir a compatibilidade com as ferramentas predominantes utilizadas neste trabalho, principalmente pela linguagem *Swift*, optou-se por exportar o modelo no formato *.mlmodel*.

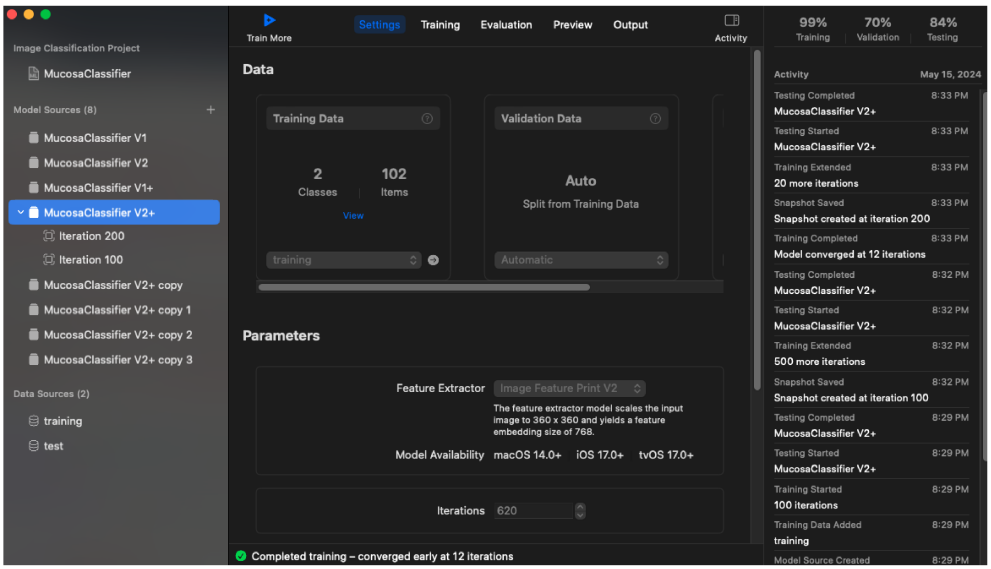
Tabela 4 – Acurácia do modelo de classificação nas diferentes fases de treinamento

Fase	**Acurácia (%)**
Treinamento	99
Validação	70
Teste	84

Fonte: O próprio autor.

A análise dos resultados do modelo de classificação revela alguns pontos importantes a serem considerados. Com uma acurácia de 99% na fase de treinamento, o modelo demonstra

Figura 15 – Imagens na plataforma *CreateML* para treinamento do modelo



Fonte: O próprio autor.

uma alta capacidade de aprender os padrões presentes no conjunto de dados de treinamento. No entanto, a queda na acurácia para 70% na fase de validação e 84% na fase de teste sugere que o modelo pode estar sofrendo de um leve sobreajuste *(overfitting)*, onde ele aprende muito bem os dados de treinamento, mas não generaliza tão bem para dados novos.

Essa diferença nas acurácias entre as fases de treinamento e teste indica que, embora o modelo seja eficaz, há espaço para melhorias, possivelmente por meio do ajuste de hiperparâmetros ou aumento da diversidade do conjunto de dados de treinamento. É importante garantir que o modelo não apenas aprenda os padrões dos dados de treinamento, mas também seja capaz de aplicar esse conhecimento a novos dados, o que é crucial para seu desempenho em aplicações reais.

Esses resultados destacam a necessidade de um balanço cuidadoso entre a complexidade do modelo e a quantidade de dados disponíveis, bem como a importância de utilizar técnicas que previnam o sobreajuste, como a validação cruzada e o uso de mais dados diversificados.

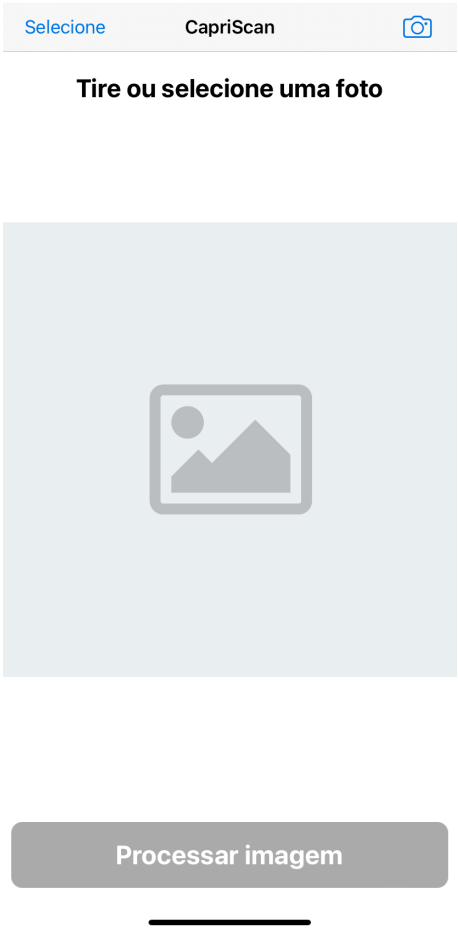
4.2 Resultados do desenvolvimento do aplicativo

O software foi instalado e testado no dispositivo móvel *iPhone 6 Plus*, especificamente escolhido devido à sua idade mais avançada, com o objetivo de avaliar o desempenho do projeto em um ambiente com recursos limitados.

A operação do aplicativo é caracterizada por uma otimização significativa, resultando

em um tempo de inicialização inferior a 3 segundos. Na tela inicial do aplicativo (Figura 16), é possível observar botões para captura e seleção de imagens.

Figura 16 – Tela inicial com placeholder



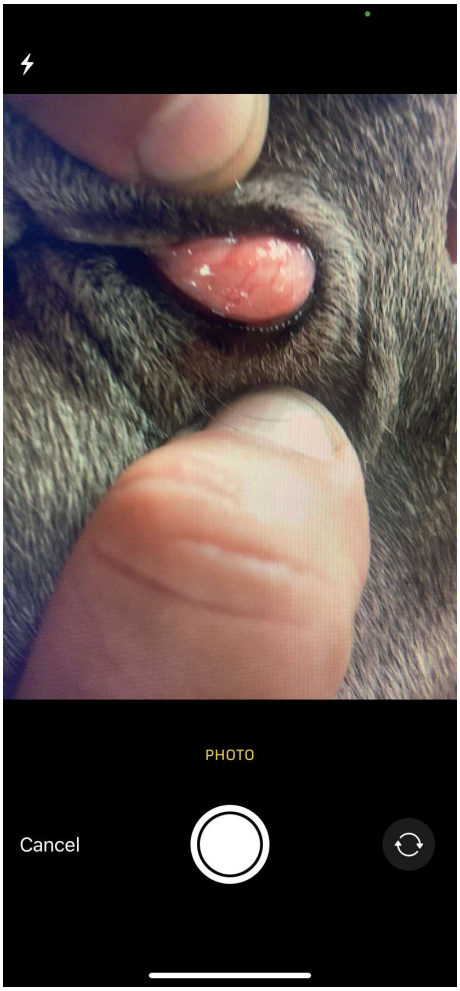
Fonte: O próprio autor.

No caso de seleção do modo de captura, a câmera do dispositivo é acionada, observe a Figura 17.

Após a captura ou seleção de uma imagem, a detecção da mucosa e a classificação de sua coloração predominante ocorrem praticamente de forma instantânea. Nas Figuras 18 e 19, podemos observar a tela de resultados para imagens de um animal saudável e de um animal doente, respectivamente. Este desempenho ágil e eficiente atesta a eficácia do aplicativo, alinhando-se de maneira consistente com seus objetivos estabelecidos.

Esses resultados evidenciam que, mesmo em um dispositivo mais antigo como o *iPhone 6 Plus*, a aplicação mantém um desempenho eficiente, reforçando sua viabilidade e robustez. Esta avaliação em um ambiente de recursos limitados é crucial para garantir que

Figura 17 – Tela de captura de imagem com a câmera do dispositivo



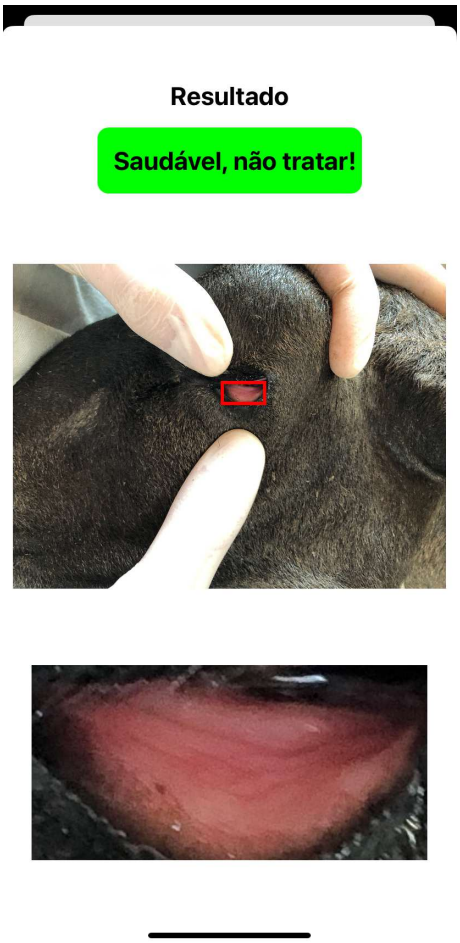
Fonte: O próprio autor.

o *software* possa ser utilizado em uma ampla gama de dispositivos, incluindo aqueles com capacidades de *hardware* mais modestas.

O estudo apresentado revela resultados promissores quanto ao desempenho do aplicativo em um dispositivo móvel *iPhone 6 Plus*, conhecido por suas limitações de *hardware* comparadas a dispositivos mais recentes. O *software* foi projetado para otimizar a operação, resultando em tempos de inicialização rápidos e processamento ágil das operações de detecção e classificação da mucosa.

No entanto, é importante reconhecer algumas limitações inerentes ao estudo. O treinamento do modelo de aprendizado de máquina foi conduzido usando o *framework CreateML* da *Apple* no modo automático, o que limitou a exploração detalhada de métricas de desempenho como número específico de épocas, estratégias de validação cruzada, desvio padrão da acurácia,

Figura 18 – Tela de resultado com um animal saudável



Fonte: O próprio autor.

precisão e *recall*. Essas métricas adicionais são cruciais para uma avaliação completa do modelo em diferentes condições e contextos.

Figura 19 – Tela de resultado com um animal doente

Fonte: O próprio autor.

5 CONCLUSÕES E TRABALHOS FUTUROS

Durante o transcorrer deste estudo, foi delineado o desenvolvimento de um aplicativo com o propósito de identificar animais anêmicos por meio da análise da coloração da mucosa ocular, empregando a linguagem *Swift* e as *frameworks* nativas da *Apple*. A aplicação visa a potencial redução das perdas na produção e a prevenção de óbitos animais, direcionando-se para fazendeiros e produtores.

Ao longo da investigação, constatou-se que o aplicativo exibe um desempenho satisfatório e é acessível mesmo para usuários com pouca familiaridade em métodos de análise específicos da ovinocaprinocultura. Ademais, o modelo implementado para a detecção da mucosa demonstrou uma acurácia de 97%, enquanto o modelo para classificação da coloração da mucosa apresentou uma taxa de acertos de 84%. Todavia, a acurácia do modelo de classificação pode ter sido afetada por limitações no conjunto de imagens, variações na iluminação em algumas fotografias ou, possivelmente, pelo estresse animal, que pode influenciar na coloração da mucosa, resultando em sobreajuste em determinados conjuntos de cores e, consequentemente, na classificação incorreta de amostras.

Assim, os objetivos delineados no início do estudo foram alcançados, e o aplicativo já pode ser considerado um MVP. Contudo, apesar do desempenho satisfatório e da usabilidade simplificada do aplicativo, ainda é necessário realizar testes em campo. Um próximo passo para aprimorar este trabalho seria, portanto, a análise dos resultados do aplicativo em ambiente real, a expansão da base de dados de imagens e a implementação de outros algoritmos visando uma maior precisão na classificação da mucosa ocular. Além disso, recomenda-se realizar uma análise complementar usando outras ferramentas ou técnicas para calcular métricas como precisão e *recall*, o que pode envolver a exportação do modelo para outras plataformas de aprendizado de máquina que ofereçam maior controle sobre o processo de avaliação.

No que tange à aplicação em si, dentre as melhorias para uma próxima versão do trabalho, destaca-se a evolução da aplicação para integrar funcionalidades adicionais através do *Core Data*. Isso incluiria a capacidade de salvar as análises realizadas, adicionar notas e observações para cada análise, e permitir o compartilhamento das análises via *email, WhatsApp* e mensagens diretamente do aplicativo. Essas funcionalidades não só melhorariam a usabilidade e a praticidade da aplicação, mas também proporcionariam um histórico de análises que pode ser útil para o acompanhamento e a tomada de decisões pelos produtores.

Outras possíveis evoluções incluem a integração com plataformas de análise de dados

em tempo real para oferecer *feedback* instantâneo sobre as condições de saúde dos animais, e a implementação de um módulo de treinamento que permita aos usuários aprimorar a precisão do aplicativo com base em novos dados e informações. Adicionalmente, considerar a incorporação de algoritmos de aprendizado profundo poderia melhorar ainda mais a acurácia e a robustez da detecção e classificação da mucosa ocular.

REFERÊNCIAS

ALMEIDA, A. M. A. **Detecção de anemia em ovinos através de aprendizagem profunda em imagens de mucosa ocular**. 113 p. Dissertação (Mestrado em Engenharia Elétrica e de Computação) — Programa de Pós-Graduação em Engenharia Elétrica e de Computação (PPGEEC), Universidade Federal do Ceará - Campus Sobral, Sobral, 2021.

ALMEIDA, M. M. R. **Avaliação de métodos de estimativa da capacidade de carga de fundações diretas em solos não saturados.** 2018. 144 f. Dissertação (Mestrado em Engenharia Civil) — Programa de Pós-Graduação em Engenharia Civil: Geotecnia, Centro de Tecnologia, Universidade Federal do Ceará, Fortaleza, 2018.

AMARANTE, A.; BRICARELLO, P.; ROCHA, R.; GENNARI, S. Resistance of santa ines, suffolk and ile de france lambs to naturally acquired gastrointestinal nematode infections. **Veterinary Parasitology**, v. 120, p. 91–106, 2004.

APPLE. **Create ML - Documentation**. 2024. Apple Developer Documentation. Disponível em: <https://developer.apple.com/documentation/createml>.

AROSEMENA, N.; BEVILÁQUA, C.; MELO, A.; GIRÃO, M. Seasonal variations of gastrointestinal nematode in sheep and goats from semi-arid areas in brazil. **Revue Médicine Véterinaire**, v. 150, p. 873–876, 1999.

BALLARD, D. H. **Computer Vision**. [S.l.]: Prentice Hall, 1982.

CHAGAS, A. C. d. S.; OLIVEIRA, M. C. d. S.; CARVALHO, C. O. d. Método FAMACHA©:um recurso para o controle da verminose em ovinos. **Circular Técnica**, v. 52, p. 1–8, 2007. Disponível em: <https://www.embrapa.br/busca-de-publicacoes/-/publicacao/37734/metodo-famacha-um-recurso-para-o-controle-da-verminose-em-ovinos>. Acesso em: 06 jun. 2021.

CONSELHO NACIONAL DE SECRETÁRIOS DE SAÚDE – CONASS. Guia de contratação de serviços e aquisição de soluções em tecnologia da informação para a gestão estadual do SUS. **CONASS documenta**, Brasília, n. 33, p. 157, 2020. Disponível em: <https://pesquisa.bvsalud.org/portal/resource/pt/biblio-1119867>. Acesso em: 29 jun. 2022.

DAVID, E.; SELFRIDGE, O. Bell telephone laboratories, murray hill, n.j. **Proceedings of the IRE**, v. 50, n. 5, p. 1093–1101, 1962. ISSN 0096-8390. Date of Current Version: 22 January 2007.

DEMOLINER, G.; ALVES, R. J. F. Anemimetro: app móvel para implementação do método famacha. **Unoesc & Ciência**, v. 8, n. 1, p. 25–32, 2017.

Embrapa. **Produtos de origem caprina e ovina: mercado e potencialidades na região do semiárido brasileiro**. 2018. Acessado em: 20 de maio de 2024. Disponível em: <https://www.embrapa.br/cim-inteligencia-e-mercado-de-caprinos-e-ovinos/apresentacao>.

EMBRAPA CAPRINOS E OVINOS. **Método Famacha**. 2024. <https://www.embrapa.br/paratec-controle-integrado-verminoses/vermes/caprinos-ovinos/famacha>. Acesso em: 20 de maio de 2024.

FONTANA, **Introdução aos Algoritmos de Aprendizagem Supervisionada**. 2020.

FOOTE, B.; YODER, J. Big ball of mud. In: **Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)**. [S.l.]: ACM, 1997. Conference paper.

GELDARD, M.; MANNING, J.; BUTTFIELD-ADDISON, P.; NUGENT, T. **Practical Artificial Intelligence with Swift**. [S.l.]: O'Reilly Media, Inc., 2019.

GERÓNIMO, D.; LÓPEZ, A. M.; SAPPA, A. D.; GRAF, T. Survey of pedestrian detection for advanced driver assistance systems. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 32, n. 7, p. 1239–1258, 2010.

GIGLIOTI, C.; GIGLIOTI, R.; SCHIAVONE, D.; CARVALHO, C.; FREITAS, A.; CHAGAS, A.; ESTEVES, S.; OLIVEIRA, M. Epidemiologia das helmintoses gastrintestinais de ovinos criados na região de são carlos-sp. In: **Simpósio de Iniciação Científica da Embrapa Pecuária Sudeste**. [S.l.: s.n.], 2006. p. 43.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016.

IBM. **O que é Computer Vision?** 2024. Acessado: 21 de maio de 2024. Disponível em: <https://www.ibm.com/br-pt/topics/computer-vision>.

INC., A. **Core Data**. 2024. Apple Developer Documentation. Persist or cache data on a single device, or sync data to multiple devices with CloudKit. Disponível em: <https://developer.apple.com/documentation/coredata/>.

KAPLAN, R.; BURKE, J. M.; TERRILL, T. H.; MILLER, J. E.; GETZ, W. R.; MOBINI, S.; VALENCIA, E.; WILLIAMS, M. J.; WILLIAMSON, L. H.; LARSEN, M.; VATTA, A. Validation of the famacha© eye colour chart for detecting clinical anaemia in sheep and goats on farms in the southern united states. **Veterinary Parasitology**, Elsevier, Amsterdam, v. 123, p. 105–120, 2004.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2012. p. 1097–1105.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. **Universidade Federal de Pernambuco, Centro de Informática**, Recife, Pernambuco, Brasil, 2024.

MALAN, F. S.; WYK, J. A. V.; WESSELS, C. D. Clinical evaluation of anaemia in sheep: early trials. **Onderstepoort Journal Veterinary Research**, v. 68, n. 3, p. 165–174, 2001.

MICROSOFT. **Algoritmos de aprendizado de máquina**. 2023. Accessed: 2023-7-21. Disponível em: <https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-are-machine-learning-algorithms>.

Microsoft. **O que é o aprendizado de máquina?** 2024. Acessado: 21 de maio de 2024. Disponível em: <https://learn.microsoft.com/pt-br/azure/cloud-adoption-framework/innovate/best-practices/machine-learning>.

MINSKY, M. **A Framework for Representing Knowledge**. [S.l.], 1974.

MOLENTO, M. B. Opções de tratamento e risco de resistência. **DBO Rural**, v. 23, n. 288, p. 18–22, 2004.

Oracle. **O que é Machine Learning?** 2024. Acessado: 21 de maio de 2024. Disponível em: <https://www.oracle.com/br/artificial-intelligence/machine-learning/what-is-machine-learning/#:~:text=%C3%A9%20Machine%20Learning%3F-,Defini%C3%A7%C3%A3o%20de%20machine%20learning,base%20nos%20dados%20que%20consomem>.

PRACIANO, F. W. S. **Desenvolvimento de um aplicativo baseado em processamento de imagens para detecção de anemia em animais**. Sobral-CE: [s.n.], 2022.

RAMOS, C.; BELLATO, V.; SOUZA, A.; ÁVILA, V.; COUTINHO, G.; DALAGNOL, C. Epidemiologia das helmintoses gastrintestinais de ovinos no planalto catarinense. **Ciência Rural**, v. 34, p. 1889–1895, 2004.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 779–788.

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

RICH, E.; KNIGHT, K. **Artificial Intelligence**. 2nd. ed. [S.l.]: McGraw-Hill, 1991.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, IBM, v. 3, n. 3, p. 210–229, 1959.

SICHMAN, J. S. Inteligência artificial e sociedade: avanços e riscos. **Estudos Avançados**, Universidade de São Paulo, v. 35, n. 101, p. 171–192, 2021. Disponível em: <https://www.revistas.usp.br/eav/article/view/185024/171209>.

TEAM, T. S. **About Swift**. 2024. Accessed: 2024-05-21. Disponível em: <https://www.swift.org/about/history/about>.

TOMAZ, R. S. Uso de métodos de inteligência artificial na avaliação animal. **UNESP**, 2022. Universidade Estadual Paulista (UNESP).

VIEIRA, L. S.; BERNE, M. E. A.; CAVALCANTE, A. C. R. Eficácia antihelmíntica em nematódeos gastrintestinais de caprinos e ovinos. In: **Anais do 6º Seminário Brasileiro de Parasitologia Veterinária**. Bagé: Colégio Brasileiro de Parasitologia Veterinária, 1989. p. 56.

VISIONARY Research. <http://www.scientificamerican.com/article.cfm?id=visionary-research>. Accessed: 2010-09-18.

WINSTON, P. **The Psychology of Computer Vision**. [S.l.]: McGraw Hill, 1975.

WU, X.; SAHOO, D.; HOI, S. C. Recent advances in deep learning for object detection. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 31, n. 12, p. 4684–4703, 2020.

ZHAO, Z.-Q.; ZHENG, P.; XU, S.-t.; WU, X. Object detection with deep learning: A review. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 30, n. 11, p. 3212–3232, 2019.