
**BRUNA CAMPOS AMARAL**

# UM NOVO ALGORITMO DE RASTREAMENTO EM TEMPO REAL PARA SUÍNOS BASEADO EM *DEEP LEARNING*

**LAVRAS - MG**
**2023**

**BRUNA CAMPOS AMARAL**

# UM NOVO ALGORITMO DE RASTREAMENTO EM TEMPO REAL PARA SUÍNOS BASEADO EM *DEEP LEARNING*

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia Agrícola, área de concentração em Construções, Ambiência e Tratamento de Resíduos, para obtenção do título de Mestre.

Prof. Dr. Alessandro Torres Campos
Orientador

Profa. Dra. Angela Green-Miller - University of Illinois
Coorientadora

**LAVRAS – MG**
**2023**

**Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).**

Amaral, Bruna Campos.
Um novo algoritmo de rastreamento em tempo real para suínos baseado em *deep learning* / Bruna Campos Amaral. - 2023.
52 p.

Orientador(a): Alessandro Torres Campos.
Coorientador(a): Angela Green Miller.
Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2023.
Bibliografia.

1. Inteligência Artificial. 2. Rastreamento. 3. Suínos. I. Campos, Alessandro Torres. II. Miller, Angela Green. III. Título.

**BRUNA CAMPOS AMARAL**

# UM NOVO ALGORITMO DE RASTREAMENTO EM TEMPO REAL PARA SUÍNOS BASEADO EM *DEEP LEARNING*

# A NOVEL REAL-TIME TRACKING ALGORITHM FOR PIGS BASED ON DEEP LEARNING

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia Agrícola, área de concentração em Construções, Ambiência e Tratamento de Resíduos, para obtenção do título de Mestre em Ciências.

APROVADA em 30 de junho de 2023

Dr. Alessandro Torres Campos - UFLA
Dr. Tadayuki Yanagi Junior - UFLA
Dr. Elcio Silverio Klosowski - UNIOESTE

gov.br
Documento assinado digitalmente
ALESSANDRO TORRES CAMPOS
Data: 06/07/2023 15:06:09-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Alessandro Torres Campos
Orientador

Profa. Dra. Angela Green-Miller - University of Illinois
Coorientadora

**LAVRAS – MG**
**2023**

## AGRADECIMENTO

Agradeço a Universidade Federal de Lavras (UFLA), pela oportunidade de realização do curso com ensino de excelência e pela contribuição à minha formação humana e profissional.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

**RESUMO**

Garantir a saúde e o bem-estar dos suínos requer um esforço significativo em termos de mão de obra, recursos materiais e tempo. O uso de métodos tradicionais de monitoramento pode ser estressante para os suínos e demandar recursos substanciais dos produtores, especialmente em sistemas de produção em larga escala, como a suinocultura industrial. Essa prática pode ter impactos negativos na saúde e bem-estar dos suínos e na lucratividade econômica da produção sunícula. Neste contexto, o objetivo deste estudo foi desenvolver um modelo usando a arquitetura YOLOv8 para detectar e rastrear suínos em um ambiente de alojamento em grupo. Foram utilizadas 690 imagens de suínos alojados em grupos de nove indivíduos. O conjunto de dados foi dividido em treinamento e validação em uma proporção de 80:20. Com o modelo desenvolvido, foi possível realizar o rastreamento e estabelecer a identificação individual de cada suíno. No entanto, as métricas utilizadas para avaliar o desempenho do modelo apresentaram resultados pouco satisfatórios, evidenciando a necessidade de aumentar o conjunto de dados de treinamento. Apesar desses desafios, o modelo demonstrou bom desempenho em termos de quadros por segundo (FPS), indicando sua viabilidade para aplicações em tempo real, garantindo que o modelo tem potencial para implementação prática no monitoramento e manejo de suínos.

**Palavras-chave:** YOLOv8. Inteligência Artificial. Bem-estar. Rastreamento. Treinamento.

## ABSTRACT

Ensuring the health and well-being of pigs requires significant effort in terms of labor, material resources, and time. The use of traditional monitoring methods can be stressful for pigs and demand substantial resources from producers, especially in large-scale production systems like industrial pig farming. This practice can have negative impacts on pig health and welfare, as well as the economic profitability of pig production. In this context, the aim of this study was to develop a model using the YOLOv8 architecture to detect and track pigs in a group housing environment. A total of 690 images of pigs housed in groups of nine individuals were used, with the dataset divided into training and validation sets in an 80:20 ratio. With the developed model, it was possible to perform tracking and establish individual identification for each pig. However, the metrics used to evaluate the model's performance yielded unsatisfactory results, highlighting the need to increase the training dataset. Despite these challenges, the model demonstrated good performance in terms of frames per second (FPS), indicating its viability for real-time applications, thus ensuring that the model has potential for practical implementation in pig monitoring and management.



**Keywords**: YOLOv8. Artificial Intelligence. Well-being. Tracking. Training.

**SUMÁRIO**

## 1. INTRODUÇÃO

O consumo de carne suína tem aumentado consistentemente ao longo dos anos, representando mais de um terço do consumo global de carne (OECD, 2021; LEBRET et al., 2022). Além disso, à medida que a economia global se eleva, as pessoas se preocupam mais com a procedência do alimento em relação à saúde e o bem-estar dos suínos (LI *et al.*, 2019; QIAO *et al.*, 2021).

Em particular, a saúde animal, o bem-estar e a adoção de práticas agrícolas inteligentes, tornaram-se temas cada vez mais relevantes tanto para os consumidores quanto para os produtores, sendo que, estas preocupações ganharam ainda mais relevância nos últimos anos, em decorrência dos impactos da pandemia do COVID-19 (MARCHANT-FORDE; BOYLE, 2020; GRANDIN, 2021).

A análise da saúde e bem-estar dos suínos, pode ser feita avaliando variáveis como o tamanho corporal, peso, desempenho comportamental ou reconhecimento sonoro (RACEWICZ *et al.*, 2021). Geralmente, esses fatores eram julgados pelos produtores com base nas observações e em suas próprias experiências. Entretanto, a observação humana para ponderar o desempenho comportamental, não é capaz de realizar o monitoramento em tempo real (GARCIA *et al.*, 2020).

Visando criar um sistema eficiente e não invasivo, novas tecnologias, como o rastreamento dos animais utilizando a visão computacional têm sido amplamente utilizada na detecção e rastreamento de suínos, demonstrando desempenho favorável em aplicações de suinocultura, devido à sua forte capacidade automática de extração de características (VAN DER ZANDE *et al.*, 2021; JUNG *et al.*, 2021; DING *et al.*, 2022; WANG *et al.*, 2022).

As redes neurais convolucionais (CNNs) desempenham um papel fundamental no campo da visão computacional, sendo amplamente utilizadas para tarefas de detecção e rastreamento (ZHENG *et al.*, 2018; YANG *et al.*, 2018; FUKUI *et al.*, 2019; GAO *et al.*, 2019; ZHOU *et al.* 2022; BHUJEL *et al.*, 2022). Essas redes são especialmente projetadas para extrair características importantes de imagens, permitindo a identificação e localização de objetos de interesse com precisão. Dentre as arquiteturas de CNN, destaca-se o modelo YOLOv8 (*You Only Look Once version* 8) (ULTRALYTICS, 2023).

O YOLOv8 é a mais recente versão da família de modelos YOLO, conhecidos por suas habilidades combinadas de detecção e segmentação. Com um mapa de recursos ampliado e melhorias na rede convolucional, o YOLOv8 oferece maior precisão e velocidade em comparação com versões anteriores (ULTRALYTICS, 2023). Apesar de ser uma versão recente, o YOLOv8 tem sido estudado e avaliado para detecção e rastreamento em várias áreas (TERVEN *et al.*, 2023; ABOAH *et al.*, 2023; LI *et al.*, 2023; ANG *et al.*, 2023; KANG *et al.*, 2023; DUMITRIU *et al.*, 2023; LOU *et al.*, 2023; KIM *et al.*, 2023; WEI *et al.*, 2023). No entanto, seu potencial na detecção de suínos ainda não foi explorado.

Para realizar o rastreamento, é necessário que haja uma comunicação entre pesquisador e computador. Para isso, existem diversas linguagens de programação disponíveis, dentre as quais se destaca a linguagem de programação *Python*, muito difundida e conhecida por ser fácil de aprender e aplicar, estando no topo das linguagens mais usadas (PIATETSKY, 2019). Muitos estudos têm adotado a linguagem *Python* para realizar o rastreamento e identificação de animais, obtendo ótimos resultados (MAGHSOUDI *et al.*, 2019; SHENK *et al.*, 2021; DE ALMEIDA *et al.*, 2022; QU *et al.*, 2022).

Diante do exposto, objetiva-se com o presente trabalho, desenvolver um modelo computacional baseado na arquitetura YOLOv8, com a capacidade de identificar e rastrear suínos na fase de terminação alojados em baia coletiva.

# 2. REFERENCIAL TEÓRICO

No presente referencial teórico, são abordadas as temáticas referentes à produção suína, e o uso da inteligência artificial na detecção e rastreamento de suínos, utilizando algoritmos e técnicas avançadas para automatizar o monitoramento e controle desses animais. Além disso, são detalhadas as redes neurais convolucionais, que são modelos de aprendizado profundo amplamente utilizados em visão computacional para tarefas como reconhecimento de objetos e segmentação de imagens. Serão apresentados os aspectos relacionados ao YOLOv8 utilizado para detecção de objetos em tempo real baseado em redes neurais convolucionais. Será abordado também a linguagem de programação Python, amplamente utilizada na área de ciência de dados e inteligência artificial, proporcionando uma ampla gama de bibliotecas e ferramentas para desenvolvimento e análise de dados.

## 2.1 Suinocultura

A produção de suínos exerce papel fundamental na economia de muitos países, gerando empregos e contribuindo para o produto interno bruto (PIB). Onde nos Estados Unidos, a indústria suína contribuiu com aproximadamente US$ 24,5 bilhões para o PIB daquele país, em 2020 (NPPC, 2021). Da mesma forma, a produção suína desempenha um papel importante na economia de outros países como China, Brasil, Alemanha, Espanha e outros (MASSOLA, 2017).

Globalmente, a carne suína é a segunda em preferência para consumo devido aos seus baixos preços relativos (USDA, 2017; OECD-FAO, 2019). Em números totais, a produção de carne suína deverá crescer aproximadamente 11 Mt (ou seja, 9,3%) até 2028 (OECD-FAO, 2019). A demanda por produtos suínos tem apresentado um crescimento significativo nos últimos anos. A expansão econômica em países em desenvolvimento, como China, tem impulsionado o aumento do consumo de carne suína. Por exemplo, a China é o maior consumidor e produtor de carne suína do mundo, e a demanda interna continua a crescer devido ao aumento da renda e mudanças nos padrões de consumo alimentar (USDA, 2021).

À medida que a demanda por carne continua a aumentar, as pessoas se preocupam mais com a procedência do alimento em relação à saúde e o bem-estar dos suínos (LI *et al.*, 2019; QIAO

*et al.*, 2021). A preocupação com o bem-estar animal e a segurança alimentar tem se tornado cada vez mais presente entre os consumidores (EUROPEAN COURT OF AUDITORS, 2018). Dentro deste cenário, o monitoramento e rastreamento podem ser inseridos como alternativas para o estudo de comportamento de um rebanho suíno, podendo levar a importantes conclusões a respeito do impacto comportamental e sobre a qualidade e produtividade dos animais.

Diante disso, faz-se necessário o uso de tecnologias e sistemas de informação que permitam conhecer melhor o comportamento dos animais, a fim de garantir o bem-estar destes, permitido conhecer a localização e deslocamento de um suíno em sua baia ou em uma área determinada, assim, aumentando em consequência a qualidade e quantidade dos produtos de origem suína (SANTOS *et al.*, 2016).

Desta forma, Santos *et al.* (2016) indicam que as tecnologias de suinocultura de precisão, podem auxiliar aos gestores ou produtores na identificação e no estudo do comportamento dos animais e, com isso, melhorar o sistema produtivo de maneira a aperfeiçoar e aumentar a produtividade do rebanho. Pois, sistemas automatizados de monitoramento em tempo real são de grande relevância para investigar e controlar o comportamento alimentar, estimativa de peso de suínos em crescimento e terminação entre outros (NIR *et al.*, 2018).

## 2.2 Inteligência artificial (IA) na detecção e rastreamento de suínos

Nos últimos anos, a Inteligência artificial (IA) tem se destacado no rastreamento visual em diversas áreas, incluindo navegação, robótica e controle de tráfego (LI *et al.*, 2018). No âmbito da criação de suínos, têm sido propostas diferentes abordagens de visão por IA para permitir o monitoramento e rastreamento em tempo real dos animais. Nesse processo, a IA é treinada para identificar e classificar diferentes objetos em imagens, como os próprios suínos, equipamentos e obstáculos na baia. Essas técnicas possibilitam a obtenção de informações importantes sobre a trajetória de movimento e o tempo de cada animal, o que pode ser útil para avaliar o estado de saúde dos animais, melhorar o bem-estar dos mesmos e evitar perdas econômicas na produção (CHEN *et al.*, 2020; ZHANG *et al.*, 2020).

Uma das principais vantagens dos sistemas automatizados de monitoramento é a coleta contínua de dados, sem perturbar os animais e minimizando fontes de parcialidade, como a

subjetividade do observador. Por operarem em tempo real, esses sistemas são capazes de identificar mudanças comportamentais imediatamente e, portanto, são ferramentas valiosas para detectar inadequações na gestão, agitação dos animais e doenças (PANDEY *et al.*, 2021).

Para garantir um bom desempenho no rastreamento de objetos, é fundamental considerar fatores como o processamento de dados, a seleção adequada de recursos e o modelo de detecção apropriado (KWON *et al.*, 2017; MUNAPPY *et al.*, 2019; GUO *et al.*, 2022). Para garantir a consistência, integridade e qualidade dos dados utilizados para treinar e alimentar os modelos de IA de rastreamento, é necessário adotar medidas que garantam que as informações sejam precisas e confiáveis. Isso inclui a utilização de metodologias de coleta de dados confiáveis e padronizadas, bem como, a validação regular desses dados para detectar e corrigir possíveis erros ou inconsistências (MUNAPPY *et al.*, 2019)

Infelizmente, é comum que os dados coletados apresentem desafios, como oclusões e desfoque de movimento, decorrentes de objetos em rápida movimentação. Esses problemas podem prejudicar a precisão dos resultados obtidos pelo modelo de IA, destacando a importância de se adotar medidas para mitigá-los e obter dados mais confiáveis (MENG; YANG, 2019).

É importante destacar, que a visão baseada em IA para detecção e rastreamento de suínos é uma tecnologia em constante evolução. Com o avanço das técnicas de *deep learning* e a utilização de dados em tempo real, é possível obter resultados cada vez mais precisos e confiáveis, contribuindo para o bem-estar dos animais e a eficiência da produção animal. Uma das principais técnicas utilizadas nesse contexto é a aplicação de Redes Neurais Convolucionais (CNNs - *Convolutional Neural Networks*), que são capazes de extrair características relevantes das imagens por meio de filtros de convolução.

## 2.3 Redes neurais convolucionais

As redes neurais convolucionais (CNNs) são amplamente reconhecidas como uma das abordagens mais comuns no campo do aprendizado profundo, especialmente quando se trata de dados de alta dimensão, como imagens e vídeos. As CNNs são frequentemente empregadas para tarefas como classificação de imagens e detecção de objetos (HAKIM *et al.*, 2022; LI *et al.*, 2022; CHAKRAVARTULA *et al.*, 2022; ROSLIN *et al.*, 2023; DUDEKULA *et al.*, 2023).

O poder das CNNs reside na utilização de camadas convolucionais, um tipo especial de camada que processa as informações de maneira localizada (HAN *et al.*, 2018; LIANG *et al.*, 2018). Essas camadas aplicam filtros convolucionais para extrair características relevantes das imagens, como bordas, texturas e padrões visuais distintos. A arquitetura das CNNs é projetada para aproveitar a estrutura espacial dos dados, permitindo uma análise eficiente de regiões específicas das imagens.

Nas pesquisas recentes, as CNNs têm sido aplicadas de diversas formas na suinocultura (XU *et al.*, 2022; MA *et al.*, 2023; SRINIVAS *et al.*, 2023; LI *et al.*, 2023) . Um exemplo é a utilização de CNNs para a fusão de imagens visíveis e infravermelhas, visando à detecção de características específicas do corpo dos suínos (ZHONG *et al.*, 2022). Outra aplicação é a fusão de recursos acústicos e informações profundas para o reconhecimento de sons de tosse de suínos, permitindo identificar padrões sonoros relacionados a doenças ou desconforto nos animais (SHEN *et al.*, 2022). Essas aplicações demonstram o potencial das CNNs na suinocultura, contribuindo para a melhoria da saúde, bem-estar e eficiência na produção desses animais.

As CNNs possuem diversos tipos e arquiteturas, cada um com suas características e aplicações específicas. Entre os modelos de CNN mais conhecidos e utilizados, destacam-se a LeNet-5, AlexNet, VGGNet, GoogLeNet (*Inception*), ResNet e YOLO (*You Only Look Once*). Cada uma dessas arquiteturas possui características distintas, como o número de camadas, o tamanho dos filtros convolucionais e a utilização de técnicas como pooling e normalização. Dentre esses modelos, o YOLO tem se destacado pela sua eficiência e desempenho na detecção em tempo real de objetos (LIANG *et al.*, 2022; LEE *et al*. 2022, ; ZHAO *et al.*, 2022; WANG *et al.*, 2022; KARAMAN *et al.*, 2023).

## 2.4 YOLOv8

O YOLO é um algoritmo de detecção de objetos que se destaca pela sua eficiência e velocidade. Ao contrário de outros métodos que exigem várias passagens pela imagem, o YOLO realiza a detecção em uma única etapa, tornando-o adequado para aplicações em tempo real. Ele divide a imagem em uma grade e atribui caixas delimitadoras e pontuações de confiança para cada objeto encontrado. Com seu desempenho rápido e capacidade de detecção precisa, o YOLO tem

sido amplamente utilizado em diversas áreas (DIWAN *et al.*, 2023; CHEN *et al.*, 2023; ROY *et al.*, 2023; KARAMAN *et al.*, 2023; SOUZA *et al.*, 2023).

O YOLO original e suas evoluções, têm sido explorados na suinocultura em diferentes contextos, apresentando bons resultados. Witte *et al.* (2022) criaram um modelo de classificação de postura de suínos usando o YOLOv5. Visando a detecção de suínos, Yin *et al.* (2021) utilizaram a versão YOLOv3, esta mesma versão foi utilizada por Bo *et al.* (2020) na identificação de cabeça e cauda dos suínos. Já a versão YOLOv4 foi utilizada para detecção de suínos, reconhecimento de postura, comportamento alimentar de leitões, detecção de cabeça e cauda, além de estimar o nível de dano da área de pastagem na produção de suínos ao ar livre (LI *et al.*, 2021; KIM *et al.*, 2021; OCEPEK *et al.*, 2021; OH *et al.*, 2023; BIN *et al.*, 2023).

Kosonen (2023) utilizou YOLOv5, YOLOv7 e YOLOv8 para deteção de tumores cerebrais, com o intuito de saber qual das CNNs apresentaria melhor desempenho. Este autor verificou que o YOLOv8, o modelo YOLO mais atualizado, apresentou os melhores resultados, com valor de mAP de 87%. Hanif *et al.* (2023) com a mesma comparação das três versões do YOLO, aplicado a um sistema de auxílio para cegos, analisando a detecção de objetos e determinação da localização, concluíram que o YOLOv8 apresentou os melhores resultados em todas as métricas avaliadas.

Khalid (2023) comparou 6 versões diferentes do YOLO (YOLOv3, YOLOv3-TINY, YOLOv4, YOLOv4-TINY, YOLOv6 E YOLOV8) na detecção precoce de pragas em culturas de campo. YOLOv8 foi considerado o melhor modelo para detecção de pragas entre os seis modelos baseados em YOLO, com maior valor de precisão média (84,7%).

Dornadula *et al.* (2023) compararam o YOLOv5 com o YOLOv8 para detecção de sombras e descobriram que o YOLOv8 apresentou melhores resultados em todas as métricas analisadas, além disso, enquanto o YOLOv5 apresentou precisão de 15,6%, o YOLOv8 chegou a uma precisão de 82,6%, resultando no aumento de precisão de 67%. Ruiz-Ponce *et al.* (2023) também comparando as duas CNNs, observaram que o YOLOv8 mostrou-se superior ao YOLOv5.

## 2.5 Linguagem de programação *Python*

O principal objetivo da visão computacional é fornecer aos computadores o tipo de capacidade de funcionalidade do cérebro do homem. Teoricamente, ela alude ao controle lógico

que estuda como separar dados de imagens em estruturas artificiais (ABIODUN *et al.*, 2018). Os autores acrescentam, que os subdomínos da visão computacional incluem detecção e reconhecimento de objetos, estimativa de objetos, posição de objetos, detecção de eventos, reconstrução de cenas, restauração de imagens, edição de imagens, aprimoramento de vídeo e aprendizado estatístico.

Nos últimos anos, a linguagem de programação *Python* ganhou atenção excepcional graças à sua simplicidade e flexibilidade (GOJKOVIĆ *et al.*, 2020). *Python* é a linguagem de fato para aprendizado de máquina e programação de ciência de dados (SHENK *et al.*, 2021). O uso da linguagem de programação *Python* fornece acesso a uma vasta gama de ferramentas e pacotes, que aumentam a eficiência e eficácia das análises (KULIKOV *et al.*, 2023). Muitos projetos de código aberto fornecem implementações de algoritmos de visão computacional de última geração (ARNOLD; TILTON, 2020).

Com as técnicas de processamento de imagem baseadas em *Python*, as imagens são convertidas em um formato de cor apropriado e as equações dos contornos da imagem podem ser resolvidas (GUPTA *et al.*, 2021). A identificação de objetos e a marcação deles em vídeos pode ajudar um grande número de usuários no processamento e detecção de imagens (MANJU; VALARMATHIE, 2021).

*Python* é uma linguagem de programação que tem uma estrutura de dados de alto nível, eficiente e uma abordagem simples, mas eficaz, para a programação orientada a objetos. A linguagem de alto nível significa que a linguagem de programação *Python* é próxima ou semelhante à linguagem cotidiana, ao contrário da linguagem de máquina, que é uma série de códigos binários (WIDODO *et al.*, 2020). Estes autores indicam que o *Python* tem vantagens em comparação com outras linguagens já existentes, inclusive em termos de facilidade, capacidade, desenvolvimento, interação com os usuários e paradigma de programação.

O uso da linguagem *Python* está crescendo rapidamente, sendo amplamente conhecida pelos programadores porque tem uma estrutura de escrita curta e simples que é fácil de aprender e entender (ABADI; TAHCFULLOH, 2022). Conforme estes autores, *Python* é de código aberto, onde todos podem criar, adicionar, desenvolver e usar bibliotecas para vários propósitos, sendo utilizado em todos os campos.

Um aspecto interessante é o fato de soluções recentes de frameworks terem escolhido *Python* como principal linguagem de programação. O principal benefício dessa seleção é a fácil interação com o ambiente do sistema operacional (*Linux* ou *Windows*) e o fato de existir uma quantidade considerável de bibliotecas com algoritmos ML, DL ou CV (ORHEI *et al.*, 2021). Além disso, essa linguagem possui uma grande rede de suporte e inúmeros recursos disponíveis para os usuários online (DICKSON-KARN; OROSZ, 2021).

## 3. MATERIAL E MÉTODOS

A pesquisa foi desenvolvido mediante parceria entre o Grupo de Pesquisa em Construções e Ambiência em BiossistemasGrupo de Pesquisa em Ambiência em Biossistemas (COAMBI) e o AWES Lab (*Animal Welfare, Environment, and Sustainability Laboratory*) do *Agricultural and Biolgical Department* da Universidade de Illinois, e as gravações foram realizadas em uma Granja comercial de suínos em Ohio, *US*.

### 3.1 Animais e habitação

No estudo, foram utilizados 54 suínos híbridos comerciais da raça Large White, em fase de crescimento e terminação, os quais foram distribuídos igualmente em 6 baias. Cada baia continha 9 animais e era separada por paredes sólidas, possuindo um piso totalmente ripado medindo 4 metros de comprimento por 2,2 metros de largura. Além disso, cada baia estava equipada com um comedouro duplo de dimensões 1,1 metros por 0,7 metros e um bebedouro. Os suínoss tinham livre acesso à comida e água.

### 3.2 Gravação dos vídeos

As imagens de vídeo foram registradas do dia 22 de novembro a 15 de dezembro de 2022, abrangendo tanto as horas diurnas quanto noturnas, permitindo capturar uma variedade de condições de iluminação. Foram instaladas seis câmeras (Amcrest UltraHD Series 5MP PoE Turret Camera) posicionadas acima das baias (1 câmera para cada baia), proporcionando uma vista superior. As gravações eram realizadas a cada 5 minutos, resultando em 12 vídeos por hora. Consequentemente, a cada dia, eram obtidos 288 vídeos para cada câmera. Essa abordagem permitiu capturar diferentes situações e variações de iluminação ao longo do dia e da noite, fornecendo um conjunto diversificado de dados para o treinamento e análise do modelo.

### 3.3 *Dataset*

No processo de preparação dos dados, foram adotadas etapas de pré-processamento para obter um conjunto de imagens adequado para a anotação e treinamento do modelo de detecção e rastreamento de suínos.

Para viabilizar a anotação, a partir dos vídeos capturados na baia 6, foram gerados frames em intervalos regulares de tempo (1 frame a cada minuto). Para garantir a representatividade dos comportamentos e posições dos suínos, foram selecionadas imagens que abrangiam diversas ações e posições dos suínos, e também diferentes períodos de tempo.

O código usado para extrair os frames (Figura 1), foi aplicado a diversas gravações de 5 minutos. Primeiro, é definido o caminho do vídeo a ser processado, em seguida, o objeto “VideoCapture” é inicializado para abrir o vídeo. O número total de *frames* do vídeo é obtido e o intervalo entre os *frames* é calculado para extrair um total de 5 frames. Um *loop* é iniciado para ler os *frames* do vídeo. A cada iteração, o próximo *frame* é lido e, se for um múltiplo do intervalo definido, é salvo como uma imagem na pasta "data". Os contadores de *frames* são atualizados e é verificado se o número desejado de *frames* foi atingido. Quando os 5 *frames* forem extraídos, o *loop* é encerrado e os recursos são liberados.

Figura 1 – Codigo usado para extração de frames de um vídeo.

```python
import cv2
import os

# Caminho para o vídeo
video_path = r'C:\Users\Windows\Desktop\Mestrado\CARGIL\Vídeos\C6P51\02_12\17\20221202-170000-170500.mp4'

try:
    # Criar pasta para armazenar os frames
    if not os.path.exists('data'):
        os.makedirs('data')
except OSError:
    print('Error: Creating directory of data')

# Inicializar o objeto VideoCapture
cam = cv2.VideoCapture(video_path)

# Obter o número total de frames no vídeo
total_frames = int(cam.get(cv2.CAP_PROP_FRAME_COUNT))

# Calcular o intervalo entre os frames para obter 5 frames no total
interval = total_frames // 5

# Variável para contar os frames
current_frame = 0
frame_count = 0

while True:
    # Ler o próximo frame
    ret, frame = cam.read()

    if ret:
        if current_frame % interval == 0:
            # Salvar o frame como imagem
            name = 'data/frame' + str(frame_count) + '.jpg'
            print('Creating...' + name)
            cv2.imwrite(name, frame)
            frame_count += 1

        # Aumentar o contador de frames
        current_frame += 1

        # Verificar se atingimos o número desejado de frames
        if frame_count == 5:
            break
    else:
        break

# Liberar recursos
cam.release()
```

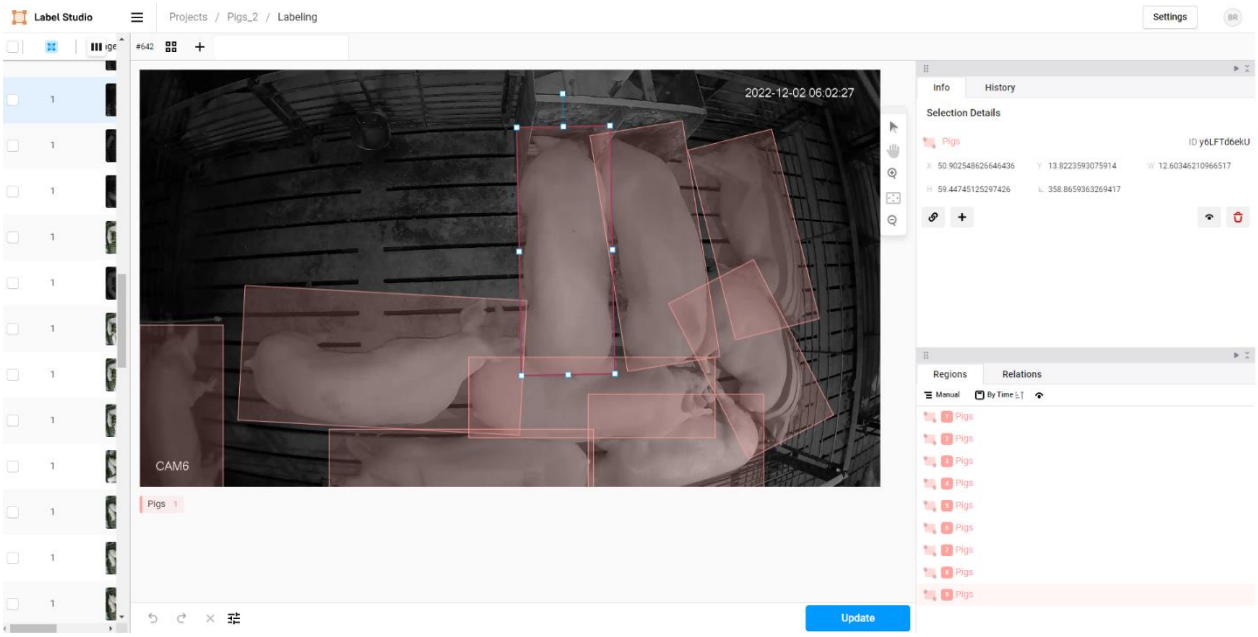
Fonte: Acervo da pesquisa.

Para fins de detecção, os objetos nas imagens devem ser rotulados com as classes apropriadas. Neste trabalho, as imagens tiveram uma única classe: *Pig*. Para rotular as imagens, foi utilizado o *software Label Studio*. Este *software* é uma ferramenta de anotação de dados que oferece uma versão gratuita, tornando-o acessível para diversos projetos de pesquisa e desenvolvimento.

Em cada imagem, os suínos são rotulados conforme a Figura 2. No Total, foram rotuladas 690 imagens e exportadas em formato YOLO. Esse tipo de exportação permite que cada imagem

anotada seja associada a um arquivo de texto no formato YOLO (.txt), que contém informações sobre as coordenadas das caixas delimitadoras e a classe identificada.

Figura 2 - Interface do Label Studio com a imagem dos suínos e suas respectivas caixas delimitadoras.




Fonte: Acervo da pesquisa.

A saída salva de cada imagem é um texto, conforme apresentado na Figura 3. A primeira coluna indica a classe (Pig), neste caso representada pelo valor “0”. A segunda coluna indica a coordenada x do canto superior esquerdo da caixa delimitadora que envolve o suíno, enquanto a terceira coluna apresenta a coordenada y do canto superior esquerdo. A quarta coluna informa a largura da caixa delimitadora, determinando a extensão horizontal do suíno na imagem. Por fim, a última coluna indica a altura da caixa delimitadora, representando a extensão vertical do suíno na imagem. Essas informações permitem a localização e identificação precisa dos suínos nas imagens anotadas.

Figura 3 – Exemplo de arquivo .txt gerado pelo *Label Studio* contendo as informações de anotação, incluindo a classe, coordenadas e dimensões da caixa delimitadora para cada um dos 9 suínos na imagem.

```
0 0.13000000000000003   0.8504858654572941    0.20285714285714293   0.2990282690854116
0 0.08714285714285715   0.468570622323003     0.12571428571428572   0.5104423280423276
0 0.230291662417913     0.6082937354198928    0.45428159955965486   0.4510843043963073
0 0.3249999999999998    0.8873378684807256    0.327142857142857     0.2253242630385482
0 0.6942857142857144    0.8669841269841269    0.31142857142857144   0.26603174603174595
0 0.8521428571428571    0.6920804232804223    0.09285714285714358   0.500283597883597
0 0.7721303726134303    0.3356890451156646    0.25604461625206537   0.4255266455050824
0 0.5977892512337154    0.4292633980144103    0.23803511719784418   0.5635197895485945
0 0.7197372342431876    0.4391743097759263    0.25522517930147204   0.5927017196745393
```

Fonte: Acervo da pesquisa.

### 3.4 Treinamento do modelo

O treinamento da CNN requer uma quantidade significativa de recursos computacionais, em termos de consumo energético. Nesse sentido, é essencial recorrer a GPUs capazes de processar dados de forma paralela, a fim de otimizar o processo de treinamento. Com esse objetivo, foi adotado o ambiente de programação *Google Colaboratory* (Colab) como plataforma de suporte para conduzir o treinamento. Devido à natureza dinâmica do Colab, cada sessão de treinamento é alocada aleatoriamente em uma GPU disponível. Neste trabalho, o modelo foi treinado utilizando a GPU Tesla T4, garantindo uma infraestrutura computacional robusta para viabilizar a execução eficiente do treinamento.

Para treinar e validar o modelo, foram divididas 80% das imagens para treinamento (552 imagens) e 20% para teste (138 imagens). A divisão 80/20 é frequentemente adotada (ZHAO *et al.*, 2020; ALY *et al.*, 2021; JAMTSHO *et al.*, 2021; ZAINUDDIN *et al.*, 2022; LUO *et al.*, 2022), pois oferece uma quantidade significativa de dados de treinamento para permitir que o modelo aprenda padrões relevantes e generalize bem para novos dados. Enquanto a validação é usada para avaliar o desempenho do modelo em dados não vistos durante o treinamento.

Para a divisão das imagens de treinamento e teste, foi utilizado o código apresentado na Figura 4, que é responsável por aleatorizar as listas de nomes de arquivos de imagens e rótulos,

definir o tamanho dos conjuntos de treinamento e teste com base na proporção 80/20, além de criar os diretórios de treinamento e teste. Em seguida, os arquivos de imagens e rótulos são copiados para os respectivos diretórios de treinamento e teste.

Figura 4 – Código para divisão das imagens em conjuntos de treinamento e teste.

```
arquivos_imagens = list(set([nome[:-4] for nome in os.listdir(divisao_caminhos_imagens)]))
arquivos_rotulos = list(set([nome[:-4] for nome in os.listdir(divisao_caminhos_rotulos)]))

print(f"--- Esta pasta de imagens contém um total de {len(arquivos_imagens)} imagens ---")
random.seed(42)
random.shuffle(arquivos_imagens)
random.shuffle(arquivos_rotulos)

tamanho_teste = int(len(arquivos_imagens) * proporcao)
tamanho_treino = len(arquivos_imagens) - tamanho_teste

os.makedirs(caminho_treino_imagens, exist_ok=True)
os.makedirs(caminho_treino_rotulos, exist_ok=True)
os.makedirs(caminho_teste_imagens, exist_ok=True)
os.makedirs(caminho_teste_rotulos, exist_ok=True)

for arquivo in tqdm(arquivos_imagens[:tamanho_treino]):
    if arquivo == 'classes':
        continue
    shutil.copy2(os.path.join(divisao_caminhos_imagens, arquivo + '.jpg'), os.path.join(caminho_treino_imagens, arquivo + '.jpg'))

for arquivo in tqdm(arquivos_rotulos[:tamanho_treino]):
    if arquivo == 'classes':
        continue
    shutil.copy2(os.path.join(divisao_caminhos_rotulos, arquivo + '.txt'), os.path.join(caminho_treino_rotulos, arquivo + '.txt'))

print(f"------ Dados de treinamento criados com uma divisão de 80% contendo {len(arquivos_imagens[:tamanho_treino])} imagens -------")

if caminho_negativo:
    imagens_negativas = list(set([nome[:-4] for nome in os.listdir(caminho_negativo)]))
    for arquivo in tqdm(imagens_negativas):
        shutil.copy2(os.path.join(caminho_negativo, arquivo + ".jpg"), os.path.join(caminho_treino_imagens, arquivo + '.jpg'))

    print(f"------ Total de {len(imagens_negativas)} imagens negativas adicionadas aos dados de treinamento -------")
    print(f"------ Total de dados de treinamento criados com {len(arquivos_imagens[:tamanho_treino]) + len(imagens_negativas)} imagens -------")

for arquivo in tqdm(arquivos_imagens[tamanho_treino:]):
    if arquivo == 'classes':
        continue
    shutil.copy2(os.path.join(divisao_caminhos_imagens, arquivo + '.jpg'), os.path.join(caminho_teste_imagens, arquivo + '.jpg'))

for arquivo in tqdm(arquivos_rotulos[tamanho_treino:]):
    if arquivo == 'classes':
        continue
    shutil.copy2(os.path.join(divisao_caminhos_rotulos, arquivo + '.txt'), os.path.join(caminho_teste_rotulos, arquivo + '.txt'))

print(f"------ Dados de teste criados com um total de {len(arquivos_imagens[tamanho_treino:])} imagens ----------")
print("------ TAREFA CONCLUÍDA -------")
```

Fonte: Acervo da pesquisa.

Foi utilizado o YOLOv8s, que é um modelo de detecção de objetos de última geração, apresentando como um de suas principais vantagens a eficiência em termos de velocidade e precisão. Ele consegue alcançar altas taxas de quadros por segundo (FPS) em tempo real, o que o torna adequado para realizar detecção e rastreamento em tempo real (ULTRALYTICS, 2023).

Para o treinamento do modelo, é importante definir o número adequado de épocas a serem executadas. Cada época representa uma passagem completa do conjunto de dados pelo modelo, permitindo que ele assimile e aprenda com os exemplos fornecidos. Levando em consideração o número reduzido de imagens usadas para o treinamento, optou-se por estabelecer o treinamento com 300 épocas, reconhecendo que esse número é mais do que o necessário para a quantidade reduzida de imagens utilizadas. No entanto, essa abordagem também permite monitorar o valor do "*loss*" ao longo das épocas, possibilitando a análise de possíveis problemas como *overfitting*, em que o modelo se ajusta em excesso aos dados de treinamento. Ao observar o comportamento do "*loss*" durante o treinamento, é possível obter *insights* sobre a capacidade de generalização do modelo e determinar o número ideal de épocas para alcançar um equilíbrio entre desempenho e evitar o *overfitting*.

### 3.5 Detecção dos suínos

Após o processo de treinamento, foi gerado o arquivo "best.pt", que representa o modelo com os melhores resultados obtidos. Esse arquivo contém os pesos e as configurações do modelo, capturando as informações aprendidas durante o treinamento. Com base nisso, o código mostrado na Figura 5 utiliza o modelo personalizado para realizar a detecção dos suínos na baia.

O código representa a implementação do modelo de detecção de suínos treinado. Ele utiliza técnicas de processamento de imagens para analisar quadro a quadro de um vídeo e identificar os suínos na baia. A detecção ocorre ao examinar cada quadro individualmente e realizar inferências com o modelo treinado. Ao percorrer o vídeo, o modelo gera informações sobre a presença e localização dos suínos detectados. Os objetos detectados são, então, marcados com caixas delimitadoras, um rótulo de confiança atribuída a cada detecção. Essas informações são sobrepostas ao quadro original, permitindo visualizar os suínos identificados.

Figura 5 – Código para detecção dos suínos na baia.

```python
from ultralytics import YOLO
import os
import cv2
from google.colab.patches import cv2_imshow

drive_path = "/content/gdrive/MyDrive/"
files = os.listdir(drive_path)

# Carregar o modelo YOLOv8
model = YOLO('best.pt')

# Abrir o arquivo de vídeo
video_path = "video2.mp4"
cap = cv2.VideoCapture(video_path)

# Definir as configurações do vídeo de saída
output_path = "output.mp4"
fourcc = cv2.VideoWriter_fourcc(*'XVID')
fps = cap.get(cv2.CAP_PROP_FPS)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

# Criar o objeto VideoWriter
out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

# Percorrer os quadros do vídeo
while cap.isOpened():
    # Ler um quadro do vídeo
    success, frame = cap.read()

    if success:
        # Executar a inferência do YOLOv8 no quadro
        results = model(frame)

        # Visualizar os resultados no quadro
        annotated_frame = results[0].plot()

        # Escrever o quadro anotado no vídeo de saída
        out.write(annotated_frame)

        # Mostrar o quadro anotado
        cv2_imshow(annotated_frame)

        # Encerrar o loop se a tecla 'q' for pressionada
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    else:
        # Encerrar o loop se o final do vídeo for alcançado
        break

# Liberar o objeto de captura de vídeo e o objeto VideoWriter
cap.release()
out.release()
cv2.destroyAllWindows()
```

Fonte: Acervo da pesquisa.

### 3.6 Rastreamento e Inserção de Identificação dos suínos

O código mostrado na Figura 6 utiliza o modelo personalizado (best.pt) para realizar o rastreamento e a inserção de identificação nos suínos.

Figura 6 – Código para rastreamento e inserção de identificação nos suínos.

```
from ultralytics import YOLO
import os
import cv2

caminho_drive = "/content/gdrive/MyDrive"
arquivos = os.listdir(caminho_drive)

# Carregar o modelo YOLOv8
modelo = YOLO('best.pt')

# Abrir o arquivo de vídeo
caminho_video = "video.mp4"
cap = cv2.VideoCapture(caminho_video)

# Definir as configurações de saída do vídeo
caminho_saida = "saida.mp4"
fourcc = cv2.VideoWriter_fourcc(*'XVID')
fps = cap.get(cv2.CAP_PROP_FPS)
largura = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
altura = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

# Criar o objeto VideoWriter
saida = cv2.VideoWriter(caminho_saida, fourcc, fps, (largura, altura))

# Inicializar um dicionário para armazenar as trilhas dos objetos
trilhas_objetos = {}

# Inicializar o contador de ID para os suínos
id_suino = 1

# Percorrer os quadros do vídeo
while cap.isOpened():
    # Ler um quadro do vídeo
    sucesso, quadro = cap.read()

    if sucesso:
        # Executar a inferência do YOLOv5 no quadro
        resultados = modelo(quadro)

        # Realizar a detecção de objetos
        if isinstance(resultados, list) and resultados:
            # Obter as coordenadas das caixas delimitadoras e as etiquetas
            caixas = resultados[0].caixas.xyxy.tolist()
            etiquetas = resultados[0].nomes

            # Criar um dicionário para armazenar os suínos detectados no quadro atual
            suinos_atuais = {}

            # Atualizar as trilhas dos objetos
            for i, caixa in enumerate(caixas):
                x1, y1, x2, y2 = [int(coord) for coord in caixa]

                # Calcular o centróide do objeto
                centroid_x = (x1 + x2) // 2
                centroid_y = (y1 + y2) // 2
```

Figura 6 (Continuação) – Código para rastreamento e inserção de identificação nos suínos.

```
                # Inicializar o ID do objeto para a caixa atual
                id_objeto_str = None

                # Verificar se existem trilhas de objetos existentes
                if trilhas_objetos:
                    # Calcular a distância euclidiana entre o centróide da caixa atual
                    # e os centróides das trilhas de objetos existentes
                    distancias = [
                        ((centroid_x - trilha['centroide'][0]) ** 2 +
                         (centroid_y - trilha['centroide'][1]) ** 2) ** 0.5
                        for trilha in trilhas_objetos.values()]

                    # Encontrar a trilha de objeto mais próxima dentro de uma distância limite
                    limite = 50  # Ajuste esse limite conforme necessário
                    indice_trilha_proxima = distancias.index(min(distancias))
                    id_trilha_proxima = list(trilhas_objetos.keys())[indice_trilha_proxima]
                    centroide_trilha_proxima = trilhas_objetos[id_trilha_proxima]['centroide']

                    if distancias[indice_trilha_proxima] < limite:
                        # Usar o ID existente da trilha mais próxima para a caixa atual
                        id_objeto_str = id_trilha_proxima

                # Se não houver trilha de objeto próxima o suficiente, criar um novo ID para a caixa atual
                if id_objeto_str is None:
                    id_objeto_str = f"Suino_{id_suino}"
                    id_suino += 1

                # Adicionar ou atualizar o ID do objeto no dicionário suinos_atuais
                suinos_atuais[id_objeto_str] = {'centroide': (centroid_x, centroid_y)}

                # Desenhar a caixa delimitadora e o ID no quadro
                cv2.rectangle(quadro, (x1, y1), (x2, y2), (0, 0, 255), 7)
                cv2.putText(quadro, id_objeto_str, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 4)

            # Atualizar o dicionário trilhas_objetos com os suínos detectados no quadro atual
            trilhas_objetos = suinos_atuais

        # Escrever o quadro com as sobreposições no vídeo de saída
        saida.write(quadro)

    else:
        # Encerrar o loop caso seja alcançado o final do vídeo
        break

# Liberar o objeto de captura de vídeo e o objeto VideoWriter
cap.release()
saida.release()
cv2.destroyAllWindows()
```

Fonte: Acervo da pesquisa.

O código apresentado tem como objetivo realizar a detecção e rastreamento dos suínos em um vídeo utilizando o modelo YOLOv8. Inicialmente, é carregado o modelo YOLOv8, em seguida, é aberto o arquivo de vídeo desejado e definidas as configurações para o vídeo de saída, como caminho, formato, taxa de quadros, largura e altura.

Durante a execução do *loop* principal, cada quadro do vídeo é lido e passado pelo modelo YOLOv8 para realizar a inferência e obter os resultados da detecção dos animais. Caso haja resultados válidos, ou seja, suínos detectados, são obtidas as coordenadas dos retângulos delimitadores e os rótulos correspondentes. Um dicionário é inicializado para armazenar o rastreamento dos animais. Também é inicializado um contador de IDs para os suínos do tipo "*pig*". Em seguida, ocorre a iteração sobre os animais detectados, onde são calculados os centróides e IDs dos suínos.

Para cada suíno, é verificado se existem rastreamentos pré-existentes. Caso existam, é calculada a distância euclidiana entre o centróide do suíno atual e os centróides dos rastreamentos pré-existentes, e o suíno é associado ao rastreamento mais próximo.

### 3.7 Análise de desempenho

Pra avaliar o desempenho do modelo, foram utilizadas quatro métricas diferentes: *precision* (precisão), *recall* (sensibilidade), *F1-score*, e *mean average precision* (mAP, medias das precisões de todas as métricas).

O parâmetro de precisão é uma medida estatística que quantifica a proporção de previsões corretas em relação ao número total de previsões feitas. Essa métrica é calculada dividindo o número de verdadeiros positivos (VP, previsões corretas) pelo número total de positivos previstos (que inclui tanto os verdadeiros positivos quanto os falsos positivos (FP)). A Equação 1 mostra o cálculo utilizado para determinar a precisão.

$$Precision = \frac{VP}{VP+FP} \qquad \text{(Equação 1)}$$

O *Recall* (Equação 2) é uma métrica que indica a capacidade de um modelo em identificar corretamente as instâncias positivas. Ele é calculado dividindo o número de verdadeiros positivos, pelo total de verdadeiros positivos (soma dos verdadeiros positivos e falsos negativos(FN)). Em outras palavras, o *Recall* representa a taxa de acerto na detecção das instâncias positivas, sendo um indicador importante da sensibilidade do modelo em relação aos casos positivos.

$$Recall = \frac{\text{VP}}{\text{VP+FN}} \qquad \text{(Equação 2)}$$

O *F1-score* é um parâmetro que combina as métricas de *Precision* e *Recall* por meio de uma média harmônica, conforme expresso pela Equação 3. A média harmônica atribui mais peso aos valores mais baixos, garantindo que uma pontuação alta de *F1-score* só seja alcançada se tanto o valor de *Precision* quanto o *Recall* forem altos. Dessa forma, o *F1-score* é uma medida que avalia o equilíbrio entre a precisão e a capacidade de recuperar corretamente as instâncias positivas do modelo.

$$F1 = \frac{VP}{VP+\frac{FN+FP}{2}} \qquad \text{(Equação 3)}$$

A métrica mAP (*mean Average Precision*) (Equação 4) calcula a média das precisões médias para diferentes níveis de confiança. O mAP considera tanto o valor de *Precision* quanto o *Recall*, fornecendo uma visão geral do desempenho do modelo. Quanto maior o mAP, melhor o desempenho na detecção e localização dos objetos.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} PMi \qquad \text{(Equação 4)}$$

Em que: “PMi” é a precisão média de cada classe e “N” é o número total de classes, no caso, neste trabalho estamos analisando apenas uma classe: *Pig*.

# 4. RESULTADOS E DISCUSSÃO

## 4.1 Modelo treinado

Na Figura 7 é mostrado que o treinamento foi interrompido antes do número total de épocas (300 épocas), devido à falta de melhoria nos resultados nas últimas 50 épocas. A melhor performance foi observada na época 49 e o modelo correspondente foi salvo como "best.pt".

Figura 7 - Interrupção do treinamento por falta de melhoria nas métricas de desempenho.


Fonte: Acervo da pesquisa.

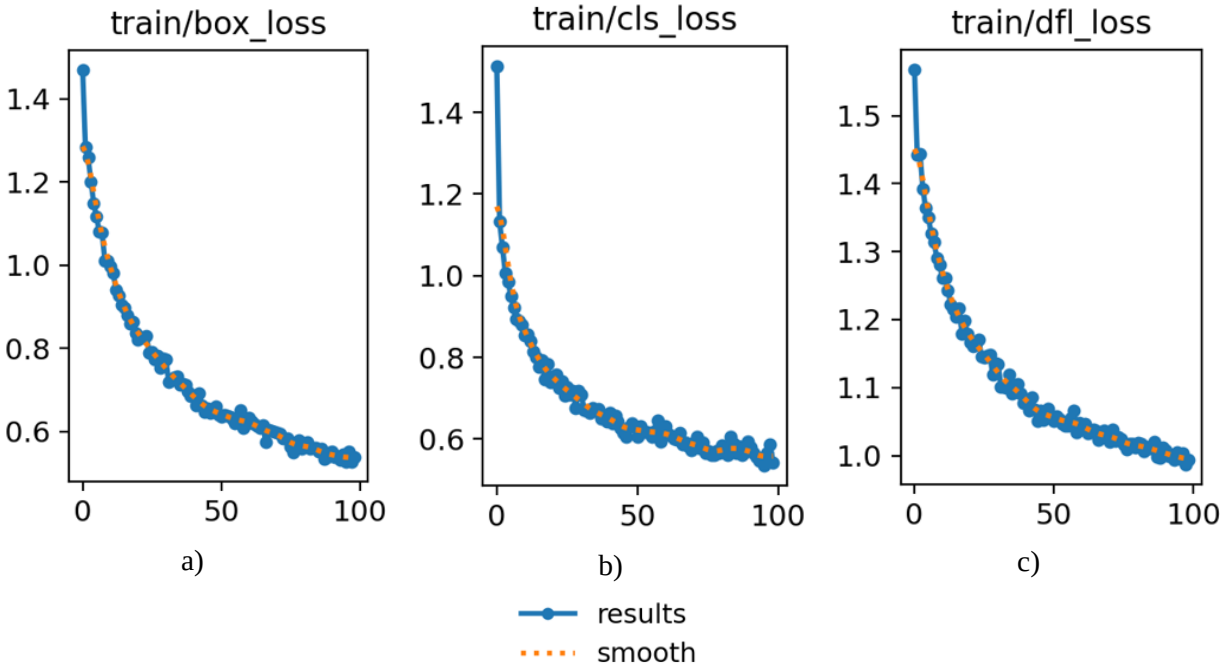
Interromper o treinamento com base na falta de melhoria nas métricas de desempenho, evita que o modelo continue a se ajustar excessivamente aos dados de treinamento e se torne muito

específico para eles. Isso permite obter um modelo que tenha uma melhor capacidade de generalização e seja capaz de realizar previsões mais precisas em dados não vistos anteriormente.

O desempenho do modelo no decorrer do treinamento está representado nas Figuras 8 e 9. Pode-se observar, a partir das figuras, uma representação visual das métricas de perda (*loss*), a *box_loss* representa a perda associada à precisão da localização dos objetos detectados, a *cls_loss* está relacionada à precisão da classificação das categorias e a *dfl_loss* captura a perda decorrente da estimação dos deslocamentos entre caixas delimitadoras.

As curvas de treinamento (*train*) (Figura 8) representam o desempenho do modelo durante o processo de treinamento, onde os pesos e parâmetros do modelo são ajustados iterativamente com base nos dados de treinamento. Essa curva mostra como a perda do modelo diminui ao longo das épocas de treinamento. É esperado que a perda diminua à medida que o modelo aprende a fazer previsões mais precisas e se ajusta melhor aos dados de treinamento.

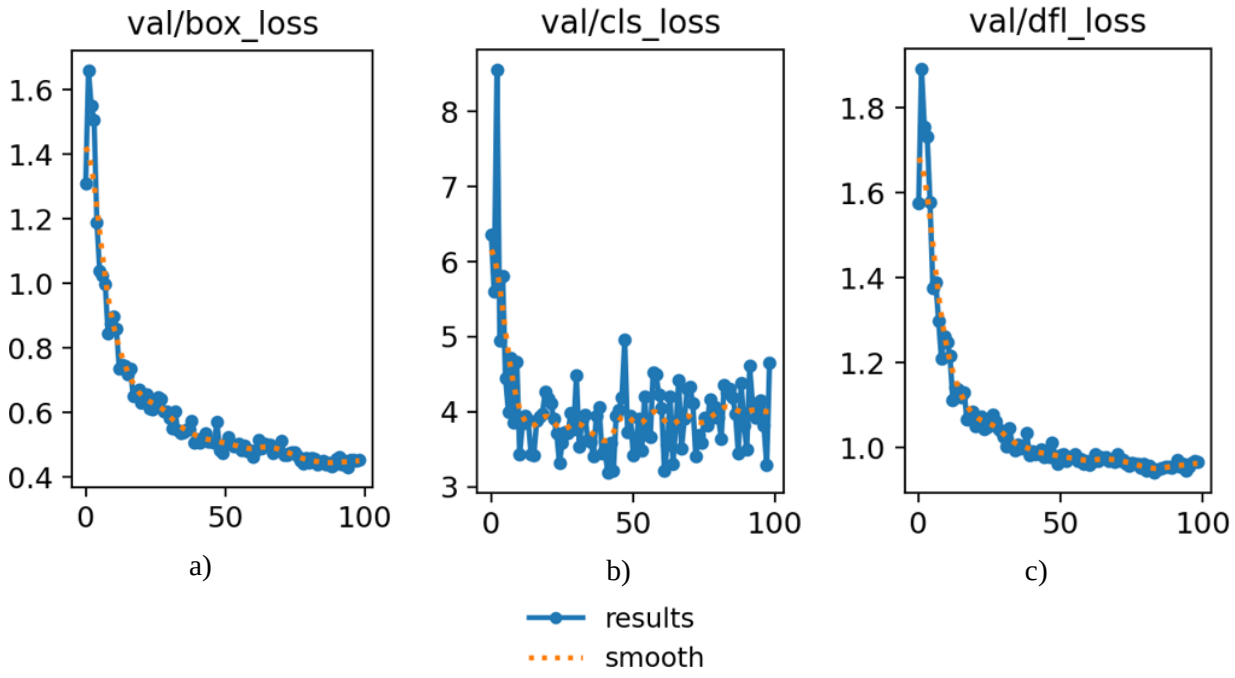
Figura 8 - Comportamento das funções de perda (*loss*) no decorrer das épocas durante o treinamento.




Fonte: Acervo da pesquisa.

Por outro lado, as curvas de validação (*val*) (Figura 9) representam o desempenho do modelo no conjunto de validação. Esse conjunto de dados é usado para avaliar o desempenho do modelo em dados não vistos durante o treinamento e verificar se o modelo está generalizando bem. A curva de validação mostra como a perda diminui à medida que o modelo é avaliado em diferentes épocas.

Figura 9 - Comportamento das funções de perda (*loss*) no decorrer das épocas durante a validação.

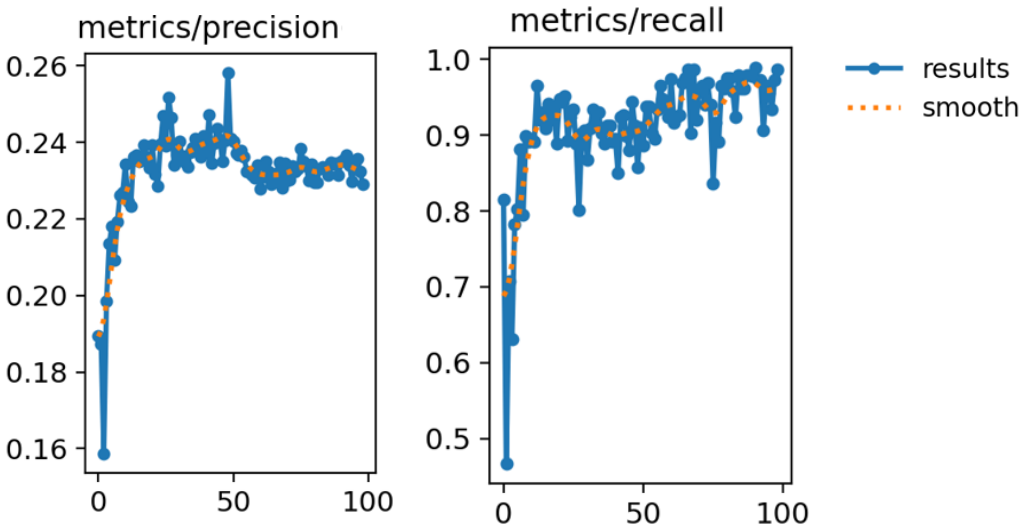



Fonte: Acervo da pesquisa.

Comparando as duas curvas, é possível avaliar o *trade-off* entre o desempenho no conjunto de treinamento e a capacidade de generalização do modelo para dados não vistos. Apesar da curva de treinamento ainda diminuir após a época 49, é possível observar (através do traçado *smooth*) que a curva de validação começa a aumentar em *cls_loss* (Figura 9 “b”), apresentando sinal de *overfitting*, indicando que o modelo está se ajustando muito bem aos dados de treinamento, mas não está generalizando bem. Portanto, a época 49 foi identificada como o ponto ideal que maximiza tanto o ajuste aos dados de treinamento quanto a capacidade de generalização do modelo.

O modelo treinado apresentou valores de *precision* e *recall* de 0,258 e 0,855 respectivamente. O valor de *recall* de 0,855 indica que o modelo conseguiu identificar corretamente 85,5% das instâncias que deveriam ter sido detectadas. Já o valor de *precision* de 0,258 sugere que, das instâncias detectadas, apenas 25,8% foram classificadas corretamente. O comportamento das métricas *precision* e *recall* está representado na Figura 10, onde é possível observar que, apesar da precisão cair após a época 49, o *recall* continua a aumentar, indicando que o modelo apresenta um melhor desempenho ao localizar e identificar os suínos na imagem, em comparação com sua precisão na classificação correta.

Figura 10 - Comportamento das métricas *precision* e *recall* no decorrer de cada época.



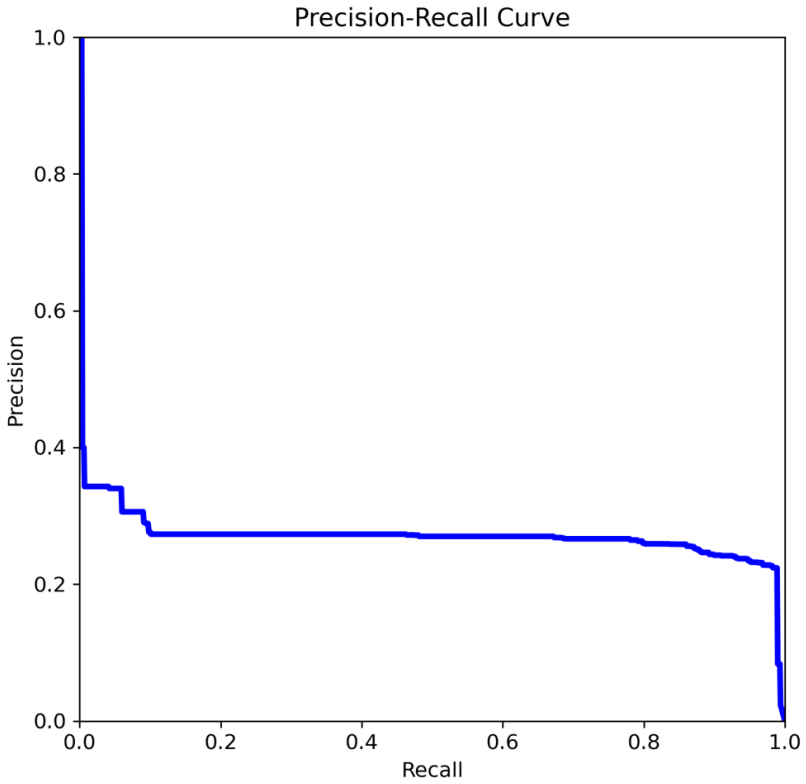

Fonte: Acervo da pesquisa.

Os estudos realizados por Ang *et al.* (2023) e Phan *et al.* (2023) demonstraram que a utilização do modelo YOLOv8 resultou em uma precisão superior a 0,9. Para atingir esses resultados, eles empregaram conjuntos de dados significativamente maiores, compreendendo 4674 e 2000 imagens, respectivamente. No entanto, neste trabalho, o modelo foi treinado com apenas 690 imagens, o que pode ter impactado negativamente no valor da precisão. A utilização de um conjunto de dados reduzido pode restringir a capacidade do modelo de capturar a diversidade de

padrões e características presentes nos objetos a serem detectados, resultando em um desempenho inferior ao ser comparado com estudos que empregaram conjuntos de dados mais abrangentes.

A curva de *recall* x *precision* (Figura 11) revela o comportamento do modelo em relação à detecção e classificação correta das instâncias. Pode-se observar que a precissão se mantém relativamente constante enquanto o *recall* continua a crescer, sugerindo que o modelo está encontrando mais instâncias sem afetar significativamente a proporção de falsos positivos. Isso indica que o modelo está sendo mais tolerante a falsos positivos, a alta taxa de *recall* indica que o modelo está identificando a maioria das instâncias relevantes, mesmo que algumas classificações estejam incorretas.

Figura 11 - Comportamento das métricas *precision* e *recall* longo das épocas de treinamento.



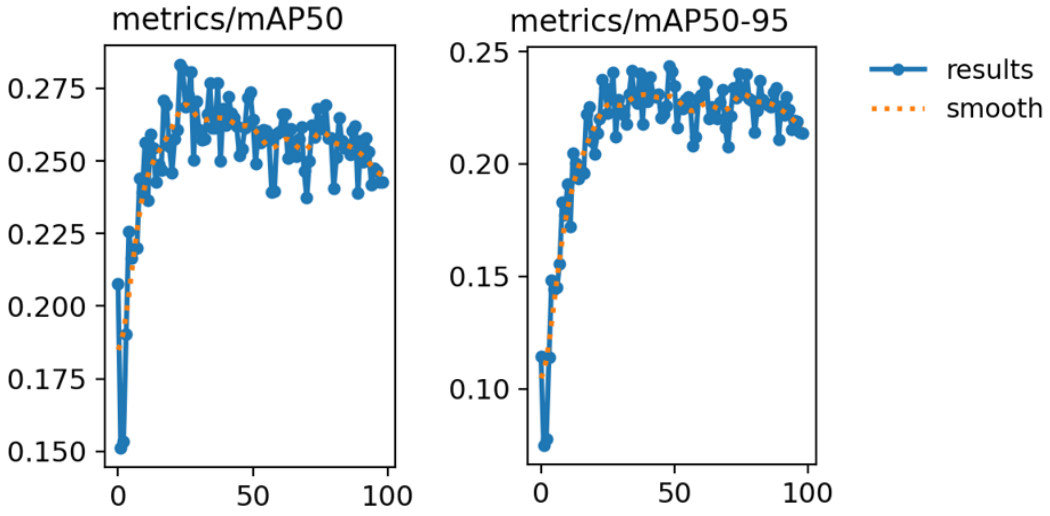

Fonte: Acervo da pesquisa.

O *recall* obteve um bom desempenho, atingindo 0,855, enquanto o valor de precisão foi baixo, com uma pontuação de 0,258 para a época 49. A métrica *F1-score*, combina *recall* e precisão

em um único valor, considerando tanto a capacidade do modelo em identificar corretamente as instâncias relevantes (*recall*) quanto a sua precisão em classificá-las corretamente. O valor de F1 obtido foi de 0,40, indicando que o modelo apresenta um baixo desempenho.

Na Figura 12, são ilustrados o desempenho da métrica mAP ao longo das épocas de treinamento. Em “a” está o mAP50, que considera apenas a precisão média para um limiar de confiança de 0,5. Enquanto em “b” é mostrado desempenho do modelo em uma faixa mais abrangente de confianças, que vai de 0,5 até 0,95. Observando as curvas de mAP, nota-se que, inicialmente, há um aumento significativo no valor de mAP, indicando uma melhora progressiva do modelo à medida que o treinamento avança. Isso sugere que o modelo está aprendendo a localizar e classificar os suínos com maior precisão ao longo do tempo.

Figura 12 - Comportamento da métrica mAP (média da Precisão Média), no decorrer de cada época.




a) b)

Fonte: Acervo da pesquisa.

Para a época 49, o mAP50 apresentou um valor de 0,27187. Isso indica que, em média, o modelo obteve uma precisão de aproximadamente 27,2% na detecção de suínos com uma sobreposição mínima de 50%. Enquanto o valor de mAP50-95 foi de 0,24356, sugerindo que o modelo enfrenta mais dificuldades em detectar suínos, tanto para uma sobreposição de 50% como

para uma sobreposição mais alta. Esses resultados apontam para a necessidade de melhorias e otimizações no treinamento do modelo, como o aumento do conjunto de dados de treinamento.

Khalid *et al.* (2023), testaram a CNN de diversas versões do YOLO (Yolo v3, Yolov3-Tiny, Yolov4, Yolov4-Tiny, Yolov6 e Yolov8), e obtiveram o melhor valor de mAP para o modelo YOLOv8, 84,7%. Para isso, eles criaram um conjunto de dados de 9875 imagens, criando um modelo eficiente para detecção de pragas em pequenas culturas. Além disso, Kim *et al.* (2021) obteveram um valor de mAP maior que 90% usando YOLO para detectar o comportamento de comer e alimentar de leitões, utilizando um total de 9880 imagens para o treinamento. Em outros estudos, também é comprovada e reconhecida a eficácia da CNN YOLO na detecção de objetos (SEO *et al.*, 2019; ALAMEER *et al.*, 2020; BHUJEL, *et al.*, 2021; TU *et al.*, 2022; ZHENG *et al.*, 2023). Portanto, a limitação dos resultados obtidos neste trabalho pode ser atribuída ao tamanho reduzido do conjunto de imagens utilizado no treinamento.

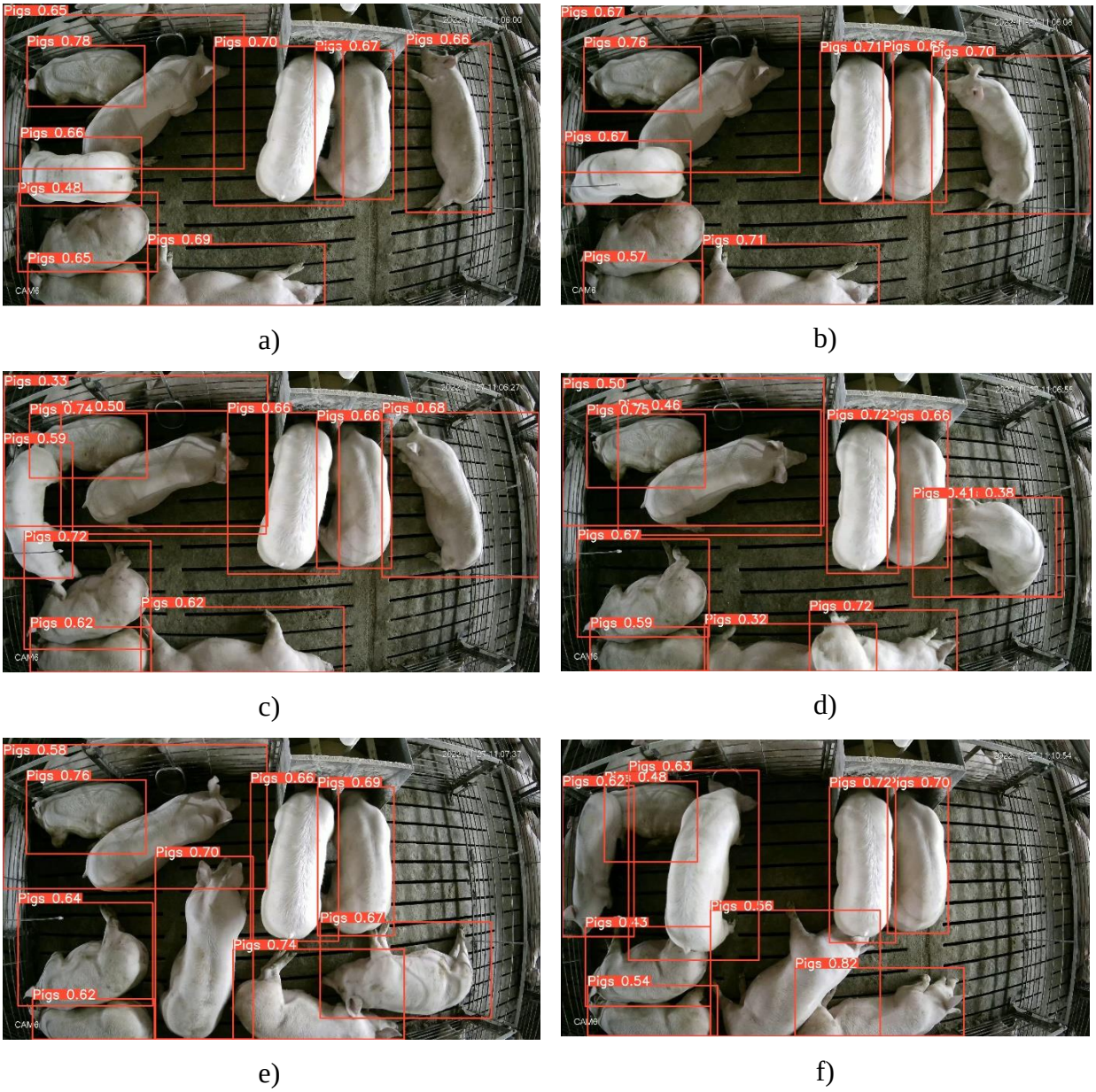
É importante destacar que a detecção de suínos pode ser um desafio quando o treinamento é realizado com um número limitado de imagens. Nas baias de suínos, é comum que os animais fiquem agrupados e próximos uns dos outros, o que pode resultar em oclusão e dificuldade na identificação individual de cada suíno. Um treinamento robusto, que leve em consideração essa característica, é essencial para garantir uma detecção precisa e confiável. Ao utilizar um conjunto de dados mais abrangente e diversificado, que represente diferentes contextos e condições de oclusão, o modelo terá a oportunidade de aprender uma variedade maior de padrões e características, melhorando sua capacidade de detecção dos suínos nas baias.

Após treinar o modelo, foi realizado um teste para avaliar o desempenho em tempo real. O modelo foi submetido a um conjunto de vídeos, e os resultados mostraram um FPS (*frames per second*) médio de 42,47. Isso indica que o modelo é capaz de processar aproximadamente 42 quadros por segundo. Wang *et al.* (2022) relataram que é necessário atingir 24 FPS para garantir a detecção em tempo real. Logo, o modelo demonstra uma taxa satisfatória para implementações em tempo real.

## 4.2 Detecção dos suínos

Na Figura 13 estão representadas algumas imagens capturadas de um vídeo no período da tarde, em que foi usado o modelo desenvolvido para detectar os suínos.

Figura 13 – Análise da detecção dos suínos durante o dia utilizando o modelo treinado.




Fonte: Acervo da pesquisa.

O vídeo analisado, foi usado apenas para avaliar a eficiência do modelo, ele não foi utilizado no treinamento. Além disso, foi escolhido um vídeo em que os suínos estavam bastante ativos para proporcionar a identificação de possíveis falhas do modelo durante a detecção.

Cada caixa delimitadora apresenta uma pontuação de confiança que permite avaliar a certeza do modelo em relação à detecção de cada objeto. Valores mais altos indicam uma maior confiança na presença desse objeto na imagem. Na figura 13 é possível observar que em “c” e “d” apresentaram mais caixas delimitadoras que suínos na imagem, já em “b” um suíno ficou sem identificação.

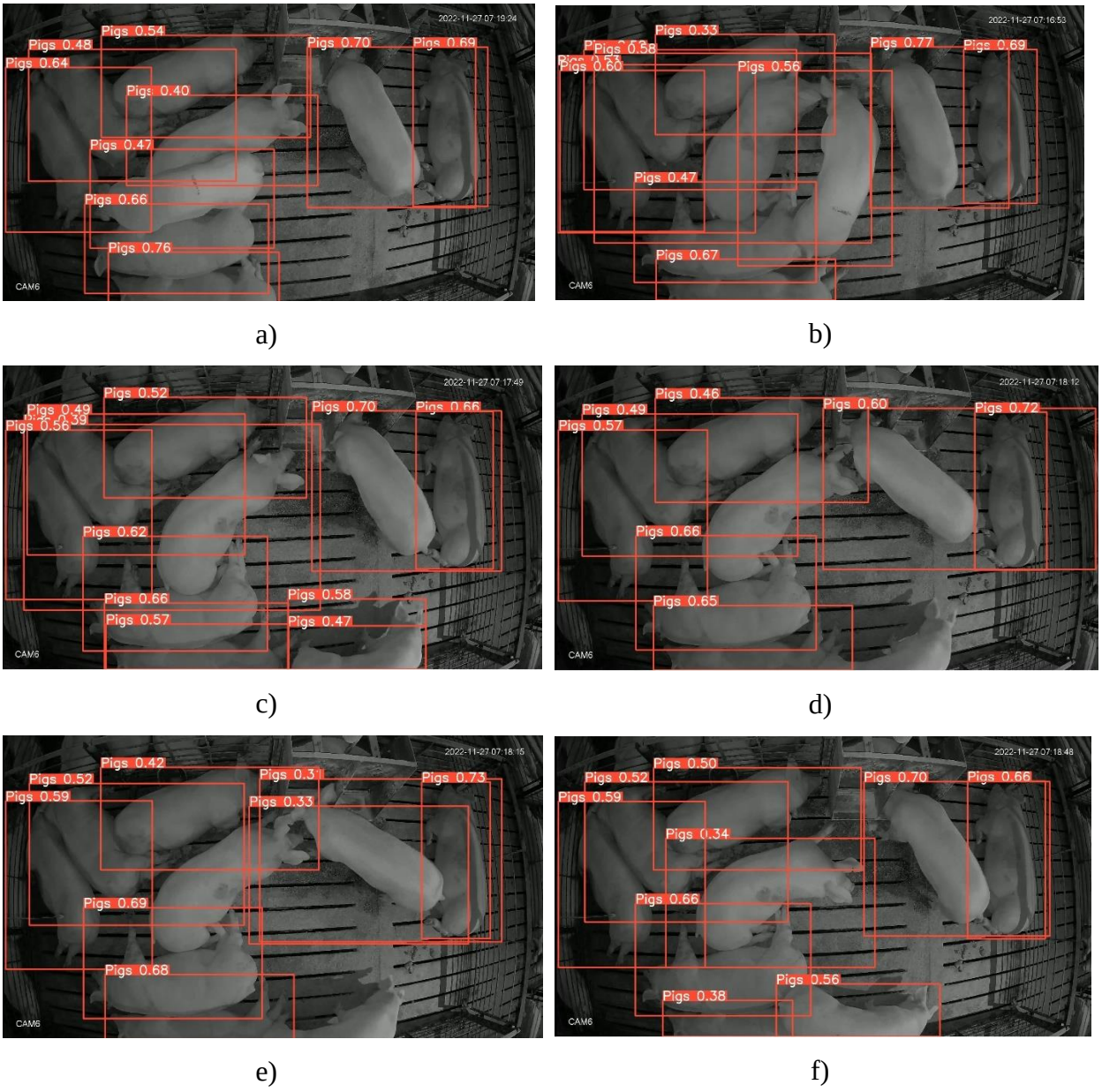
O vídeo foi analisado em câmera lenta e foi possível observar que em 27,22% do tempo do vídeo, foi apresentado alguma falha na detecção, dessas falhas, na maior parte do tempo foi apresentado um número de caixas delimitadoras maior do que a quantidade de suínos. A maior parte do vídeo os animais foram detectados corretamente utilizando o modelo, porém a pontuação de confiança não ultrapassou o valor de 0,85.

Analisando o modelo para detecção dos suínos no escuro (Figura 14), foi possível observar que em 37,5% do vídeo analisado, foi apresentado falhas. Diferente do vídeo durante o dia, a maior parte das falhas foi por falta de detecção de suínos, como é apresentado na Figura 14 em “d” e “e”. Na maior parte do tempo do vídeo os animais foram detectados corretamente, e para esta situação, a pontuação de confiança não ultrapassou o valor de 0,81.

No geral, a detecção dos suínos por meio do modelo desenvolvido foi considerada satisfatória. No entanto, foram observadas falhas especialmente quando os suínos estavam em movimento intenso e exibiam comportamentos rápidos e imprevisíveis. Essas situações resultaram em algumas omissões e imprecisões na detecção. Além disso, a pontuação de confiança associada às caixas delimitadoras dos suínos detectados poderia ser aprimorada.

O aumento do conjunto de dados de treinamento, alimentando o modelo com uma maior variedade de imagens que representem diferentes ângulos, poses e condições de iluminação, forneceria ao modelo a oportunidade de aprender melhor os padrões e características dos suínos, aumentando a sua capacidade de detecção e atribuição de pontuações de confiança mais precisas.

Figura 14 – Análise da detecção dos suínos no escuro utilizando o modelo treinado.




Fonte: Acervo da pesquisa.

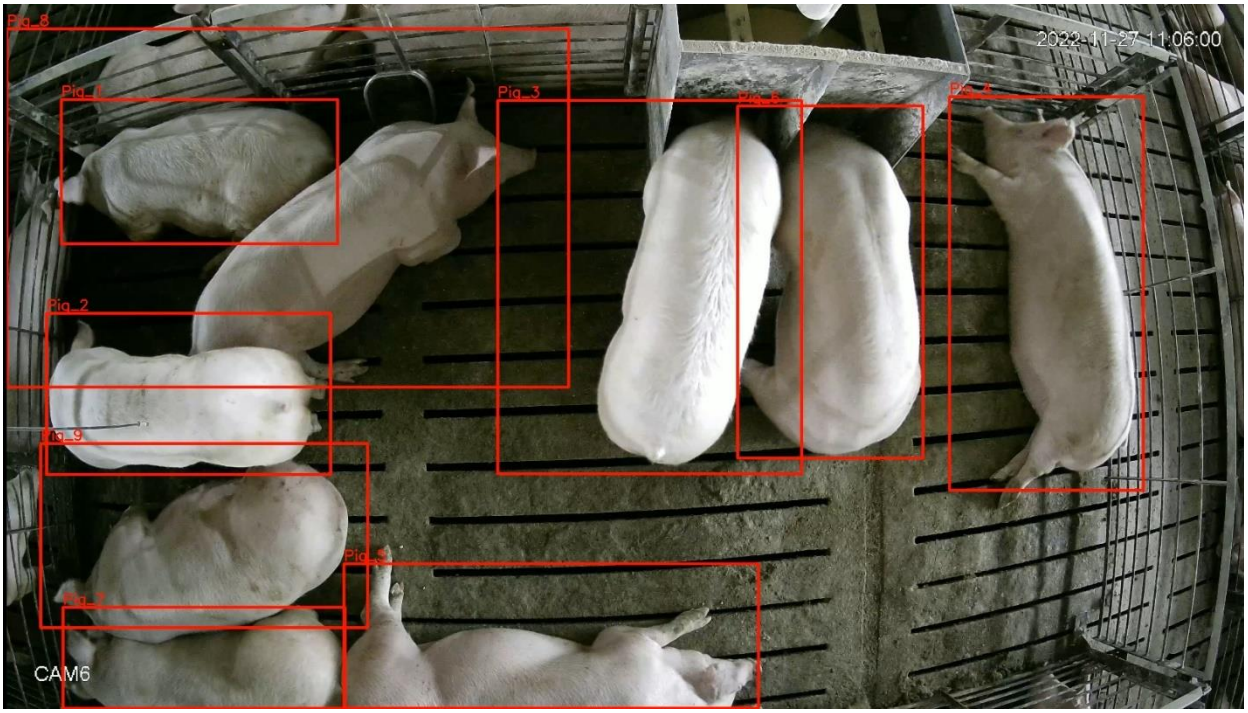
### 4.3 Rastreamento e Inserção de Identificação nos suínos

O código desenvolvido, utilizando a arquitetura YOLO, foi capaz de rastrear e atribuir um ID único a cada animal, garantindo uma identificação individual ao longo do tempo. No entanto

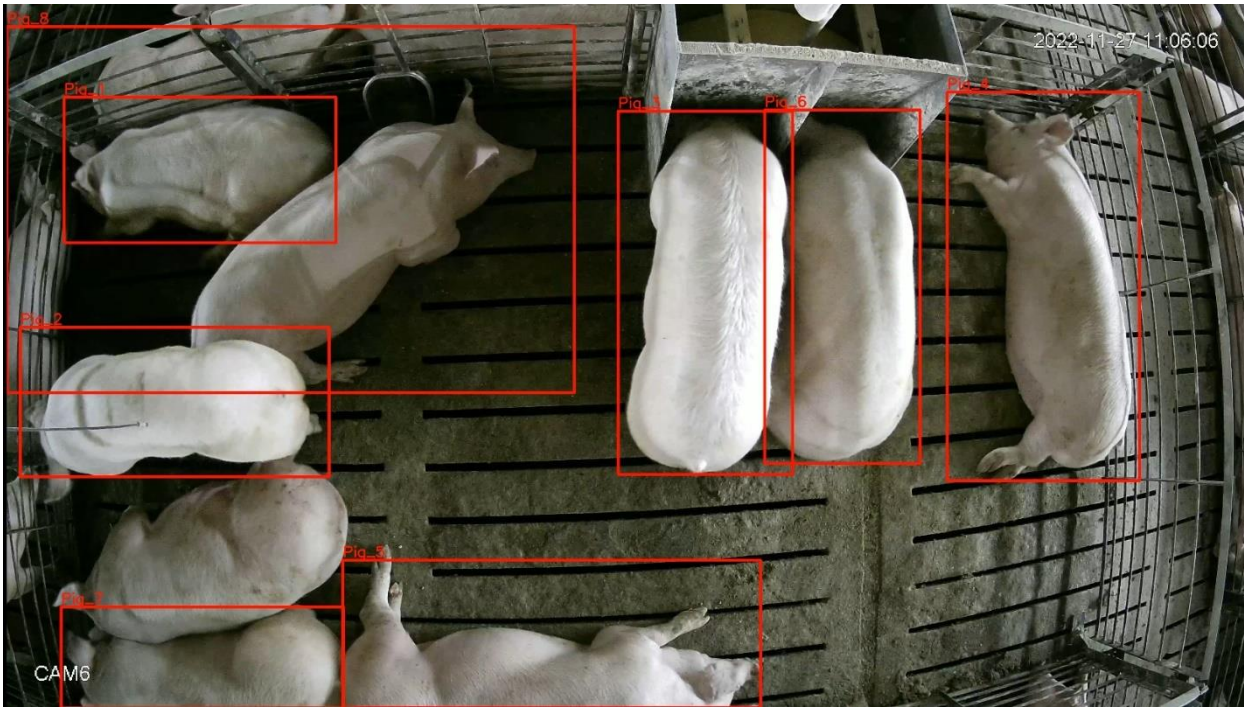
não foi capaz de lidar com as falhas de detecção. Inicialmente, cada suíno recebe um ID que consegue acompanhar o movimento do animal. Porém, quando ocorre uma falha na detecção, o código não foi capaz de estabelecer o mesmo ID que o suíno recebeu no início. Pode-se observar na Figura 15, que todos os suínos são indificados com um ID específico em “a”. já em “b” é mostrado o exato momento que que o suíno 9 deixa de ser detectado. Quando o suíno volta a ser detectado, ele já apresente um novo ID (Figura 15 “c”).

O rastreamento e identificação dos suínos também foram impactados por que o modelo desenvolvido não estava totalmente robusto. Houve momentos em que as caixas delimitadoras dos suínos foram perdidas durante o rastreamento, resultando em perdas de identificação e inconsistências no acompanhamento dos animais ao longo do vídeo. Essas dificuldades no rastreamento comprometeram a obtenção de trajetórias precisas e confiáveis dos suínos.

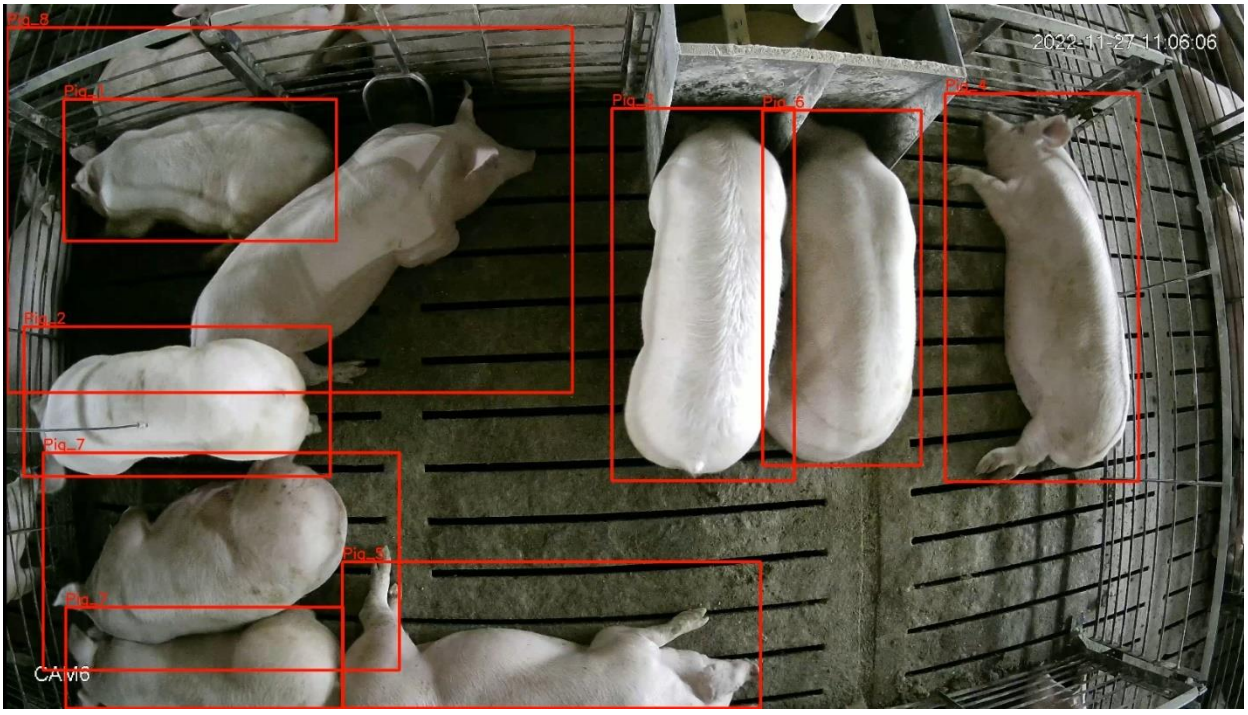
Figura 15 – Análise do rastreamento e identificação dos suínos.




a)




b)




c)

Fonte: Acervo da pesquisa.

Mesmo com um modelo robusto, podem ocorrer falhas, uma abordagem que pode melhorar significativamente o rastreamento e a identificação dos suínos é combinar o modelo desenvolvido com o algoritmo *Deep SORT* (*Simple Online and Realtime Tracking with a Deep Association Metric*). Alguns trabalhos têm usado a associação dessas metodologias para criar um modelo híbrido que aproveita as capacidades de detecção precisa do modelo desenvolvido e a precisão do rastreamento e identificação do *Deep SORT* (ZHANG *et al.*, 2020; HASIBUAN *et al.*, 2021; PARICO *et al.*, 2021; EGI *et al.*, 2022; GAO *et al.*, 2022). Essa combinação pode ser uma alternativa valiosa para melhorar a qualidade do rastreamento e a inserção confiável de identificadores nos suínos, contribuindo para um monitoramento mais eficiente e preciso das suas trajetórias.

## 5. CONCLUSÃO

O estudo propôs e desenvolveu um modelo de rastreamento de suínos utilizando a arquitetura YOLOv8. Os resultados obtidos mostraram que o modelo apresentou baixa eficiência devido ao número limitado de imagens utilizadas no treinamento. Porém foi possível rastrear e estabelecer uma identificação individual para cada suíno. Além disso, o modelo demonstrou uma promissora taxa de FPS, o que indica sua capacidade de processamento em tempo real e viabilidade para aplicações práticas.

Os resultados, embora evidenciem a necessidade de um conjunto de treinamento mais robusto, destacam o potencial do modelo proposto para contribuir com a gestão e bem-estar animal na suinocultura. Com o aprimoramento do conjunto de dados e a otimização do modelo, espera-se superar as limitações iniciais e alcançar um desempenho ainda mais satisfatório.

## REFERÊNCIAS

ABADI, Aji Bijaksana; TAHCFULLOH, Syahfrizal. Digital Image Processing for Height Measurement Application Based on Python OpenCV and Regression Analysis. **JOIV: International Journal on Informatics Visualization**, v. 6, n. 4, p. 763-770, 2022.

ABIODUN, Oludare Isaac *et al*. State-of-the-art in artificial neural network applications: A survey. **Heliyon**, v. 4, n. 11, p. e00938, 2018.

ABOAH, Armstrong *et al.* Real-time multi-class helmet violation detection using few-shot data sampling technique and yolov8. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**, pág. 5349-5357, 2023.

ALAMEER, Ali; KYRIAZAKIS, Ilias; BACARDIT, Jaume. Automated recognition of postures and drinking behaviour for the detection of compromised health in pigs. **Scientific reports**, v. 10, n. 1, p. 1-15, 2020.

ALY, Ghada Hamed *et al.* YOLO based breast masses detection and classification in full-field digital mammograms. **Computer Methods and Programs in Biomedicine**, v. 200, p. 105823, 2021.

ANG, Guang Jun Nicholas *et al.* A novel application for real-time arrhythmia detection using YOLOv8. **arXiv preprint arXiv**:2305.16727, 2023.

ARNOLD, Taylor; TILTON, Lauren. Distant viewing Toolkit: A python package for the analysis of visual culture. **Journal of Open Source Software**, v. 5, n. 45, p. 1800, 2020.

BHUJEL, Anil *et al.* A lightweight Attention-based convolutional neural networks for tomato leaf disease classification. **Agriculture**, v. 12, n. 2, p. 228, 2022.

BHUJEL, Anil *et al.* Deep-Learning-Based Automatic Monitoring of Pigs' Physico-Temporal Activities at Different Greenhouse Gas Concentrations. **Animals**, v. 11, n. 11, p. 3089, 2021.

BIN, Li *et al.* Pig posture detection based on improved YOLOv4 model. **Acta Agriculturae Zhejiangensis**, v. 35, n. 1, p. 215, 2023.

BO, Li *et al.* Head and Tail Identification Method for Group-housed Pigs Based on YOLO v3 and Pictorial Structure Model. **Nongye Jixie Xuebao/Transactions of the Chinese Society of Agricultural Machinery**, v. 51, n. 7, 2020.

BOYLE, Coleen A. et al. The public health response to the COVID-19 pandemic for people with disabilities. **Disability and Health Journal**, v. 13, n. 3, p. 100943, 2020.

CHAKRAVARTULA, Swathi Sirisha Nallan *et al.* Use of convolutional neural network (CNN) combined with FT-NIR spectroscopy to predict food adulteration: A case study on coffee. **Food Control**, v. 135, p. 108816, 2022.

CHEN, Chunling *et al.* YOLO-Based UAV Technology: A Review of the Research and Its Applications. **Drones**, v. 7, n. 3, p. 190, 2023.

CHEN, Guang, *et al.* Efficient pig counting in crowds with key points tracking and spatial-aware temporal response filtering. In: **2020 IEEE International Conference on Robotics and Automation (ICRA)**. IEEE, p. 10052-10058, 2020.

COVID-19 in the pork industry. **Animals**, v. 11, n. 3, p. 830, 2021.

DE ALMEIDA, Tulio Fernandes *et al.* PyRAT: an open-source *Python* library for animal behavior analysis. **Frontiers in Neuroscience**, v. 16, p. 505, 2022.

DICKSON-KARN, Nicole M.; OROSZ, Steven. Implementation of a Python Program to Simulate Sampling. **Journal of Chemical Education**, v. 98, n. 10, p. 3251-3257, 2021.

DING, Qi-an, *et al.* Activity detection of suckling piglets based on motion area analysis using frame differences in combination with convolution neural network. **Computers and Electronics in Agriculture**, v. 194, p. 106741, 2022.

DIWAN, Tausif; ANIRUDH, Gupta; TEMBHURNE, Jitendra Vitthal. Object detection using YOLO: Challenges, architectural successors, datasets and applications. **multimedia Tools and Applications**, v. 82, n. 6, p. 9243-9275, 2023.

DORNADULA, Sai Paavan Kumar; BRUNET, Pascal; ELIAS, Susan. AI driven shadow model detection in agropv farms. **arXiv preprint arXiv**:**2304.07853**, 2023.

DUDEKULA, Khasim Vali *et al.* Convolutional Neural Network-Based Personalized Program Recommendation System for Smart Television Users. **Sustainability**, v. 15, n. 3, p. 2206, 2023.

DUMITRIU, Andrei *et al.* Rip Current Segmentation: A Novel Benchmark and YOLOv8 Baseline Results. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**, p. 1261-1271, 2023.

EGI, Yunus; HAJYZADEH, Mortaza; EYCEYURT, Engin. Drone-Computer Communication Based Tomato Generative Organ Counting Model Using YOLO V5 and Deep-Sort. **Agriculture**, v. 12, n. 9, p. 1290, 2022.

EUROPEAN COURT OF AUDITORS. **Animal Welfare in the EU: Closing the Gap between Ambitious Goals and Practical Implementation**. 2018.

FUKUI, Hiroshi *et al.* Attention branch network: Learning of attention mechanism for visual explanation. In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**, p. 10705-10714, 2019.

GAO, Bo. Research on Two-Way Detection of YOLO V5s+ Deep Sort Road Vehicles Based on Attention Mechanism. In: **Journal of Physics: Conference Series.** IOP Publishing, p. 012057, 2022.

GARCIA, Rodrigo, *et al.* A systematic literature review on the use of machine learning in precision livestock farming. **Computers and Electronics in Agriculture**, v. 179, p. 105826, 2020.

GOJKOVIĆ, Ljubomir; MALIJEVIĆ, Stefan; ARMAKOVIĆ, Stevan. Modeling of fundamental electronic circuits by the Euler method using the Python programming language. **Physics Education**, v. 55, n. 5, p. 055016, 2020.

GRANDIN, Temple. Methods to prevent future severe animal welfare problems caused by COVID-19 in the pork industry. **Animals**, v. 11, n. 3, p. 830, 2021.

GUO, Qinghua, *et al.* Video-based Detection and Tracking with Improved Re-Identification Association for Pigs and Laying Hens in Farms. In: **Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications**. SciTePress, 2022.

GUPTA, Ankur *et al*. Automated determination of interfacial tension and contact angle using computer vision for oil field applications. **Journal of Petroleum Exploration and Production Technology**, p. 1-9, 2021.

HAKIM, Wahyu Luqmanul *et al.* Convolutional neural network (CNN) with metaheuristic optimization algorithms for landslide susceptibility mapping in Icheon, South Korea. **Journal of environmental management**, v. 305, p. 114367, 2022.

HAN, Su-Hyun *et al.* Artificial neural network: understanding the basic concepts without mathematics. **Dementia and neurocognitive disorders**, v. 17, n. 3, p. 83-89, 2018.

HANIF, Murtaza *et al.* An Accessible Agent for Blind's Self-Localization and Navigational Assistance. **Preprint submitted to Elsevier**. 2023.

HASIBUAN, Nisma Novita; ZARLIS, Muhammad; EFENDI, Syahril. Detection and tracking different type of cars with YOLO model combination and deep sort algorithm based on computer vision of traffic controlling**. Sinkron: jurnal dan penelitian teknik informatika**, v. 6, n. 1, p. 210-221, 2021.

JAMTSHO, Yonten; RIYAMONGKOL, Panomkhawn; WARANUSAST, Rattapoom. Real-time license plate detection for non-helmeted motorcyclist using YOLO. **Ict Express**, v. 7, n. 1, p. 104-109, 2021.

JUNG, Wonseok, *et al.* An AIOT monitoring system for multi-object tracking and alerting. **CMC-Computers Materials & Continua**, v. 67, n. 1, p. 337-348, 2021.

KANG, Jichang *et al.* Research on an Improved YOLOV8 Image Segmentation Model for Crop Pests. **Advances in Computer, Signals and Systems**, v. 7, n. 3, p. 1-8, 2023.

KARAMAN, Ahmet *et al.* Robust real-time polyp detection system design based on YOLO algorithms by optimizing activation functions and hyper-parameters with artificial bee colony (ABC). **Expert Systems with Applications**, v. 221, p. 119741, 2023.

KHALID, Saim *et al.* Small Pests Detection in Field Crops Using Deep Learning Object Detection. **Sustainability,** v. 15, n. 8, p. 6815, 2023.

KIM, Jun-Hwa; KIM, Namho; WON, Chee Sun. High-Speed Drone Detection Based On Yolo-V8. In: **ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE**, p. 1-2, 2023.

KIM, MinJu *et al.* A deep learning-based approach for feeding behavior recognition of weanling pigs**. Journal of animal science and technology**, v. 63, n. 6, p. 1453, 2021.

KOSONEN, Jussi. Tekoälypohjainen hahmontunnistus–YOLO-algoritmin käyttöönotto, kouluttaminen ja käyttöliittymän kehittäminen. **All Sciences Proceedings**. 2023.

KULIKOV, Andrey *et al*. Modelling Optimal Capital Structure in Gas and Oil Sector by Applying Simulation Theory and Programming Language of Python (Qatar Gas Transport Company). **Energies**, v. 16, n. 10, p. 4067, 2023.

KWON, Taehyung, *et al.* Tracing the breeding farm of a domesticated pig using feature selection (Sus scrofa). **Asian-Australasian Journal of Animal Sciences**, v. 30, n. 11, p. 1540, 2017.

LEBRET, Bénédicte *et al.* Pork quality attributes from farm to fork. Part I. Carcass and fresh meat. **Animal**, v. 16, p. 100402, 2022.

LEE, Jeonghun; HWANG, Kwang-il. YOLO with adaptive frame control for real-time object detection applications. **Multimedia Tools and Applications**, v. 81, n. 25, p. 36375-36396, 2022.

LI, Dan, *et al.* Research advance on computer vision in behavioral analysis of pigs. **Journal of Agricultural Science and Technology (Beijing)**, v. 21, n. 7, p. 59-69, 2019.

LI, Guangbo; SHI, Guolong; JIAO, Jun. YOLOv5-KCB: A New Method for Individual Pig Detection Using Optimized K-Means, CA Attention Mechanism and a Bi-Directional Feature Pyramid Network. **Sensors**, v. 23, n. 11, p. 5242, 2023.

LI, Mengying *et al.* Structural damage identification using strain mode differences by the iFEM based on the convolutional neural network (CNN). **Mechanical Systems and Signal Processing**, v. 165, p. 108289, 2022.

LI, Peixia, *et al.* Deep visual tracking: Review and experimental comparison. **Pattern Recognition**, v. 76, p. 323-338, 2018.

LI, Shudong *et al.* Detection method for individual pig based on improved YOLOv4 Convolutional Neural Network. **In: 2021 4th International Conference on Data Science and Information Technology.** pág. 231-235, 2021.

LI, Yiting *et al.* A Modified YOLOv8 Detection Network for UAV Aerial Image Recognition. **Drones**, v. 7, n. 5, p. 304, 2023.

LIANG, G. *et al.* Combining convolutional neural network with recursive neural network for blood cell image classification. **IEEE Access 6**, 36188–36197, 2018.

LIANG, Siyuan *et al.* Edge YOLO: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles. **IEEE Transactions on Intelligent Transportation Systems**, v. 23, n. 12, p. 25345-25360, 2022.

LOU, Haitong *et al.* DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor. **Electronics**, v. 12, n. 10, p. 2323, 2023.

LUO, Ziyuan; YAN, Wei Qi; NGUYEN, Minh. Kayak and sailboat detection based on the improved YOLO with Transformer. In: **Proceedings of the 5th International Conference on Control and Computer Vision**. p. 36-41, 2022.

MA, Chuang; DENG, Minghui; YIN, Yanling. Pig face recognition based on improved YOLOv4 lightweight neural network. **Information Processing in Agriculture**, 2023.

MAGHSOUDI, Omid Haji, *et al.* Open-source *Python* software for analysis of 3d kinematics from quadrupedal animals. **Biomedical Signal Processing and Control**, v. 51, p. 364-373, 2019.

MANJU, Athulya; VALARMATHIE, P. Video analytics for semantic substance extraction using OpenCV in *python*. **Journal of ambient intelligence and humanized computing**, v. 12, p. 4057-4066, 2021.

MARCHANT-FORDE, Jeremy N. *et al.* COVID-19 effects on livestock production: a one welfare issue. **Frontiers in veterinary science**, v. 7, p. 585787, 2020.

MASSOLA, Victor Hugo *et al*. **O CRESCIMENTO DA SUINOCULTURA EM TOLEDO PR.** 13°ENICTEC, 2017.

MENG, Lu *et al.* A survey of object tracking algorithms. **Acta Automatica Sinica**, v. 45, n. 7, p. 1244-1260, 2019.

MUNAPPY, Aiswarya, *et al.* Data management challenges for deep learning. In: **2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. IEEE, p. 140-147, 2019.

NPPC - National Pork Producers Council. **Economic Value of the Pork Industry,** 2021. Disponível em: https://nppc.org/who-we-are/economic-impact/. Acesso em: 24 jun. 2024.

NIR, Oron *et al.* 3D Computer-vision system for automatically estimating heifer height and body mass. **Biosyst. Eng**. 173, 4–10., 2018.

OCEPEK, Marko *et al.* DigiPig: First developments of an automated monitoring system for body, head and tail detection in intensive pig farming**. Agriculture**, v. 12, n. 1, p. 2, 2021.

OECD/FAO – Organization for Economic Co-operation and Development "OECD-FAO Agricultural Outlook", **Meat Consumption**, 2021.

OECD-FAO - Organisation for Economic Co-operation and Development/Food and Agriculture Oraganization of the United Nations). Agricultural Outlook 2019- 2028. **Special Focus: Latin America**. OECD Publishing, Rome, Italy, 2019.

OH, Sang-Hyon; PARK, Hee-Mun; PARK, Jin-Hyun. Estimating vegetation index for outdoor free-range pig production using YOLO. **Journal of Animal Science and Technology**, v. 65, n. 3, p. 638, 2023.

ORHEI, Ciprian *et al*. End-to-end computer vision framework: An open-source platform for research and education. **Sensors**, v. 21, n. 11, p. 3691, 2021.

PANDEY, Santosh, *et al.* Behavioral monitoring tool for pig farmers: ear tag sensors, machine intelligence, and technology adoption roadmap. **Animals**, v. 11, n. 9, p. 2665, 2021.

PARICO, Addie Ira Borja; AHAMED, Tofael. Real time pear fruit detection and counting using YOLOv4 models and deep SORT. **Sensors**, v. 21, n. 14, p. 4803, 2021.

PHAN, Quoc Bao; NGUYEN, Tuy. A Novel Approach for PV Cell Fault Detection using YOLOv8 and Particle Swarm Optimization. **Techrxiv,** 2023.

PIATETSKY, Gregory. *Python* leads the 11 top data science, machine learning platforms: Trends and analysis. **KDnuggets**. 2019.

QIAO, Yongliang, *et al.* Cattle segmentation and contour extraction based on Mask R-CNN for precision livestock farming. **Computers and Electronics in Agriculture**, v. 165, p. 104958, 2019.

QIAO, Yongliang, *et al.* Intelligent perception for cattle monitoring: A review for cattle identification, body condition score evaluation, and weight estimation. **Computers and Electronics in Agriculture**, v. 185, p. 106143, 2021.

QU, Susu, *et al.* EasyFlyTracker: A Simple Video Tracking *Python* Package for Analyzing Adult Drosophila Locomotor and Sleep Activity to Facilitate Revealing the Effect of Psychiatric Drugs. **Frontiers in Behavioral Neuroscience**, v. 15, p. 359, 2022.

RACEWICZ, Przemysław *et al.* Welfare health and productivity in commercial pig herds. **Animals**, v. 11, n. 4, p. 1176, 2021.

ROSLIN, Alexandra *et al.* Processing of micro-CT images of granodiorite rock samples using convolutional neural networks (CNN), **Part II: Semantic segmentation using a 2.5 D CNN. Minerals Engineering**, v. 195, p. 108027, 2023.

ROY, Arunabha M. *et al.* WilDect-YOLO: An efficient and robust computer vision-based accurate object localization model for automated endangered wildlife detection. **Ecological Informatics**, v. 75, p. 101919, 2023.

RUIZ-PONCE, Pablo *et al.* POSEIDON: A Data Augmentation Tool for Small Object Detection Datasets in Maritime Environments. **Sensors**, v. 23, n. 7, p. 3691, 2023.

SANTOS, D. S. et al. Redes bluetooth associadas a redes neurais artificiais para monitoramento de suínos. **Archivos de zootecnia**, v. 65, n. 252, p. 557-563, 2016.

SEO, Jihyun *et al.* A yolo-based separation of touching-pigs for smart pig farm applications. In: **2019 21st International Conference on Advanced Communication Technology** (ICACT). IEEE, p. 395-401, 2019.

SHEN, Weizheng *et al.* Fusion of acoustic and deep features for pig cough sound recognition. **Computers and Electronics in Agriculture**, v. 197, p. 106994, 2022.

SHENK, Justin, *et al.* Traja: A *Python* toolbox for animal trajectory analysis. **Journal of Open Source Software**, v. 6, n. 63, p. 3202, 2021.

SOUZA, Bruno José *et al.* Hybrid-YOLO for classification of insulators defects in transmission lines based on UAV. **International Journal of Electrical Power & Energy Systems**, v. 148, p. 108982, 2023.

SRIDHAR, Vivek Hari *et al.*. Tracker: image-based automated tracking of animal movement and behavior. **Methods in Ecology and Evolution**, v. 10, n. 6, p. 815-820, 2019.

SRINIVAS, Pilla; BHATTACHARYYA, Debnath; MIDHUNCHAKKARAVARTHY, Divya. Prediction of Swine Flu (H1N1) Virus Using Data Mining and Convolutional Neural Network Techniques. **In: Proceedings of the International Conference on Cognitive and Intelligent Computing: ICCIC 2021, Volume 2. Singapore: Springer Nature Singapore**, p. 557-573, 2023.

TERVEN, Juan; CORDOVA-ESPARZA, Diana. A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. **arXiv preprint arXiv:2304.00501**, 2023.

TU, Shuqin *et al.* Automated Behavior Recognition and Tracking of Group-Housed Pigs with an Improved DeepSORT Method. **Agriculture**, v. 12, n. 11, p. 1907, 2022.

ULTRALYTICS. Computer Vision on the Edge. **GitHub**. Disponível em: https://github.com/ultralytics/ultralytics. (Acessado em: 21 de junho de 2023).

USDA - UNITED STATES DEPARTMENT OF AGRICULTURE. Foreign Agricultural Service. **Livestock and Poultry: World Markets and Trade**, 2021. Disponível em: https://apps.fas.usda.gov/psdonline/circulars/livestock_poultry.pdf. Acesso em: 24 jun. 2023.

USDA - UNITED STATES DEPARTMENT OF AGRICULTURE. **Swine reports.** [Online], 2017. Disponível em: https://www.ams.usda.gov/market-news/swine-reports. Acesso em: 24 jun. 2023.

VAN DER ZANDE, Lisette E. *et al.* Individual detection and tracking of group-housed pigs in their home pen using computer vision. **Frontiers in Animal Science**, v. 2, p. 669312, 2021.

WANG, Shunli, *et al.* The Research Progress of Vision-Based Artificial Intelligence in Smart Pig Farming. **Sensors**, v. 22, n. 17, p. 6541, 2022.

WANG, Zhipeng *et al.* Apple stem/calyx real-time recognition using YOLO-v5 algorithm for fruit automatic loading system. **Postharvest Biology and Technology**, v. 185, p. 111808, 2022.

WEI, Zihe; CHANG, Miaosen; ZHONG, Yipeng. Fruit Freshness Detection Based on YOLOv8 and SE attention Mechanism. **Academic Journal of Science and Technology**, v. 6, n. 1, p. 195-197, 2023.

WIDODO, Candraningratri Ekaputri *et al*. Medical image processing using *python* and open cv. In: **Journal of Physics: Conference Series**. IOP Publishing, p. 012003, 2020.

WITTE, Jan-Hendrik; MARX GÓMEZ, Jorge. Introducing a new Workflow for Pig Posture Classification based on a combination of YOLO and EfficientNet. **In: Proceedings of the 55th Hawaii International Conference on System Sciences**. 2022.

XU, Jinyang *et al.* Automatic scoring of postures in grouped pigs using depth image and CNN-SVM. **Computers and Electronics in Agriculture**, v. 194, p. 106746, 2022.

YANG, Quing Ming *et al*. Automatic pig drinking behavior recognition with machine vision. **Transactions of the CSAM**, v. 49, n. 6, p. 232-238, 2018.

YIN, Dan *et al.* Pig Target Detection from Image Based on Improved YOLO V3. In: Advances in Artificial Intelligence and Security: 7th International Conference, ICAIS 2021, Dublin, Ireland, July 19-23, 2021, Proceedings, Part I 7. **Springer International Publishing**, p. 94-104, 2021.

ZAINUDDIN, Zaryanti *et al.* The Waste Detection System of Shrimp Feeding with a Waterproof Camera using Yolo Algorithm. In: **Journal of Physics: Conference Series. IOP Publishing**, p. 012083, 2022.

ZHANG, Mengjie, *et al.* Wearable Internet of Things enabled precision livestock farming in smart farms: A review of technical solutions for precise perception, biocompatibility, and sustainability monitoring. **Journal of Cleaner Production**, v. 312, p. 127712, 2021.

ZHANG, Xiangnan, *et al*. Camshaft tracking method based on correlation probability graph for model pig. **EURASIP Journal on Wireless Communications and Networking**, v. 2020, p. 1-12, 2020.

ZHANG, Yao; CHEN, Zhiyong; WEI, Bohan. A sport athlete object tracking based on deep sort and yolo V4 in case of camera movement. In: **2020 IEEE 6th International Conference on Computer and Communications (ICCC). IEEE**, p. 1312-1316, 2020.

ZHAO, Yunfan; DENG, Xueyuan; LAI, Huahui. A yolo-based method to recognize structural components from 2d drawings. In: **Construction Research Congress 2020: Computer Applications. Reston, VA: American Society of Civil Engineers**, p. 753-762, 2020.

ZHAO, Zuopeng *et al.* PIS-YOLO: Real-Time Detection for Medical Mask Specification in an Edge Device. **Computational Intelligence and Neuroscience**, v. 2022, 2022.

ZHENG, Chan *et al.* Automatic recognition of lactating sow postures from depth images by deep learning detector. **Computers and electronics in agriculture**, v. 147, p. 51-63, 2018.

ZHENG, Zhiyang; LI, Jingwen; QIN, Lifeng. YOLO-BYTE: An efficient multi-object tracking algorithm for automatic monitoring of dairy cows. **Computers and Electronics in Agriculture**, v. 209, p. 107857, 2023.

ZHONG, Zhen. A novel visible and infrared image fusion method based on convolutional neural network for pig-body feature detection. **Multimedia Tools and Applications**, p. 1-19, 2022.

ZHOU, Zhili *et al.* Spatio-temporal feature encoding for traffic accident detection in VANET environment. **IEEE Transactions on Intelligent Transportation Systems**, v. 23, n. 10, p. 19772-19781, 2022.