UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Detecção de animais com risco de extinção utilizando arquiteturas *You Only Look Once* (YOLO) para rodovias inteligentes com suporte a computação de borda

Gabriel Souto Ferrante

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Gabriel Souto Ferrante

Detecção de animais com risco de extinção utilizando arquiteturas *You Only Look Once* (YOLO) para rodovias inteligentes com suporte a computação de borda

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Rodolfo Ipolito Meneguette

USP – São Carlos
Janeiro de 2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

F373d

Ferrante, Gabriel Souto
Detecção de animais com risco de extinção utilizando arquiteturas You Only Look Once (YOLO) para rodovias inteligentes com suporte a computação de borda / Gabriel Souto Ferrante; orientador Rodolfo Ipolito Meneguette. -- São Carlos, 2023.
109 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. Computação de borda. 2. Detecção Animal . 3. Redes Neurais Convolucionais. 4. Visão Computacional . 5. YOLO. I. Meneguette, Rodolfo Ipolito, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Gabriel Souto Ferrante

Detection of endangered animals using You Only Look Once (YOLO) architectures for smart highways with edge computing support

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Rodolfo Ipolito Meneguette

USP – São Carlos
January 2024

Este trabalho é dedicado a comunidade brasileira de pesquisadores ambientais, que lutam diariamente pela existência e manutenção da fauna e flora do Brasil. E no geral, a todos os pesquisadores bolsistas que em tempos difíceis de negação a ciência, sobreviveram com resistência e bravura.

AGRADECIMENTOS

Agradeço em primeiro lugar, a minha família que me apoiou desde sempre nos estudos. Em especial, meus pais, que apesar da baixa escolaridade e grau de instrução, sempre acreditaram que os estudos era o melhor caminho para a formação do individuo. Também agradeço as diversas pessoas que dividiram moradia comigo durante todo este tempo em São Carlos, bem como amigos e conhecidos, que de alguma forma, me motivaram e deram ideias para meu projeto. Aos próximos de república, que apesar das rotinas e áreas de estudos diferentes, trouxeram apoio ao longo desta jornada.

Agradeço a CAPES e ao ICMC, pela oportunidade de bolsa integral e auxílio financeiro para todas as minhas atividades e materiais de pesquisa. Aos professores do programa de pós-graduação, que mesmo nas disciplinas, me trouxeram aprendizados e novos conhecimentos com entusiasmo e alegria. Agradeço também, ao Parque ecológico de São Carlos, que possibilitou o acesso para criação de materiais de mídia sobre os animais abordados nesta pesquisa.

Em adição, também agradeço aos pesquisadores, professores e amigos membros do grupo *Urban Smart Solutions Lab* (USSL), que me auxiliaram com dicas de apresentação, estruturação de artigos e conhecimentos relevantes sobre diversas frentes da computação urbana, que enriqueceram minha caminhada neste mestrado. Agradeço especialmente aos professores Rodolfo I. Meneguette, Luis H. V. Nakamura e Fernando R. H. Andrade que desde o início, trouxeram contribuições significativas para a condução da pesquisa, com boas práticas científicas e condutas éticas. E por último e não menos importante, agradeço aos animais domésticos Madalena, Salem, Patrício, Sindel, Fênix Maria, Naruto e demais outros animais que me acompanharam neste período e que me reconfortaram com suas companhias.

“Que o dinheiro nunca compre sua postura !”
(Sabotage)

RESUMO

FERRANTE, G. S. **Detecção de animais com risco de extinção utilizando arquiteturas *You Only Look Once* (YOLO) para rodovias inteligentes com suporte a computação de borda**. 2024. 109 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

No Brasil, o problema de acidentes rodoviários envolvendo animais é recorrente, tendo um grande número de incidentes em todas as regiões e biomas. Este problema afeta negativamente a vida de espécies silvestres em áreas florestais próximas às pistas. Com a falta de medidas de proteção e monitoramento ou alternativas de passagem segura para os animais, esse problema se intensifica com os animais realizando o cruzamento das vias, gerando risco a suas vidas e a dos condutores. Dado a falta de medidas de proteção ambiental nas estradas, organizações voluntárias e científicas brasileiras criaram alguns sistemas para entender o fenômeno dos acidentes com os animais. No entanto, tais sistemas não são ágeis e nem automáticos, pois não implementam soluções computacionais para o monitoramento inteligente. Este trabalho apresenta uma proposta de elaboração de mecanismos de detecção de animais silvestres com risco de extinção que estão mais envolvidos em acidentes rodoviários no Brasil. Tais mecanismos são especializados para rodovias inteligentes com uso de visão computacional e computação de borda. Assim, a proposta possui etapas fundamentais como a aquisição e criação de um conjunto de imagens especializado para o tema, juntamente com aplicação de técnicas de aumento de dados, treinamento de modelos de detecção de objetos baseados em arquitetura YOLO (YoloV4, Scaled-YoloV4, YoloV5, YoloR, YoloX e YoloV7), testes e validação em dados de imagem e vídeo. Ainda dentro do contexto da proposta, foi realizada uma avaliação de desempenho em ambiente de borda com um recurso computacional limitado. Em conjunto, também é feita uma análise qualitativa sobre os modelos em desafios clássicos de visão computacional. Como conclusão, a execução em tempo-real dos modelos aplicados a dispositivos de computação de borda com baixo poder computacional ainda é um desafio, visto que os modelos mais complexos e robustos em detecção tiveram dificuldades de serem executados, obtendo baixas velocidades de inferência e alto consumo de memória, inviabilizando suas implementações. Já os modelos em suas versões menos complexas e não tão acuradas, permitiram a execução dos experimentos em tempo real. Nenhum modelo utilizado teve desempenho adequado que fosse capaz de superar as problemáticas de detecção dos cenários propostos, demonstrando que tais problemas ainda são desafiadores para as arquiteturas tradicionais do YOLO em geral.

Palavras-chave: Classificação, Computação de borda, Detecção animal, Redes Neurais Convolucionais, Rodovias inteligentes, Visão computacional, YOLO.

ABSTRACT

FERRANTE, G. S. **Detection of endangered animals using You Only Look Once (YOLO) architectures for smart highways with edge computing support**. 2024. 109 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

In Brazil, the problem of road accidents involving animals is recurrent, with a large number of incidents in all regions and biomes. This problem negatively affects the lives of wild species in forest areas close to the roadways and with the lack of protection and monitoring measures or safe passage alternatives for animals, this problem intensifies with animals crossing the roadways, risking their lives and those of the drivers. With the lack of environmental protection measures on the roads, Brazilian voluntary and scientific organizations have created some systems to understand the phenomenon of animal accidents, but these systems are neither agile nor automatic, as they do not implement computer solutions for intelligent monitoring. This dissertation presents a proposal to develop mechanisms for detecting wild animals at risk of extinction that are most often involved in road accidents in Brazil. These mechanisms are specialized for intelligent highways using computer vision and edge computing. Thus, the proposal has fundamental stages such as the acquisition and creation of a specialized image dataset for the topic, along with the application of data augmentation techniques, training of object detection models based on YOLO architecture (YoloV4, Scaled-YoloV4, YoloV5, YoloR, YoloX, and YoloV7), testing and validation on image and video data. Also within the context of the proposal, a performance evaluation was carried out in an edge environment with limited computational resources. A qualitative analysis of the models in classic computer vision challenges is also carried out. In conclusion, the real-time execution of the models applied to edge computing devices with low computing power is still a challenge, since the most complex and robust models in detection had difficulties being executed, obtaining low inference speeds and high memory consumption, making their implementations unfeasible. None of the models used had adequate performance to overcome the problems of detecting the proposed scenarios, demonstrating that these problems are still challenging for traditional YOLO architectures in general.

Keywords: Animal Detection, Classification, Computer Vision, Convolutional Neural Networks, Edge Computing, Smart Highways, YOLO.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
AP	*Average Precision*
API	*Application Programming Interface*
AS	área de sobreposição
AU	área de união
CBEE	Centro Brasileiro de Ecologia em Estradas
CNN	*Convolutional Neural Networks*
CNT	Confederação Nacional do Transporte
COCO	*Common Objects in Context*
DETR	*The Detection Transformer*
FN	falso negativo
FP	falsos positivos
FPS	*Frames per Second*
GPU	*Graphic Processing Unit*
HD	*High Definition*
HOG	*Histograms of Oriented Gradient*
I.A	Inteligência Artificial
ICMbio	Instituto Chico Mendes de Conservação da Biodiversidade
Intel NCS2	Intel Neural Compute Stick 2
IoT	*Internet of Things*
IoU	intersecção sobre união (*Intersection Over Union*)
ITS	Sistemas Inteligentes de Transporte
KNN	*K-Nearest Neighbors*
LAB	*L-star, a-star, b-star*
mAP	média da *Average Precision*
mAP	média da *Average Precision*
MB	megabyte
OpenCV	*Open Source Computer Vision Library*
P	precisão
PCA	*Principal Component Analysis*
R-CNN	Region-based Convolutional Neural Networks

RAM	*random access memory*
RF	*Random Forest*
RGB	*Red, Green, Blue*
SAM	*The Segment Anything Model*
SIFT	*Scale Invert Feature Transform*
SSD	*Single Shot MultiBox Detector*
SVM	*Support Vector Machine*
UFLA	Universidade Federal de Lavras
VN	verdadeiros negativos
VP	verdadeiros positivos
WTB	"*Where's the Bear?*"
XML	*Extensible Markup Language*
YOLO	*You Only Look Once*

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Com a alta demanda de processamento de dados gerados pela sociedade em geral e a evolução da computação, um movimento recorrente é a criação de sistemas para ambientes urbanos que procuram ofertar maior qualidade de vida para as pessoas em diversos aspectos, especialmente para melhorar o tráfego urbano e solucionar suas problemáticas, se baseando nestes dados gerados nas cidades. Nas últimas décadas foi observada uma crescente no uso de dispositivos de computação urbana e um aumento na criação de serviços inteligentes com o intuito de gerenciar o tráfego. Neste contexto, os Sistemas Inteligentes de Transporte (ITS) são projetados para fornecer uma série de novas aplicações (WEI *et al.*, 2011; MENEGUETTE ROBSON A F LOUREIRO, 2019).

Esta categoria de sistemas pode ser implementada e aplicada a diversos contextos urbanos que variam, por exemplo, desde sistemas para alertas de trânsito e rodovia (VOURGIDIS *et al.*, 2020; ALAM; FERREIRA; FONSECA, 2016; HERNANDEZ-JAYO; IGLESIA; PEREZ, 2015), a sistemas que procuram auxiliar no planejamento e criação rotas distribuídas com a sincronização cooperativa entre veículos (AKABANE *et al.*, 2020; KIM; KIM; LEE, 2020; AKABANE *et al.*, 2018; AN; JUNG, 2018; AKABANE *et al.*, 2018). Também existem sistemas que visam fornecer mecanismos de distribuição de imagens e vídeos urbanos (PADILLA *et al.*, 2019; IMMICH; CERQUEIRA; CURADO, 2019; GERAT *et al.*, 2017; IMMICH; CERQUEIRA; CURADO, 2016; QUADROS *et al.*, 2012), serviços inteligentes de consultoria estratégica para análises detalhadas (HUSSAIN; JAHROMI; CETIN, 2020; SIMCHON; RABINOVICI, 2020; WANG *et al.*, 2019), para recursos de condução e comunicação de veículos autônomos (WANG *et al.*, 2019; VIANA; AOUF, 2018; CHEN *et al.*, 2015) e entre outras demandas sociais que permitam de alguma forma auxiliar entidades governamentais sobre possíveis problemas.

Apesar de os ITSs estarem ganhando cada vez mais espaço nos ambientes de transportes, especialmente nos países desenvolvidos, no Brasil, o uso desses sistemas ainda não é uma realidade. Há altos índices de problemas com trânsito destacados em relatórios de acidentes

nas rodovias federais do país. A Confederação Nacional do Transporte (CNT) vem criando, desde 2007, relatórios de monitoramento dos acidentes em rodovias federais para auxiliar em análises da situação brasileira. Em seu relatório de 2020 (CNT, 2021), foi constatado 63.447 acidentes, sendo que os acidentes por colisões frontais são os mais frequentes, com 59,4% dos casos, seguidos pelos casos envolvendo saída da pista, capotamento e atropelamentos, com 15,7%, 12,2% e 7,1% respectivamente. Nesta contagem de atropelamentos, não é especificado se são contabilizados somente atropelamentos que envolvam humanos ou que se constam também acidentes envolvendo animais, tendo somente uma análise superficial e geral. Essa falta de clareza é também identificada e apontada por ecologistas e pesquisadores da área de ecologia em estradas, como explicado por Cymbaluk (2018) em uma matéria especial de jornal que apresenta informações sobre as mortes de animais em rodovias brasileiras. Assim, sendo este um dos principais problemas em aberto sobre o cenário rodoviário. Conforme a reportagem, estima-se que 475 milhões de animais silvestres são mortos em estradas brasileiras, segundo o Centro Brasileiro de Ecologia em Estradas (CBEE) da Universidade Federal de Lavras (UFLA). Dos 475 milhões acidentes anuais apontados, 90% são acidentes que envolvem animais de pequeno porte, 8% são acidentes com animais de médio porte e apenas 2% envolvem espécies grandes.

A problemática ocorre em todos os biomas brasileiros e em meio a este cenário, o governo brasileiro emprega poucos recursos para amenizar o problema dos acidentes, que apesar de existir uma legislação de licenciamento ambiental federal[1], a maioria das rodovias não são adequadas para a segurança animal, possuindo poucas pontes de fauna, travessias subterrâneas ou grades laterais para evitar a entrada dos animais na estrada (MARQUES, 2020). Algumas concessionárias permitem a ligação gratuita para reportar animais acidentados ou não, mas esse serviço é pouco divulgado e, portanto, não é aproveitado de forma eficiente (NACIONAL, 2023). Com poucos investimentos para a conservação, iniciativas como a do U-SAFE[2] e o Sistema Urubu[3], desenvolvidos pelo CBEE, permitem um melhor entendimento dos acidentes com animais nas estradas, onde os pesquisadores possam realizar suas estimativas, baseadas nos relatos feitos por condutores que encontram os animais na pista e as imagens sejam avaliadas e classificadas por especialistas e biólogos.

Analisando os dados do sistema fornecido pelo CBEE, nos últimos cinco anos (2018, 2019, 2020, 2021 e 2022) os animais mamíferos com grau de vulnerabilidade ou perigo estão em destaque nos índices de atropelamentos. Conforme os dados abertos disponibilizados no site

[1] O documento do licenciamento ambiental federal para rodovias e ferrovias feito em 2020 pode ser acessado pela página (URL): <https://www.gov.br/infraestrutura/pt-br/assuntos/sustentabilidade/arquivos-sustentabilidade/manual_laf-1308-web.pdf>

[2] O U-Safe é um aplicativo desenvolvido pela Universidade Federal de Lavras para condutores se alertarem a acidentes em geral nas estradas, incluindo animais. O aplicativo informa quilômetro a quilômetro o risco do trecho, permitindo que o condutor possa redobrar a atenção e reduzir a velocidade quando necessário. Site oficial <https://sousafe.com.br/conheca-o-aplicativo/>

[3] Principal sistema brasileiro para reportar acidentes de trânsito envolvendo animais em rodovias, pode ser acessado pela (URL): <https://sistemaurubu.com.br>

oficial, existem 14 espécies de mamíferos identificados nos registros de atropelamento neste período, como pode ser observado na Figura 1. Seguindo a Tabela 1, depois do ano de 2020, os atropelamentos aumentaram consideravelmente, com algumas espécies tendo a quantidade de registros de atropelamento na plataforma triplicados em 2022 quando comparado a 2021, e até mesmo espécies de risco novas no laudo nestes anos. No geral, as espécies mais afetadas no período foram as: do Tamanduá-Bandeira, Raposa-do-campo, Anta e Lobo-Guará. Estes animais comumente aparecem nas estimativas, indicando que não houve redução dos acidentes com estes animais, tão pouco a implementações de novas soluções rodoviárias em trechos de fauna ativa. Pode-se atribuir, também, que 2020 foi o ano que não teve dados, devido ao isolamento social e restrição do transporte em rodovias, que dificultou a coleta e a criação de estimativas, uma vez que o sistema depende dos condutores reportarem os casos.

Figura 1 – Mapeamento nacional dos registros de atropelamento de animais do sistema Urubu nos anos de 2018 a 2022 para animais com grais de ameaça, de vulnerabilidade ou de perigo.

Fonte: CBEE (2023).

A maioria destes animais são medianos e grandes, pertencentes ao grupo de animais com risco de extinção, conforme é informado pelo livro vermelho (ICMBIO, 2018) do Instituto Chico Mendes de Conservação da Biodiversidade (ICMbio)[4]. Por ser o único órgão a divulgar estatísticas dos acidentes em tempo real, o sistema do CBEE é utilizado como base para o governo federal fazer suas análises, devido a uma falta de dados oficiais sobre o tema de acidentes com animais.

A falta de melhorias estruturais nas estradas não é o único problema. Existe uma lacuna na criação e uso de tecnologias computacionais voltadas para o monitoramento das vias brasileiras,

[4] Principal órgão do governo brasileiro sobre estudos ecológicos e de conservação, pode ser acessado pelo site (URL): <https://www.icmbio.gov.br/portal/>

Tabela 1 – Frequência de atropelamentos registrados pelo sistema Urubu nos anos de 2018 a 2022 por espécies.

Espécie	2018	2019	2020	2021	2022	Total
Tamanduá-Bandeira (*Myrmecophaga tridactyla*)	5	22		28	84	139
Raposa-do-campo (*Lycalopex vetulus*)	4			8	14	26
Anta (*Tapirus terrestris*)		3		4	17	24
Lobo-Guará (*Chrysocyon brachyurus*)	1	6	3	6	5	21
Gato-do-mato-pequeno (*Leopardus guttulus*)		1		1	7	9
Gato-do-mato (*Leopardus tigrinus*)		1		1	2	4
Onça-Parda (*Puma concolor*)		2		1	1	4
Gato-maracajá (*Leopardus wiedii*)		1			2	3
Gato-do-mato-grande (*Leopardus geoffroyi*)				1	1	2
Queixada (*Tayassu pecari*)		2				2
Bugio-de-mãos-ruivas (*Alouatta belzebul*)					1	1
Sagui-caveirinha (*Callithrix aurita*)	1					1
Onça-pintada (*Panthera onca*)					1	1
Tatu-canastra (*Priodontes maximus*)					1	1
Total	11	38	3	50	136	238

Fonte: CBEE (2023).

visto os poucos investimentos do governo em pesquisa e a quase nula existência de um sistema computacional para o tema. Neste contexto, o campo de pesquisa de detecção e classificação animal vem ganhando espaço, demonstrando que o uso de tecnologias computacionais de visão computacional com auxílio de redes neurais especializadas e processamento de imagem estão tendo resultados positivos e precisos, permitindo a criação de sistemas eficazes, com objetivos de melhorar a conservação e até mesmo a criação de mecanismos para detectar animais em pistas ou parques ecológicos (FERRANTE *et al.*, 2021).

1.1 Objetivos

Considerando os problemas citados anteriormente sobre os acidentes com animais nas estradas e a falta de implementações de sistemas computacionais para este tema, este trabalho tem por objetivo principal modelar, implementar e fornecer mecanismos de detecção e classificação de animais baseado no paradigma de redes neurais convolucionais com arquitetura *You Only Look Once* (YOLO). Para atingir esse objetivo se faz necessário a criação de um banco de dados especializado sobre os animais que mais sofrem acidentes, com enfase em animais de médio e grande porte em extinção, visando viabilizar a construção de sistemas inteligentes de detecção para ambientes rodoviários. Os modelos de detecção escolhidos são: YoloV4, Scaled-YoloV4, YoloV5, YoloR, YoloX e YoloV7 para sistemas inteligentes de monitoramento de animais em rodovias. Além disso, os modelos são aplicados em dispositivos de computação de borda. Dessa forma, os objetivos específicos elencados são:

1. Criar e organizar um conjunto de dados de imagens com animais de médio e grande porte que mais sofrem acidentes nas estradas brasileiras.

2. Realizar a implementação dos detectores propostos, utilizando aprendizado por transferência, e especializando-os para as classes de animais utilizados no conjunto de dados de imagem proposto sem e com a utilização de técnicas de aumento de dados.

3. Avaliar os detectores por métricas da área de detecção, com uso de dados reais e de validação do conjunto de dados proposto, considerando os limites de um dispositivo de computação de borda.

4. Avaliar os detectores sobre situações reais com problemas clássicos de visão computacional, como oclusão de ambiente, objetos pequenos e com vídeos com pouca qualidade de imagem.

5. Comparar os resultados com outros trabalhos relacionados e apresentar suas principais diferenças. Além disso, pontuar as oportunidades em aberto na área de detecção animal no Brasil.

1.1.1 Questões de pesquisa e hipóteses

- A principal questão de pesquisa para o trabalho é se os modelos de detecção baseado em YOLO podem realizar detecções e classificações de animais precisas, em cenários comuns e desfavoráveis, sobre um dispositivo de computação de borda com velocidade, que permita ser subsídio de uma aplicação em tempo real para ambientes rodoviários inteligentes? Como hipótese, acredita-se que os modelos permitam, com o treinamento sobre o conjunto de dados proposto, realizar tais tarefas em ambientes com recursos reduzidos.

- Outra questão de pesquisa é verificar a viabilidade da metodologia de criação de conjunto de imagens utilizando mecanismo de busca de imagens simples pode criar uma base de dados que não gere problemas de treinamento para criação de modelos de detecção de objetos. Como hipótese, acredita-se que a metodologia possibilite a criação rápida e eficaz de base de dados que permite modelos de detecção serem treinados e que garante a não ocorrência de problemas de sobre-ajuste.

- Em adição, também há como questão de pesquisa, a verificação se versões das arquiteturas atuais com maiores complexidade possam ser executadas com desempenho semelhante a versões enxutas com pesos leves nos dispositivos de borda. Como hipótese, acredita-se que possam ser utilizados para inferência veloz de forma que demonstre a viabilidade no uso de tais pesos para o cenário com dispositivos limitantes, no qual demonstre a evolução computacional de cada versão do YOLO.

1.2 Estrutura do Documento

A estrutura desta dissertação de mestrado segue a seguinte ordem: O Capítulo 2 visa informar sobre os principais conceitos em torno do tema do projeto. O Capítulo 3 pretende listar alguns trabalhos semelhantes e suas relações com a proposta deste trabalho. O Capítulo 4 traz a proposta com mais detalhes, apresentando o fluxo de trabalho e todas as suas etapas. O Capítulo 5 apresenta as formas de avaliação dos métodos propostos, bem como a visão geral de testes. O Capítulo 6 exibe os resultados alcançados perante os testes e análises e, por fim, o Capítulo 7 apresenta as conclusões em frente as hipóteses estipuladas, além de trazer informações referente a trabalhos futuros.

CAPÍTULO

2

FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os principais conceitos computacionais para entendimento da proposta deste trabalho. É explicado uma breve introdução à área de inteligência artificial e suas aplicações, no mesmo tópico é aprofundado o conceito de redes neurais. Na sequencia, os conceitos de visão computacional são apresentados, com ênfase na área de detecção de objetos. Por fim, este capítulo é finalizado com os principais conceitos de computação de borda.

2.1 Inteligência artificial

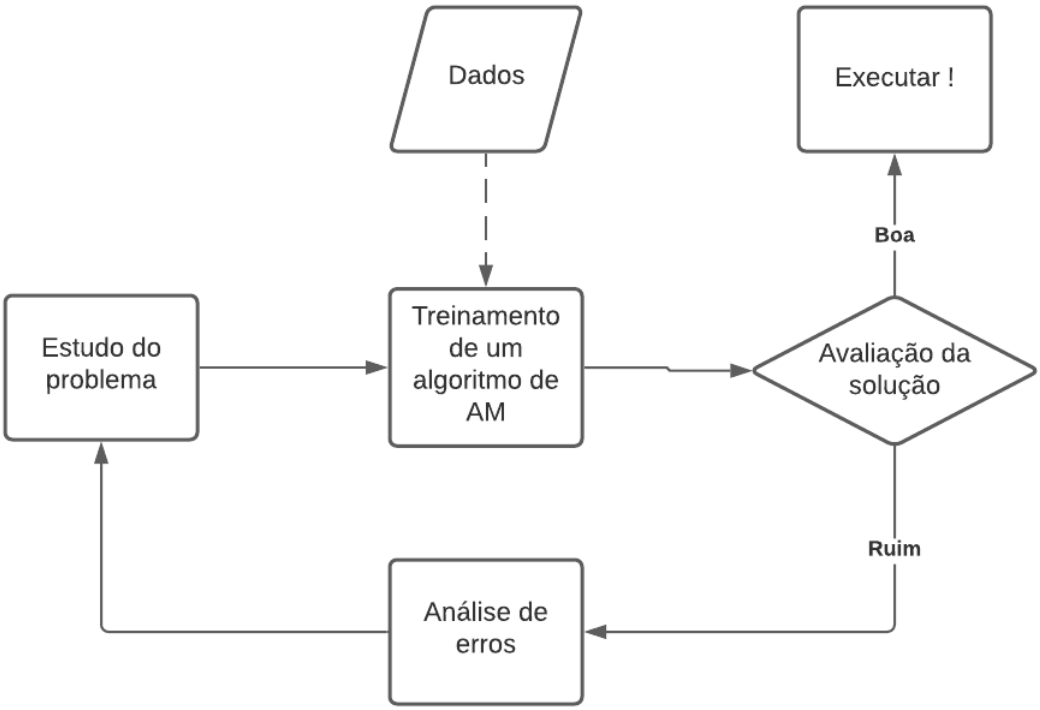
A Inteligência Artificial (I.A) é uma área da computação que procura atribuir aos sistemas computacionais a capacidade de realizar tarefas iguais aos dos seres humanos, com uso de tecnologias que permitem simular as funções físicas e cognitivas humanas. Em uma descrição formal, Rich e Knight (1994) define como sendo: *"Inteligência artificial é o estudo de como fazer os computadores realizarem tarefas as quais, até o momento, os homens fazem melhor"*. A I.A herdou da ciência cognitiva a ideia de que um humano constrói seu conhecimento sobre o mundo com um modelo mental, capaz de identificar padrões e relacionar com objetos (ROSA, 2011). Essa ideia permeia até os dias atuais, nas mais diversas aplicações que envolve a I.A, a cada trabalho na área, na maioria das vezes, é apresentado um modelo construído e aplicável.

Duas principais sub-áreas da I.A se tornaram destaque no campo da pesquisa devido a resoluções de problemas complexos de forma inteligente e sem intervenção humana, refere-se ao Aprendizado de Máquina (AM) e o aprendizado profundo (AP). O aprendizado de máquina é definido por Mitchell (1997) *"A capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência"*. Geralmente é o processo de indução de uma hipótese conforme a experiência passada e tem uma forte base estatística, matemática e teoria da computação. Os sistemas de AM devem ter a característica de generalização de hipóteses para novos dados sobre determinado problema/situação, que por suas vezes podem se tratar de dados estruturados ou

não (imagens, áudios, textos).

Dentre a área de AM, os algoritmos são categorizados como supervisionados, semi-supervisionados e não-supervisionados. São supervisionados aqueles que permitem o conhecimento o valor do atributo de saída de cada nova instância de dado de acordo com seus atributos de entrada. Os não-supervisionados não possuem essa característica, sendo mais atribuídos para tarefas descritivas de dados (agrupamento e sumarização de dados, por exemplo) diferentemente dos algoritmos supervisionados, empregados para tarefas preditivas (problemas de classificação e regressão, por exemplo), e os semi-supervisionados seriam a mistura destas duas abordagens (FACELI, 2011). A Figura 2 exibe a abordagem tradicional para a criação de um sistema com aprendizado de máquina onde, conforme o problema e os dados, o treinamento de um algoritmo de AM é realizado e avaliado. Caso tenha exito na tarefa em questão, o algoritmo é executado para novos dados, caso não seja, é feita uma análise dos erros e refeito o estudo do problema.

Figura 2 – Fluxograma de uma abordagem tradicional de aprendizado de máquina (Adaptado)

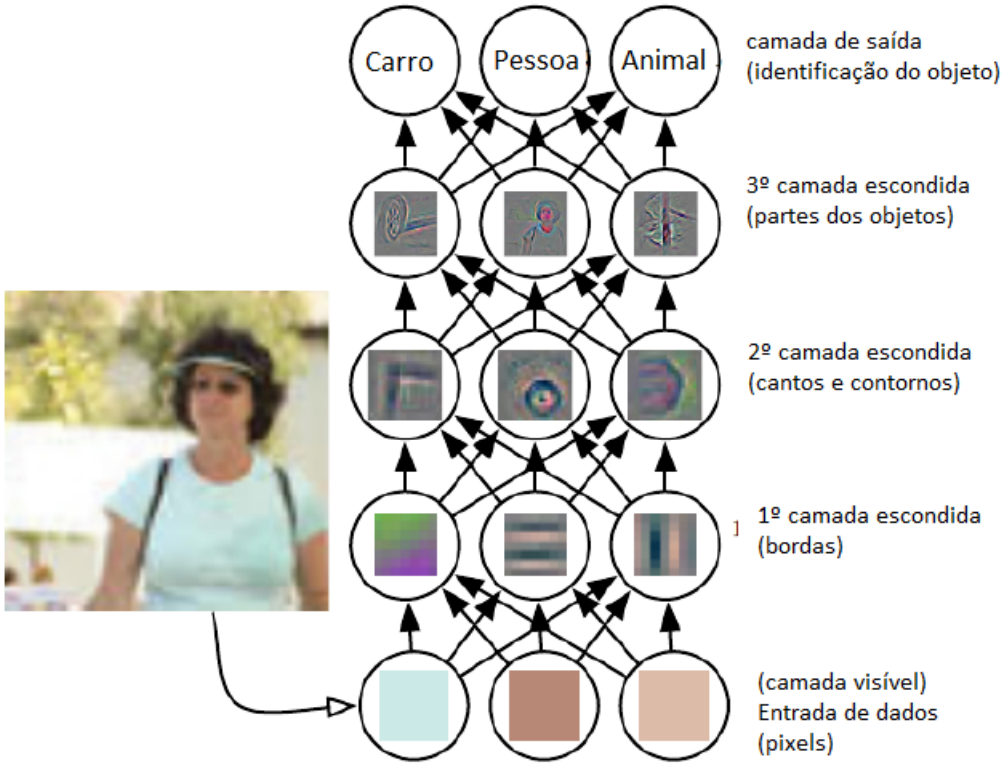


Fonte: Geron (2017).

De acordo com Goodfellow, Bengio e Courville (2016) o aprendizado profundo é uma sub-área do aprendizado de máquina, onde é pretendido dar a característica de aprender sem a intervenção humana ao computador, através da hierarquia de conceitos. Esta hierarquia parte de conceitos simples sobre determinado evento ou elemento e pela combinação destes, conceitos mais complexos são aprendidos. A Figura 3 apresenta um exemplo da hierarquia de conceitos com uma imagem. Na camada visível é a camada que recebe os dados de entrada. Nas camadas escondidas, é extraído recursos cada vez mais abstratos da imagem. Essas camadas são chamadas

"escondidas" porque seus valores não são dados que conhecemos vindos da entrada, em vez disso, o modelo deve determinar quais conceitos são úteis para explicar as relações nos dados observados. Na primeira camada escondida do exemplo, o modelo identifica arestas através de técnicas de processamento de imagem, estas arestas são subsídios para a camada posterior, que servem para detectar cantos e contornos que também serão utilizados na camada seguinte, para determinar partes inteiras de objetos, fazendo o mapeamento das características. Ao final, a camada de saída informará qual objeto a imagem contém devido às partes inteiras descobertas.

Figura 3 – Exemplo das camadas de hierarquia de conceitos de um modelo de aprendizado profundo sobre uma imagem (Adaptado)



Fonte: Goodfellow, Bengio e Courville (2016).

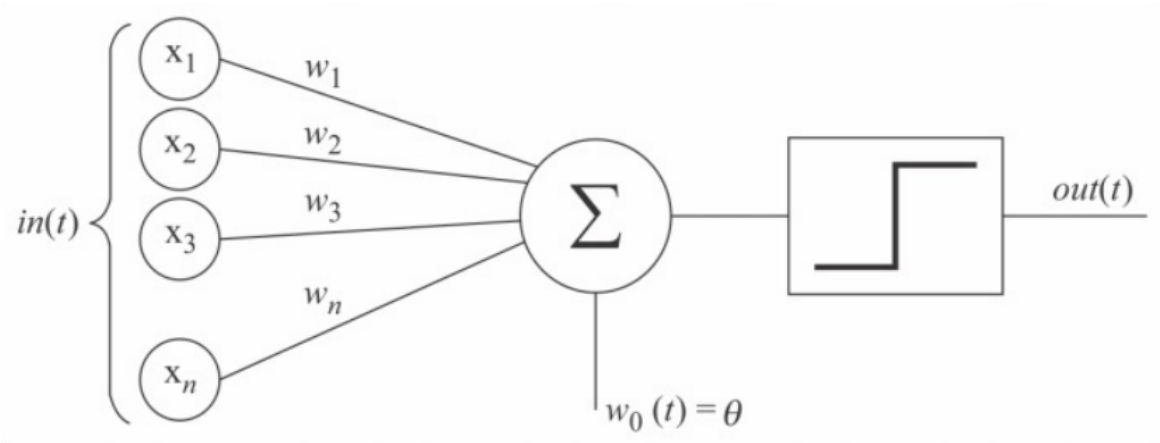
Assim como o aprendizado de máquina, o aprendizado profundo tem uma forte base na estatística e na matemática aplicada, como álgebra linear, probabilidade e computação numérica. Atualmente, aplicações para inferência de predições, recomendação e visão computacional estão em destaque no mercado e na pesquisa, fomentando cada vez mais trabalhos em diversos problemas.

2.1.1 Redes Neurais Artificiais

Os autores Minichino e Howse (2015) definem que as redes neurais artificiais são modelos estatísticos formados por pares de elementos em um determinado conjunto e uma função que deve gerar os dados semelhantes aos do conjunto. O objetivo de uma rede neural é simplificar uma realidade complexa de forma que se deduz uma função para representar

observações estatísticas que se esperaria dessa realidade. Para tal, as redes neurais utilizam de neurônios que são unidades capazes de realizar uma aproximação com uma função simples as funções que criaram as entradas na rede, que geralmente são complexas. Além disso, uma rede é geralmente caracterizada como neural se os neurônios conseguem aproximar uma função não linear. Assim como demonstrado na Figura 3, os neurônios são cada nós nas camadas escondidas. O tipo mais simples de rede é a chamada perceptron (Figura 4), sendo uma função (neurônio) que recebe várias entradas (*Xn*) e gera um único valor (*out(t)*). Cada uma das entradas tem um peso (*w*) associado que significa a importância da entrada, na Figura 3 utiliza-se uma rede multi-perceptron.

Figura 4 – Exemplo de uma rede mono-perceptron com uma função Sigmoid



Fonte: Minichino e Howse (2015).

Em uma rede multi-perceptron, os neurônios estão interconectados entre si, e o conjunto de pesos de cada neurônio define a força da conexão com outros neurônios. Esses pesos são adaptativos, isto significa que eles mudam com o tempo de execução de um algoritmo de aprendizado e com uma estratégia de retro propagação (algoritmos que afetam os pesos com base em erros de classificação). Geralmente, uma função Sigmoid indica que a função produz um valor 0 ou 1. O discriminante é um valor limite e se a soma ponderada das entradas for maior que um determinado limiar, o perceptron produz uma classificação binária.

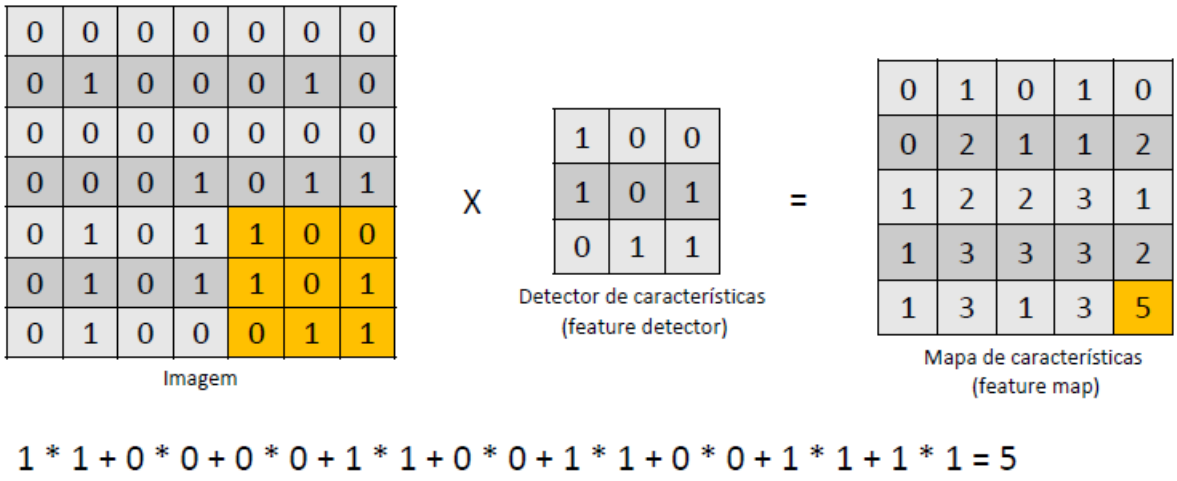
2.1.2 Redes Neurais Convolucionais

Uma rede neural convolucional usufrui de uma rede neural com filtragem de neurônios, com o acréscimo de uma camada próxima à camada de entrada, chamada de camada de convolução. Essa camada permite extrair dos dados as principais características, que servirão de subsídio para as camadas totalmente conectadas (rede neural convencional) posteriores. Este tipo de rede permite que o problema de sobre-ajuste (*overfitting*) seja mitigado. Além disso, pela redução da conectividade dos neurônios da rede, o tempo de execução também diminui (SINGH, 2019). Este tipo de rede tem grande valor na construção de aplicações de visão computacional, em

tarefas de classificação e detecção de objetos pela sua rapidez e em sistemas de reconhecimento de padrões.

De acordo com Pujari (2018) o processo geral de uma convolução tem três etapas fundamentais: Etapa de operação de convolução, *Pooling* e *Flattening*. A operação de convolução é executada com um uso de um *kernel*. Este *kernel* é uma matriz que se aplica as entradas da rede. No caso de uma entrada de uma imagem (2D), a matriz também deve ser de duas dimensões. O *kernel* pretende coletar as principais características da imagem realizando uma multiplicação com a entrada, assim criando mapas de características. O *kernel* vai "deslizando"sobre partes da matriz de uma imagem e realizando as multiplicações. Após multiplicar determinada parte da imagem, é feito a soma das multiplicações e atribuído o valor da soma no mapa de características. Esse mapa de características tem uma menor dimensionalidade. O processo pode ser visto na Figura 5. Uma rede convolucional normalmente apresenta várias camadas de convolução, portanto é tido diversos mapas de características feitos por diversos *kernels*, o conjunto destes mapas são chamados filtros de características (GANESH, 2019).

Figura 5 – Exemplo da aplicação de um kernel sobre uma imagem (Adaptado)

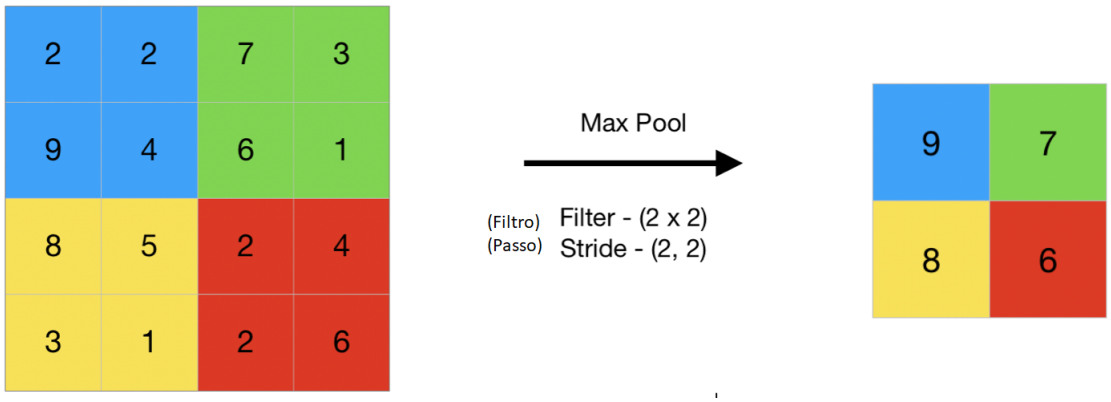


Fonte: Expert (2021).

A segunda etapa é a de *Pooling*. Esta etapa usa os mapas de características criados nas operações de convolução para realizar mais uma mudança de dimensionalidade. Com o *Pooling*, nos mitigamos o problema dos mapas de características de detecção das mesmas características mais relevantes na imagem que contém o mesmo objeto ou elemento de interesse, porém com ruídos ou com variações (rotação, tonalidade, etc.). A Figura 7 apresenta um exemplo da aplicação do *Pooling* com um filtro de duas dimensões e com uma janela deslizante (passo) de dois em dois valores do mapa. Para cada passo, é armazenado somente o valor mais alto, esta estratégia se chama *Max-Pooling*.

Ao final da redução de dimensionalidade é feita a etapa de *Flatterning*. Esta etapa é responsável pela conversão da imagem resultado do *Max pooling* em uma estrutura de dados

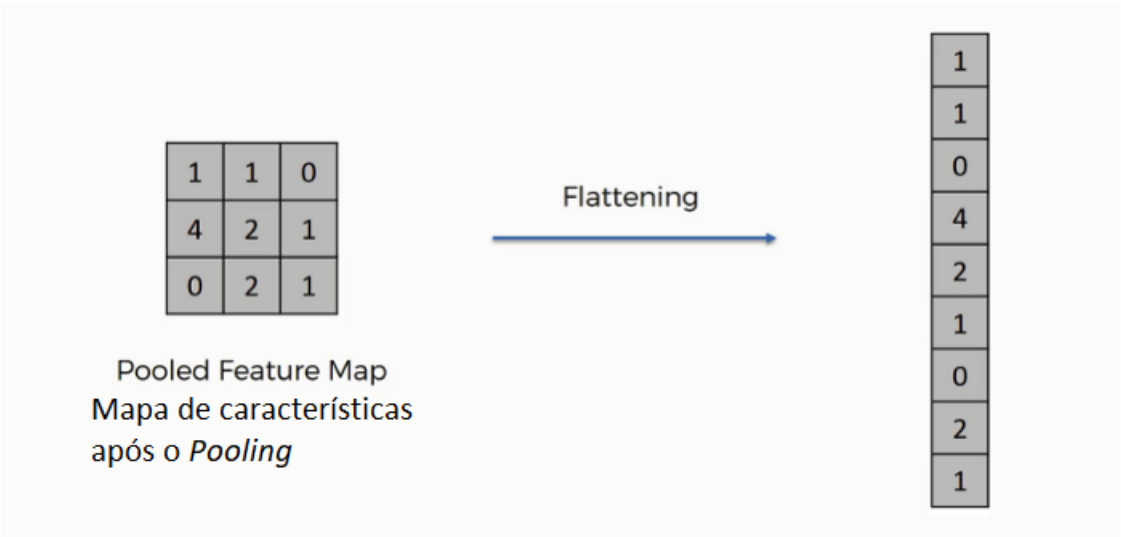
Figura 6 – Exemplo da aplicação de um *Pooling* em um mapa de características (Adaptado)



Fonte: GeeksforGeeks (2019).

simples, geralmente uma lista unidimensional para servir de entrada para camadas totalmente conectadas da rede.

Figura 7 – *Flatterning* da imagem para uma lista unidimensional (Adaptado)



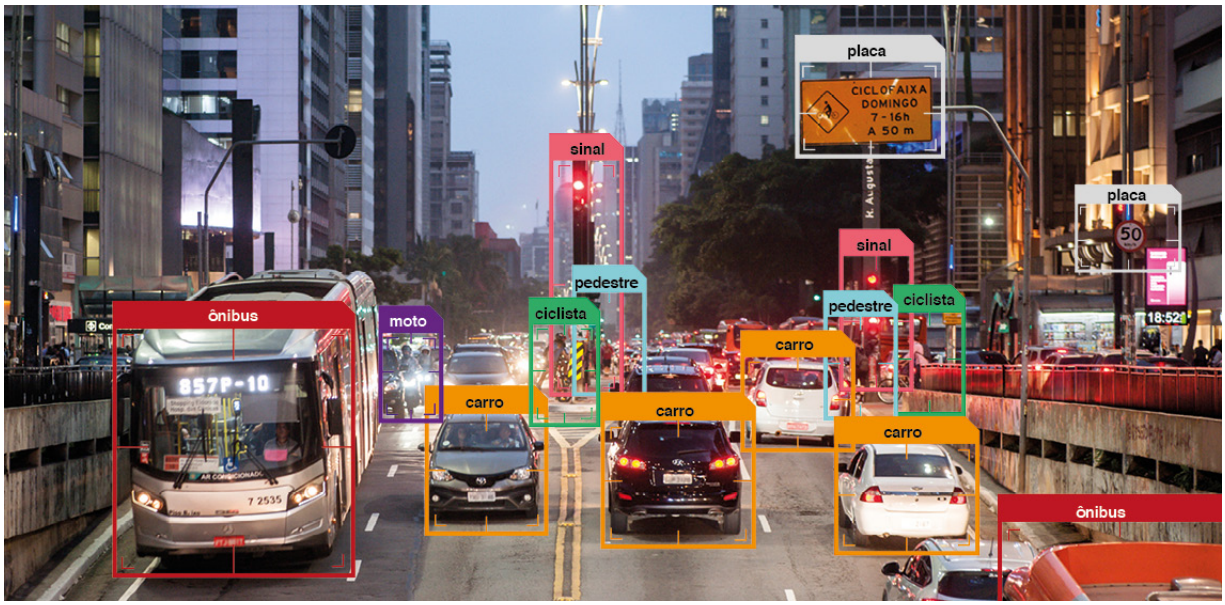
Fonte: Clappis (2019).

2.2 Visão Computacional

O campo de estudo da visão computacional está crescendo rapidamente e muitos *frameworks*, ferramentas e bibliotecas já foram desenvolvidos. A visão computacional tem sido muito requisitada para automatizar as tarefas diárias com dispositivos ou máquinas (ÖZKAYA; YILLIKCI, 2015). As técnicas de visão computacional podem ser aplicadas em muito contextos, alguns exemplos são nas construções de carros autônomos (PADILLA *et al.*, 2019) (Figura 8, no melhoramento de diagnósticos de doenças na medicina, na detecção de descobertas arqueológicas, na biologia para análise de imagens obtidas de microscópios, na detecção de objetos e seres

vivos (FILHO; NETO, 1999). Com este crescimento e popularidade, grandes empresas no ramo de computação em nuvem também apresentaram no mercado seus serviços de visão computacional através de *Application Interface Programming* (API), dentre eles, APIs de reconhecimento de objetos, de face e emoções (KHANAL *et al.*, 2018; AL-OMAIR; HUANG, 2018).

Figura 8 – Representação artística de um sistema de reconhecimento usado por um carro autônomo



Fonte: Andrade (2019).

A visão computacional pode simular a visão humana através de recursos de *software* e *hardware* que são usados para analisar e processar informações e dados visuais. Isso inclui os processos de aquisição, pré-processamento, segmentação, normalização das características de interesse, classificação e reconhecimento de padrões, à resolução do problema. O funcionamento de um sistema de visão computacional possui uma estrutura sequencial das etapas se divide em três estágios: (1) visão em baixo nível (aquisição e pré-processamento), (2) visão ao nível intermediário (segmentação, extração e normalização de características) e (3) visão em alto nível (classificação, reconhecimento e correspondência) como descrito por Forsyth e Ponce (2003). A partir da imagem ou sequência de imagens, o conhecimento e a compreensão do mundo externo é adquirida e as informações relevantes do objeto são coletadas, segundo Lemley, Bazrafkan e Corcoran (2017) e Li e Shi (2018).

2.2.1 Detecção de objetos

A detecção de objetos é uma das tarefas no campo de visão computacional que procura identificar objetos de interesse em uma imagem ou vídeo. Normalmente a tarefa de detecção também inclui a tarefa de classificação dos objetos (Bogusław Cyganek, 2013). Diversos são os desafios para a detecção, pois podem existir alguns fatores e cenários onde a detecção pode ser comprometida, como, por exemplo, oclusões de ambiente, pouca ou muita iluminação,

baixa resolução, cenários noturnos e até a falta dados de amostra. Existem diversas abordagens para a detecção de objetos já aplicadas na literatura, como o uso de *Histograms of Oriented Gradient* (HOG), *Support Vector Machine* (SVM), *Scale Invert Feature Transform* (SIFT), *Principal Component Analysis* (PCA). Atualmente, o estado da arte na área de detecção de objetos é com uso de redes neurais convolucionais (*Convolutional Neural Networks* (CNN)) (FERRANTE *et al.*, 2021). Devido às camadas de convolução destas redes, foi possível obter performance que permite aplicações em tempo real com uma alta precisão. Com o sucesso de seu uso, muitas arquiteturas de redes convolucionais foram criadas, algumas para propósitos gerais como a AlexNet, VGGNet, GoogLeNet e ResNet (ROSEBROCK, 2017) e outras para melhorar especificamente a tarefa de detecção e de classificação, como abordagens Region-based Convolutional Neural Networks (R-CNN) (GIRSHICK *et al.*, 2014), *Fast* R-CNN (GIRSHICK, 2015), *Faster* R-CNN (REN *et al.*, 2016), *You Only Look Once* (YOLO) (REDMON *et al.*, 2016), *The Detection Transformer* (DETR) (CARION *et al.*, 2020), *eformable DETR* (ZHU *et al.*, 2020) e DINO (ZHANG *et al.*, 2022). Dentre as arquiteturas, se destacam *Single Shot MultiBox Detector* (SSD) (LIU *et al.*, 2015), *The Segment Anything Model* (SAM) (OSCO *et al.*, 2023) e YOLO, como o estado-da-arte para detecções. Estas possuem a estratégia de *single-shot*, ou seja, utilizar a entrada somente uma única vez para as detecções, isso permitiu grandes ganhos em desempenho.

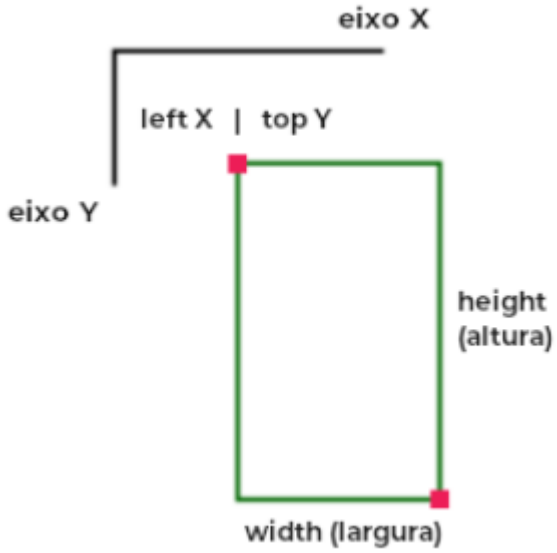
Em uma tarefa de detecção, geralmente nos queremos prever para cada imagem, a localização do objeto no espaço e qual sua classe correspondente. Todas as redes neurais convolucionais para este proposito, usufruem de suas camadas de convoluções para extrair características relevantes para realizar a localização e a classificação. A localização de um objeto é dado pelas caixas delimitadores. A caixa delimitadora (Figura 9) possui quatro variáveis necessárias para ser desenhada sobre a imagem. De início, é necessário ter a coordenada mais à esquerda do objeto no eixo X e a coordenada mais ao topo do objeto no eixo Y para formar o ponto inicial da caixa delimitadora. A partir dele, basta saber as outras duas variáveis, altura e largura, para formar a caixa ao entorno do objeto.

Como todo processo de aprendizado de máquina, uma rede neural também necessita de um treinamento para efetuar as detecções em novos dados. Para isso, o conjunto de imagens de treinamento em questão deve estar rotulado, ou seja, com as devidas caixas delimitadoras corretas para que o modelo possa aprender e já identificar as características dos objetos de interesse.

2.2.2 Arquiteturas You Only Look Once (YOLO)

Em especial na arquitetura YOLO, foi criado por Redmon *et al.* (2016) e atualmente está em sua quarta versão, sendo o detector mais rápido para aplicações em tempo real. A detecção é feita através de uma única passada pela imagem. O processo geral de detecção pode ser observado na Figura 10. A imagem é dividida em diversas grades de dimensões 13x13 (para versões mais atuais, está sendo utilizado 19x19). Para cada uma dessas grades, é detectado cinco

Figura 9 – Formação de uma caixa delimitadora



Fonte: Expert (2021).

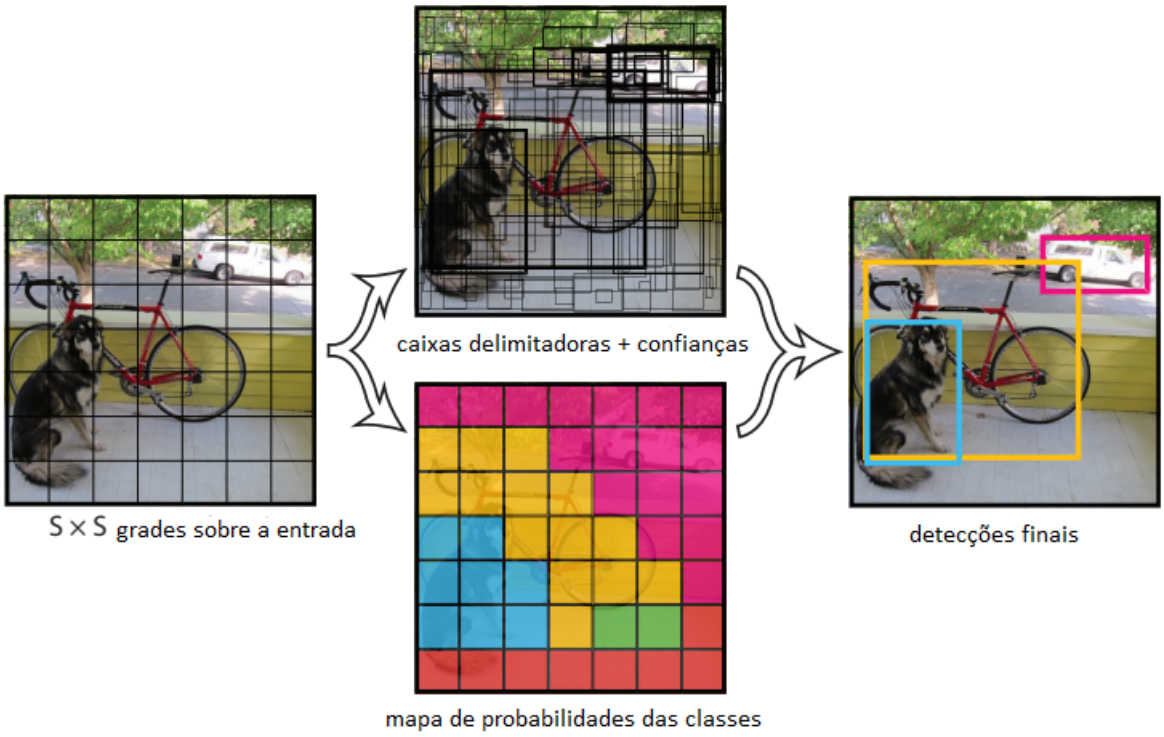
caixas delimitadoras sobre quaisquer objetos que tenham identificado. O modelo YOLO cria para cada caixa detectada nas grades, o seu porcentual de confiança de que há um objeto ali e uma porcentagem para cada classe do modelo para o objeto. Estes dados de porcentagens são somados e no final é deduzido qual classe tem mais probabilidade de ser. A maioria das caixas detectadas possuem um baixo grau de confiança, portanto, sempre é definido um limite (*threshold)* para descartar estas detecções e realizar junções de caixas delimitadoras compartilhadas de um mesmo objeto (conceito de *non-max supression*).

Outro conceito importante na arquitetura YOLO é o conceito de âncoras. As âncoras são retângulos de proporções pré-definidas usados para ter maior correspondência entre as caixas delimitadoras previstas e as esperadas (EXPERT, 2021). Elas possuem tamanhos próximos aos tamanhos das caixas dos objetos e serão redimensionadas para o tamanho do objeto usando algumas saídas da rede neural.

2.2.2.1 YoloV4 e YoloV5

Na quarta versão do YOLO (YOLOv4) feita por Bochkovskiy, Wang e Liao (2020) a arquitetura (Figura 11) é otimizada para maior precisão e acurácia. Os detectores em geral podem pertencer a dois níveis de abstração, o mais superficial chamada detecção em dois estágios são detectores que realizam as tarefas de detecção e classificação separadamente e outra chamada de detecção de um estágio não ocorre este desacoplamento das tarefas, sendo a abordagem do YOLOv4. A camada de entrada recebe uma imagem, que logo entra na camada *Backbone*, onde existe uma rede neural convolucional pré-treinada pelo *dataset* ImageNet, ela tem por objetivo de extrair as principais características da imagem. A rede neural neste caso é a CSPDarknet53 e

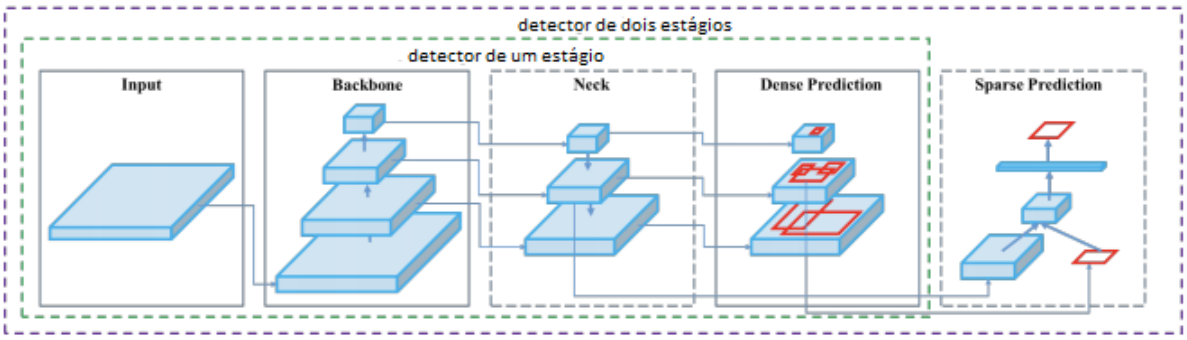
Figura 10 – Processo geral de detecção de objetos da arquitetura YOLO (Adaptado)



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

a técnica de aprendizado por transferência com o modelo pré-treinado auxilia a rede identificar rapidamente quais filtros geram os melhores mapas de características, assim economizando tempo de treinamento de um novo detector.

Figura 11 – Arquitetura da quarta versão do YOLO (Adaptado)



Fonte: Bochkovskiy, Wang e Liao (2020).

Após a extração é adentrado a camada *Neck*, que realiza a junção das saídas do *Backbone*, que possuem as características, para o preparamento da detecção. A *Neck* possui a abordagem PANet para a agregação de recursos da rede, descrita por Tan, Pang e Le (2020). Ao final, na etapa *Dense Prediction* é realizado a detecção dos múltiplos objetos e suas devidas classificações, desenhando na imagem suas caixas delimitadoras.

Com a vinda do YoloV4, não demorou muito para o YoloV5 (JOCHER *et al.*, 2020) ser desenvolvido. Porém, o YoloV5 não possui um artigo científico que evidencie sua melhora no desempenho em comparação com os anteriores. Além disso, diferentemente do YoloV4, a quinta versão foi criada a partir do framework Pytorch feito em Python e não em C, como no framework Darknet. O YoloV5 forneceu cinco tipos de tamanhos de redes (N,S,M,L e X) diferentes, para diversas demandas e cenários de poder de processamento e precisão. Estruturalmente, YoloV5 e YoloV4 são muito semelhantes nas camadas de Backbone, Neck e Head. Portanto, o YoloV5 se tornou uma alternativa do YoloV4, mas não uma versão comprovadamente melhor no aspecto de precisão.

2.2.2.2 Scaled-YoloV4 e YoloR

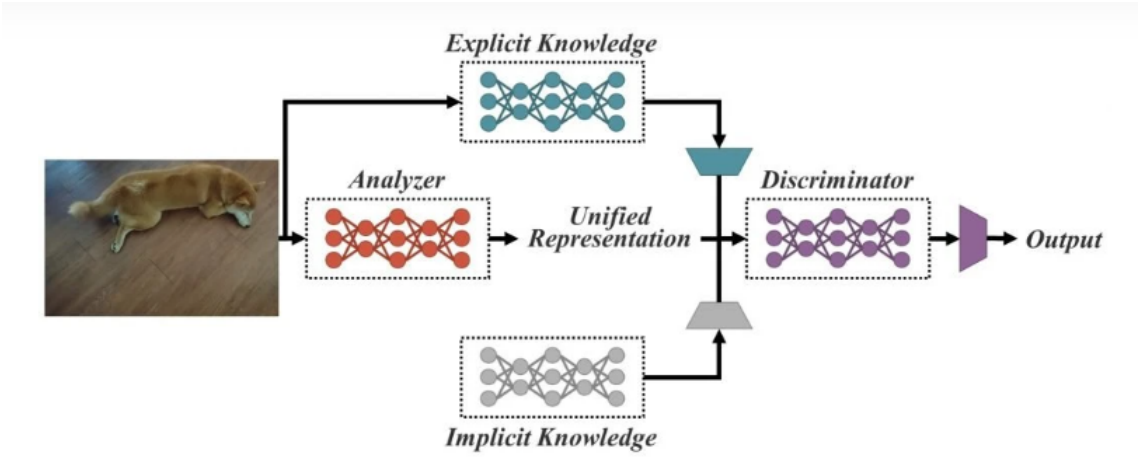
Com o uso do Pytorch no YoloV5, houve um melhoramento no tempo de treinamento de modelos, e isso possibilitou uma implementação melhorada do YoloV4, o Scaled-YoloV4 (WANG; BOCHKOVSKIY; LIAO, 2021). Um dos principais motivos dessa versão ser rápida, é o uso de um redes neurais convolucionais criadas seguindo os conceitos de Cross-Stage Partial Networks (WANG *et al.*, 2019) como no YoloV4, porém a principal contribuição dessa versão é o aumento da profundidade e número de estágios nas camadas de Backbone e Neck, assim, melhorando o desempenho na detecção de objetos grandes em imagens de alta resolução. Outra característica diferente do YoloV4, é que o Scaled-YoloV4 utiliza menos data augmentation sobre o conjunto de dados de treinamento. Em contra-partida, no conjunto de teste, é feito o Test Time Augmentations, que aplica esses aumentos de dados entre resultados de predições, assim melhorando a performance. No geral, o Scaled-YoloV4 é superior em performance sobre o YoloV4, demonstrado sobre o MS COCO.

No mesmo ano que a versão Scaled-YoloV4 era lançada, algumas novas técnicas sobre o Yolo estavam sendo implementadas, como, por exemplo, o YoloR (WANG; YEH; LIAO, 2021), que traz uma abordagem supervisionada de aprendizado (aprendizado implícito) misturada com o aprendizado explicito, que é baseado na entrada imediata que foi dada à rede. A ideia dessa versão é permitir que a máquina possa, com uma única entrada, ter interpretações que servem a várias tarefas, ou seja, novos ângulos de aprendizado e não somente utilizando daquilo que foi aprendido previamente. Fundamentalmente, a ideia possui três partes (Figura 12) para o YoloR funcionar. Primeiramente é feito o processo de alinhamento de espaço do kernel, refinamento de previsão e a criação de uma rede neural convolucional com aprendizado multi-tarefa. Essa CNN não só aprende como obter a saída correta, mas também retorna as outras possíveis saídas coerentes, que representa as diversas interpretações da imagem.

2.2.2.3 YoloX, YoloS e YoloV7

Outra variante é o YoloX (GE *et al.*, 2021b). Ele tem como baseline o YoloV3, porém com algumas melhorias, principalmente para o treinamento. o YoloX aplica as tecnicas de data

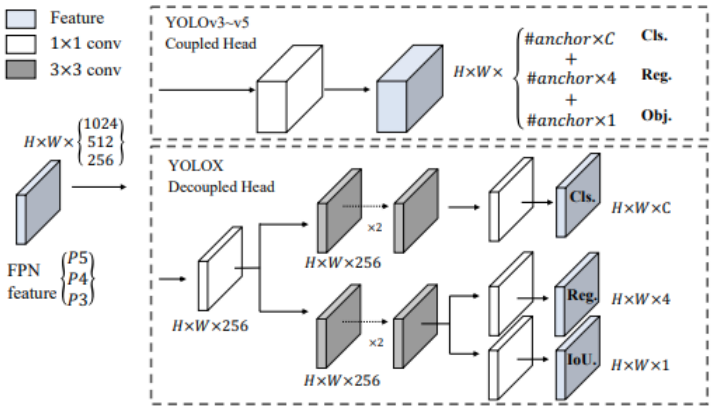
Figura 12 – Arquitetura proposta para a unificação do aprendizado implícito e explicito do YoloR.



Fonte: Wang, Yeh e Liao (2021).

augmentation, como, por exemplo, RandomHorizontalFlip, ColorJitter, Multi-scale, Mosaic e MixUp sobre conjunto de dados de treinamento. Outro ponto importante e inovador dessa versão é o não uso de âncoras, que apesar de ter sido amplamente utilizada por versões anteriores para detecções de mais objetos em um mesmo grid, existem algumas desvantagens, como, por exemplo, estipular âncoras ideias antes do treinamento é demorado com o método de análise de agrupamento. Além disso, o fato de que detectar mais de um possível objeto no grid afeta diretamente a performance em alguns sistemas. Essa mudança é ilustrada na Figura 13. Entre outras inovações é a implementação do SimOTA, uma versão otimizada do método OTA[1] com uso do algoritmo Sinkhorn-Knopp.

Figura 13 – Estrutura da camada Head de classificação e regressão desacoplada do YoloX.



Fonte: Ge *et al.* (2021b).

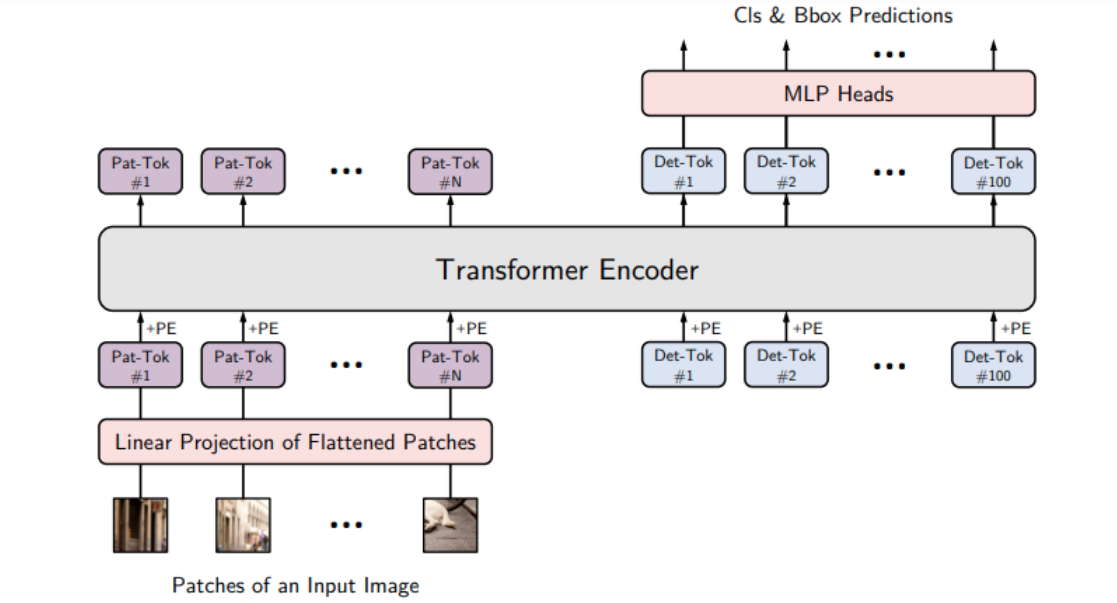
Em uma outra linha e mesclando conceitos de transformers[2], temos o YoloS (FANG *et*

[1] Método avançado de atribuição de rótulos candidatos à objetos. Para mais informações consultar no trabalho de Ge, Z., Liu, S., Li, Z., Yoshie, O. e Sun, J. (2021) (GE *et al.*, 2021a).

[2] Modelos de aprendizado de máquina para dados em sequência. Este tipo de técnica vem substituindo

al., 2021). Ele utiliza do vanilla Vision Transformer e uma versão adaptada da base do Bert para o tratamento de lotes de imagens de forma sequencial, de tamanho fixo e sem sobreposições. Sua arquitetura pode ser vista na Figura 14. Para testar o modelo, os autores utilizaram o ImageNet para pre-treinar os modelos. Um dos fatores que os autores destacam é que as detecções ainda são muito sensível ao pré-treinamento. Além disso, os autores informam que essa versão não tem como foco obter altas performances e sim ser uma alternativa de detectores.

Figura 14 – Arquitetura do YoloS com Transformers.

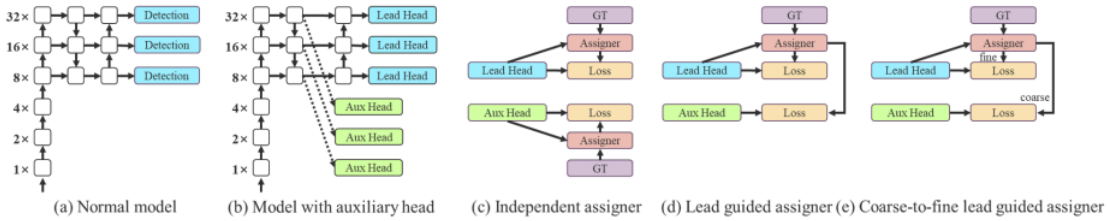


Fonte: Ge *et al.* (2021b).

Finalmente, a versão estado-da-arte em detecção de objetos foi lançada no trabalho de Wang, Bochkovskiy e Liao (2022), denominado YoloV7. Foi procurado aumentar a precisão na predição de caixas delimitadoras. Em suma, as principais contribuições dessa versão é a diminuição da propagação do gradiente no back-propagation, que possui uma co-relação com a quantidade de memória que é utilizado para armazenar as camadas da rede. Essa contribuição ajuda na rapidez do aprendizado da rede. Para isso, foi proposto em sua arquitetura o uso de Extended Efficient Layer Aggregation Network (E-ELAN). Outra contribuição, é que o YoloV7 escalam seus modelos em profundidade, largura e resolução enquanto concatenam as saídas das camadas. A re-parametrização também é usada no YoloV7, ela permite que pesos possam se tornar mais robustos na identificação de características gerais do modelo que se pretende criar. Além do mais, o YoloV7 implementa na faixa intermediaria da rede as Auxiliary Head Coarse-to-Fine. São camadas Head auxiliares para supervisionar o rumo das futuras detecções que serão realizadas nas camadas finais. Elas não possuem tanta precisão quanto as futuras predições dos objetos, mas indicam como o modelo pode estar se comportando durante o treinamento. Pode ser observado essa estratégia na Figura 15.

redes neurais recorrentes tradicionais para aplicações de processamento de linguagem natural.

Figura 15 – Auxiliary Head Coarse-to-Fine propostas no YoloV7.



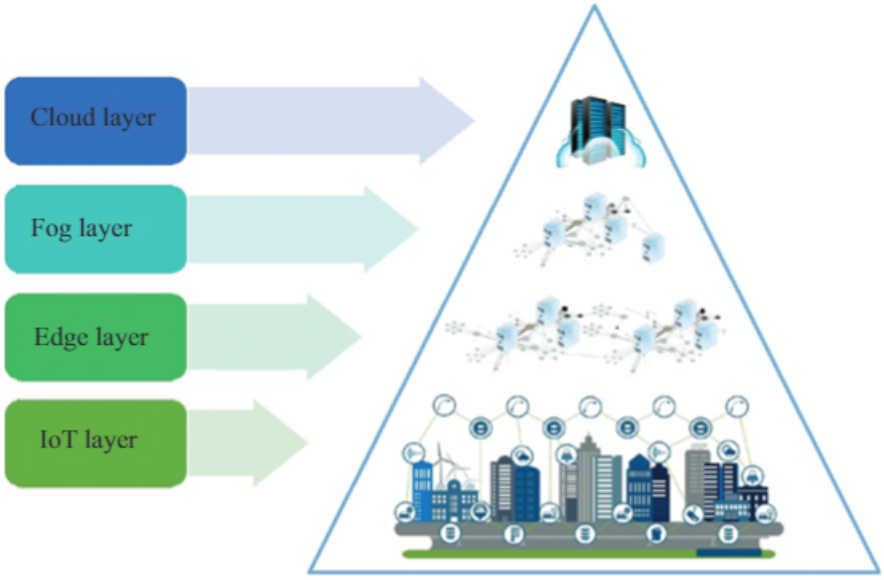
Fonte: Wang, Bochkovskiy e Liao (2022).

2.3 Computação de borda

A computação de borda é definido de acordo com Pang e Tan (2004 apud TAHERI; Shuiguang Deng, 2020) como "*um sistema que distribui métodos de programa e os dados correspondentes para a borda da rede para melhorar o desempenho do sistema*", outra definição feita por Satyanarayanan (2017 apud TAHERI; Shuiguang Deng, 2020) "*Computação de borda é um novo paradigma no qual recursos substanciais de computação e armazenamento (também chamados de cloudlets, micro data centers ou nós de névoa) são colocados na borda da Internet em estreita proximidade com dispositivos móveis ou sensores.*". O termo borda se refere à conectividade nos dispositivos de uso final e cotidiano, e que também possibilitou o surgimento e avanço da *Internet of Things* (IoT). As principais motivações da criação da computação de borda são as dificuldades encontradas na computação em nuvem, como a velocidade do transporte de dados, que impede o fluxo de tráfego a medida que a quantidade de dados gerados na borda continua a aumentar, e a quantidade de dados na borda é muito grande e utiliza muita largura de banda e recursos de computação desnecessários. Além disso, a exigência de proteção da privacidade dificulta a computação em nuvem na IoT. Outro motivo é que dispositivos de borda geram muitos fluxos de dados e a execução da análise em uma nuvem distante obstrui a tomada de decisões em tempo real (TAHERI; Shuiguang Deng, 2020). A Figura 16 apresenta a arquitetura geral do modelo de computação de borda. A camada base (*IoT layer*) representa a conectividade dos dispositivos de IoT e a geração e consumo de dados. Na camada superior (*Edge layer*), estão os dispositivos de processamento de borda cujo objetivo é aproximar o processamento para os dispositivos finais a suprir as necessidades de velocidade e entrega de dados em tempo real. Logo acima, a camada de névoa (*Fog layer*) faz o intermédio de comunicação com preparação de dados entre os dispositivos de borda com a camada de computação em nuvem (*Cloud layer*), que realiza o processamento de rotas e implementa serviços de *Application Programming Interface* (API) para as camadas de baixo para distribuição de dados para sites e outros sistemas no núcleo da rede.

Com o advento dos sistemas de aprendizado de máquina e profundo, as intenções de aplicações destes tipos na borda com uso de I.A aumentaram. Porém, as aplicações com computação de borda comumente têm como foco tarefas menos complexas do que aplicar algoritmos

Figura 16 – Arquitetura geral do modelo de computação de borda



Fonte: Taheri e Shuiguang Deng (2020).

de I.A (LI; XU; ZHAO, 2015)(BLAIECH *et al.*, 2019), pela limitação de processamento dos dispositivos e que não é possível utilizar uma *Graphic Processing Unit* (GPU) tradicional para melhorar o desempenho destes algoritmos, que muitas vezes são para detecção e classificação de objetos (WANG *et al.*, 2017) (HAO *et al.*, 2020). Com o desafio de permitir que dispositivos de computação de borda e sistemas embarcados pudessem executar algoritmos e sistemas de I.A, tendo em vista a gama de arquiteturas de dispositivos, algumas empresas como a Intel, Google e NVIDIA procuraram fornecer algum tipo de processamento especializado para inferências na borda, por exemplo, as arquiteturas Intel Neural Compute Stick 2 (Intel NCS2)[3](Figura 17), NVIDIA Jetson[4] (Figura 18) e Google Coral[5]. Com a popularização do uso destes módulos especializados, avaliações, comparativos e testes são muito necessários para entender a eficiência de inferências, uso de memória e processador, e gasto energético, fundamentais para a manutenibilidade de aplicações na borda e o desgaste de *hardware* dos dispositivos com o tempo (BAI; LIU; YI, 2019)(KLJUCARIC; JOHNSON; GEORGE, 2020). Portanto, com ascensão da área de visão computacional, esta categoria de dispositivos está se tornando cada

[3] É uma arquitetura especializada para aplicações de aprendizado profundo da Intel, que possuí uma *Vision Processing Unit* (VPU) Intel® Movidius™ Myriad™ programável, um processador especializado para tarefas de visão computacional. Para mais informações, acesse (URL): <https://www.intel.com.br/content/www/br/pt/products/details/processors/movidius-vpu.html>

[4] "Os módulos NVIDIA Jetson *Family* oferecem recursos de computação acelerada em diferentes níveis de desempenho e preços para atender a uma variedade de aplicações autônomas.". Fonte: <https://www.nvidia.com/pt-br/autonomous-machines/>

[5] O Coral apresenta um coprocessador de borda do tipo TPU (*Tensor Processing Unit*) para aplicações que utilizam por exemplo o *framework* Tensorflow, permitindo inferência de aprendizado de máquina em uma ampla variedade de sistemas que tenham conexão e porta USB. Para mais detalhes, acesse (URL): <https://coral.ai/products/accelerator/>

vez mais essenciais para aplicações, que envolvem desde reconhecimento facial para segurança pública até sistemas de robótica no campo agrícola (SANTOS *et al.*, 2020). Entretanto, apesar do maior poder computacional com estes dispositivos, o preço de aquisição e implantação ainda são elevados, podendo custar até cinco vezes mais que um dispositivo de borda convencional. Portanto, apesar de ser citado a existência de tais dispositivos, neste trabalho é usado somente dispositivos sem especialização para I.A, visando testar as técnicas com dispositivos de baixo custo.

Figura 17 – Imagem ilustrativa de um Intel NCS2 sendo implementado em um veículo aéreo não-tripulado

Fonte: Nogueira (2018).

Figura 18 – Imagem ilustrativa do NVIDIA Jetson Nano para implementação de um robô autonomo.

Fonte: NVIDIA (2023).

CAPÍTULO

3

TRABALHOS CORRELATOS

Este capítulo visa apresentar os principais trabalhos relacionados com o tema de detecção e classificação de animais para solucionar diversos problemas utilizando técnicas de visão computacional e processamento de imagens. Ao final, é feito um comparativo com o presente trabalho, ressaltando as diferenças e contribuições.

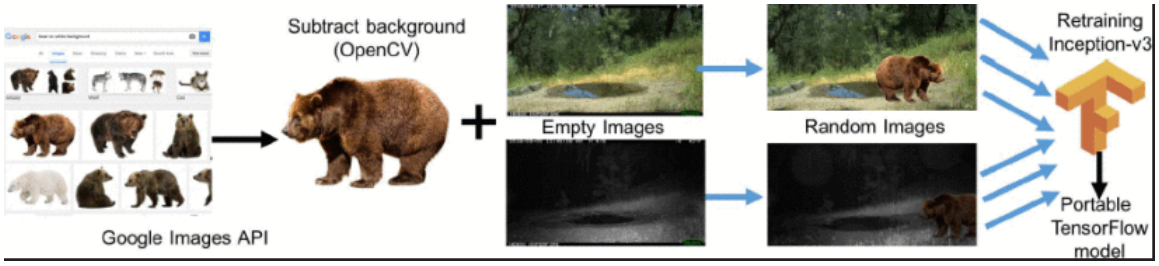
Elias *et al.* (2017) desenvolveram o sistema denominado "*Where's the Bear?*"(WTB), visando monitorar animais da reserva ambiental UCSB Sedgwick para o entendimento e conservação da reserva, como contagem e estimativas de população de espécies. O WTB utiliza um mecanismo de IoT distribuído utilizando câmeras e um dispositivo para o processamento na borda e serviços em nuvem (público e privado) para o treinamento do modelo de redes neurais convolucionais para a detecção e classificação de ursos, veados e coiotes. Para o treinamento do modelo de visão computacional, um conjunto de dados feito com auxílio do Google Imagens de imagens somente de fundo vazio do ambiente e editadas com os animais foi utilizado. O WTB foi testado sobre um conjunto de imagens com mais de 1,1 milhão de imagens da reserva.

A Figura 19 mostra a etapa de treinamento realizada. As imagens dos animais são buscadas do Google Images e retiradas seus fundos para ter somente o animal com uso do *Open Source Computer Vision Library* (OpenCV). Após isso, as imagens dos animais são colocadas em cenários do ambiente real e enviadas para uma rede neural com arquitetura Inception-v3[1] no TensorFlow[2]. Outra característica do sistema é ser adaptável, ou seja, permite a integração a diversos algoritmos de classificação e processamento de imagem, via APIs ou *scripts*. Este modelo foi incorporado em uma reserva ecológica e teve resultados positivos, tanto em taxa de velocidade de imagens para a nuvem quanto em precisão do modelo nas detecções.

[1] É a terceira versão da arquitetura de CNN desenvolvida pelo Google para detecção de objetos. (SZEGEDY *et al.*, 2015)

[2] O TensorFlow é uma plataforma que facilita a criação e a implantação de modelos de ML. É disponibilizado como pacote Python ou como API para treinamentos distribuídos. Mais sobre em (URL): <https://www.tensorflow.org/about?hl=pt-br>

Figura 19 – Etapa de treinamento de um modelo de detecção com imagens sintéticas de animais



Fonte: Elias *et al.* (2017).

No trabalho de Song e Lin (2018) é utilizado dois modelos de rede neural convolucional (um com arquitetura MobileNet[3] e outro Inception-v3), treinados com auxílio de aprendizado por transferência[4] de modelos pré-treinados pelo *dataset* ImageNet[5]. O objetivo é realizar a classificação e identificação de espécies em tempo-real, visando ajudar os departamentos de ecologia a coletar melhores informações sobre espécies com risco de extinção na China, como os pandas, íbis-do-japão, tigre-siberiano, macacos dourados, esturjão-chinês, guepardos, flamingos e coalas, animais com risco de extinção na China. Os dados utilizados são abertos a comunidade local e foram melhorados utilizando a técnica *Data augmentation*, que aumenta o número de imagens, criando manualmente variações das mesmas imagens, mas com filtros e modificações que não interfiram no tipo de imagem. Ao todo, para cada uma das classes, foi obtido 200 imagens, totalizando 1,600 imagens, separadas em 80% para treinamento, 10% para verificação e os outros 10% para teste. A Figura 20 apresenta a abordagem feita pelos autores. A etapa 1 é a entrada da imagem para as camadas de convolução do modelo para a extração das características. A etapa 2 representa o vetor de características final adentrando as camadas totalmente conectadas especializadas nos dois modelos testados. Com o resultado das probabilidades da detecção para cada classe, é aplicado uma função de ativação de somente um neurônio (Softmax) para determinar qual classe de fato foi obtida (etapas 3, 4 e 5). Foi comparado os dois modelos e o modelo baseado em MobileNet teve o melhor resultado com uma acurácia de 89% sobre o conjunto de teste e com detecções e classificações bem sucedidas em um aplicativo móvel.

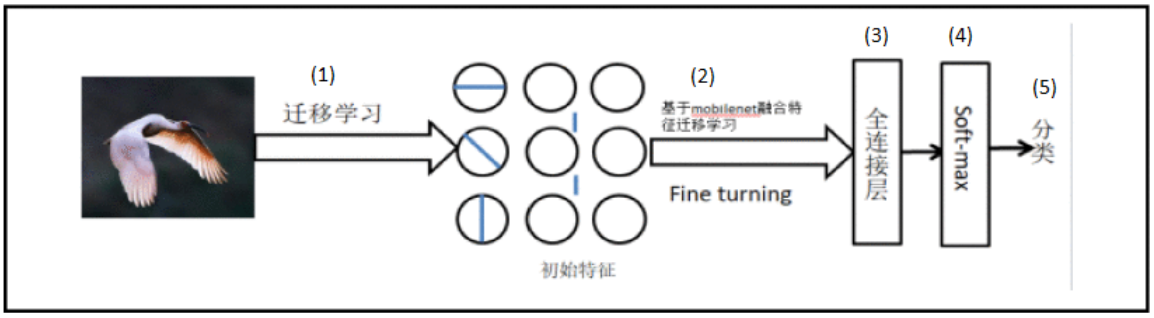
No trabalho de Arruda *et al.* (2018) é abordado o problema da identificação e mensuração de densidade de espécies no Pantanal brasileiro, que ajuda amenizar os problemas da biodiversidade ser danificada pelas demandas socioeconomias, que envolvem a pecuária, agricultura, turismo e caças ilegais. Essas tarefas fazem parte do trabalho de diversos pesquisadores de

[3] MobileNet é uma arquitetura para redes neurais convolucionais otimizada para dispositivos móveis. (HOWARD *et al.*, 2017)

[4] No aprendizado por transferência são retiradas as camadas que não se deseja de uma rede pronta e colocadas novas camadas sobre esta rede, estas novas são as camadas especializadas no problema em questão.

[5] ImageNet é um conjunto de imagens para *benchmark* de sistemas de detecção de objetos. É um dos mais famosos e contém mais de 3.2 milhões de imagens divididas em mais de 1000 classes. (DENG *et al.*, 2009)

Figura 20 – Processo de detecção e classificação de animais em extinção da China (Adaptado)



Fonte: Song e Lin (2018).

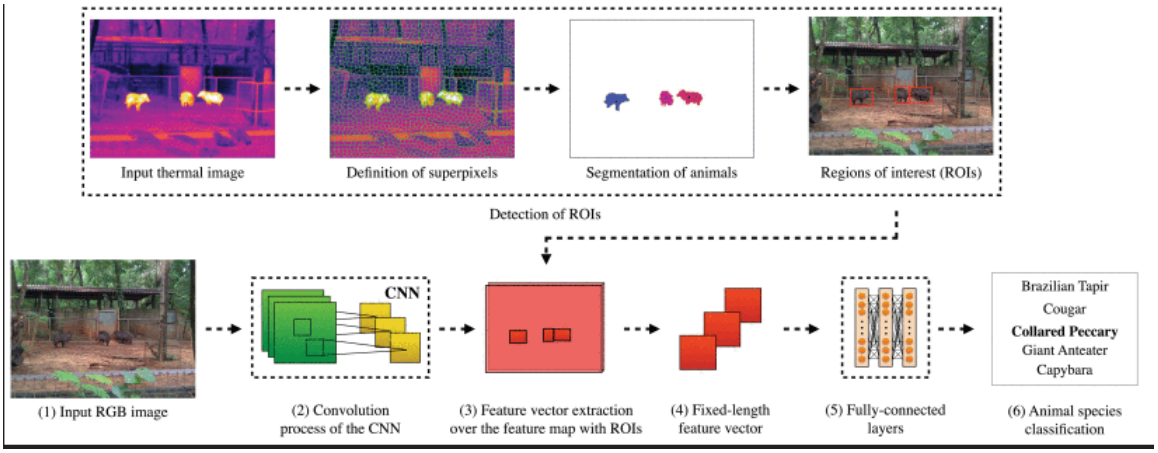
ecologia para o combate a degradação do Pantanal e que geralmente é muito lenta e custosa, feito movimentações na mata por vários dias e com uso de câmeras fotográficas armadilhas para capturar os animais em determinado espaço. Os autores, portanto, criaram uma abordagem para detectar e identificar automaticamente as espécies animais do Pantanal usando CNN com segmentação de regiões de interesse em imagens térmicas e RGB com o algoritmo SLIC[6]. A CNN usada é baseada na arquitetura VGGNet com 16 camadas ao todo para identificação sendo comparada com a rede *Fast* R-CNN clássica. É usado como conjunto de imagens para treinamento do modelo de 4.800 imagens de um sub-conjunto do ImageNet com oito classes do bioma Pantanal, como a Anta brasileira, Arara-azul-e-amarela, Onça-parda, Caititu, Capivara, Papagaio-verdadeiro, Quati-de-cauda-anelada e o Tamanduá-bandeira. O conjunto de imagens de teste é composto por 1.600 imagens capturadas no próprio bioma, são térmicas e RGB.

A abordagem proposta é apresentada na Figura 21. A etapa 1 é a imagem RGB servindo de entrada para a rede convolucional, logo após, na etapa 2 ocorre o processo de convolução usando a imagem inteira para obter o mapa de características da última camada convolucional. A etapa 3 realiza a extração de mapa de características usando as regiões de interesse com a técnica de segmentação. A seguir (etapa 4) as características são convertidas em vetores de tamanho fixo na camada de *max-pooling*, que servirão de entrada para a rede totalmente conectada (etapa 5) para realizar a predição com as caixas delimitadoras na etapa 6. Os resultados alcançados demonstram que a abordagem foi melhor nas métricas de *Average Precision* (AP) e *f-measure* comparada ao *Fast* R-CNN clássica para a tarefa de detecção.

No trabalho de Schneider, Taylor e Kremer (2018) foi realizado o uso de redes neurais profundas para tarefas de detecção de objetos para a identificação, contagem e localização de animais em imagens provindas de câmeras armadilhas. O grande desafio é que estas imagens podem apresentar diversas variações de posicionamento do animal de interesse, como oclusões de ambiente, iluminações irregulares, poses em ângulos desafiadores e cortes da localização completa do animal, sendo desafiador para algoritmos de visão computacional, como pode ser

[6] O algoritmo SLIC é usado para segmentação com base na similaridade da cor CIELab e distância espacial. São amplamente utilizados em imagens coloridas, de sensoriamento remoto óptico, de cenas naturais e outras tarefas de segmentação de imagem(ACHANTA *et al.*, 2012).

Figura 21 – Abordagem proposta para detecção de animais do bioma Pantanal



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

visto em um dos experimentos (Figura 22). É comparado um modelo de dois estágios Faster R-CNN com um modelo de único estágio YoloV2. Estes modelos são comparados nos aspectos de velocidade para detecções em tempo real e acurácia sobre dois datasets de imagens armadilhas, o The Reconyx Camera Trap (RTC) e o Gold Standard Snapshot Serengeti (GSSS). Os resultados mostraram que o modelo Faster R-CNN demonstrou melhor desempenho em relação à acurácia que o modelo YoloV2 em ambos os conjuntos de imagens, concluindo que esta técnica é bem eficaz para detecções em processamento de imagens de câmera armadilhas.

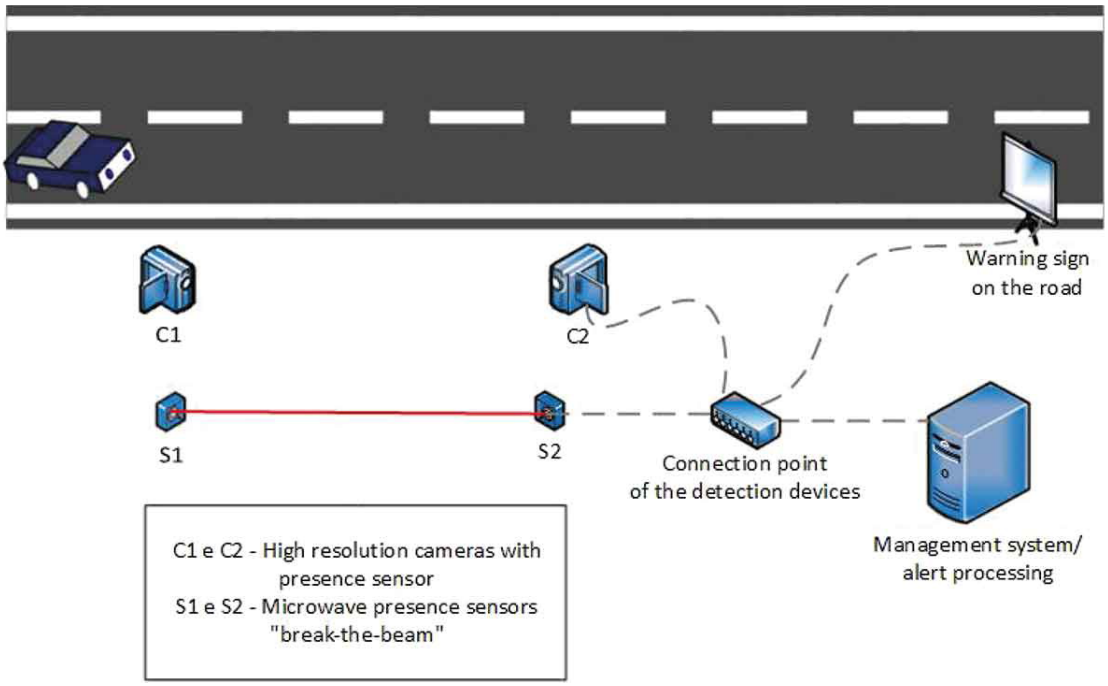
Figura 22 – Uso do algoritmo Faster R-CNN para detectar 10 gazelas-de-thomson do conjunto de dados GSSS e demonstrando as dificuldades das distâncias dos animais durante a detecção.

Fonte: Schneider, Taylor e Kremer (2018).

Para o problema de acidentes com animais no Brasil, o trabalho de Antônio *et al.* (2019)

propõe um sistema simples de detecção de animais, uma arquitetura e uma metodologia para detecção de animais em imagens de câmeras em estradas. A arquitetura pode ser vista na Figura 23, onde é proposto o uso das câmeras das rodovias, com sensores de presença de borda ao lado. A comunicação se dá pelo uso da abordagem de cliente/servidor, onde os clientes são as câmeras que enviam dados a um sistema de gerenciamento (servidor). Ao final, é transmitido o sinal da presença ou não de animais em dispositivo de visualização no acostamento da pista. O conjunto de dados utilizado é um conjunto próprio e criado de forma sintética, colocando os animais em cenários rodoviários.

Figura 23 – Arquitetura proposta para um sistema de detecção animal para rodovias.



Fonte: Antônio *et al.* (2019).

A metodologia segue o modelo tradicional de um sistema de AM. O sistema permitiu a extração de características de regiões da imagem e o uso de técnicas de AM supervisionado para classificar as áreas entre dois valores, se tem ou não um animal. Os autores propuseram cinco abordagens para retirar as características de animais ou não animais baseadas no conceito de blocos na imagem. Esta abordagem foi utilizada com dois blocos, um para reconhecimento do ambiente e outro para o reconhecimento do objeto para a classificação. Simultaneamente, também foi aplicado quatro diferentes espaços de cores, como o *Red, Green, Blue* (RGB), *L-star, a-star, b-star* (LAB)[7], o modelo HSV[8] e o formato de imagem em escala cinza. Para o aprendizado supervisionado foi escolhido dois algoritmos, o *K-Nearest Neighbors* (KNN)[9] e o

[7] É também chamado de CIELab *color*. O *L-star* representa a luminosidade, *a-star* representa os valores de vermelho e verde, *b-star* representa os valores de azul e amarelo.

[8] Cor H representa a cor primária mais próxima, valor S é referente a saturação e o valor V representa o quão escuro é a imagem.

[9] KNN é um algoritmo de AM para aprendizado supervisionado, é um método de classificação e

Random Forest (RF)[10]. O dataset que foi utilizado possui somente 20 imagens sintéticas, criadas para representar situações onde há ou não um animal na pista. A abordagem com uso do KNN se sobressaiu perante a RF, com melhores resultados.

O trabalho de Biswas *et al.* (2021) procurou realizar um comparativo de redes neurais convolucionais para detecção no reconhecimento de espécies de pássaros na cidade de Bangladesh, na Índia, que conta com mais de 800 espécies e devido a essa alta variedade, manualmente seria inviável realizar a classificação, porém, com a aplicação de modelos de aprendizado de máquina isso é possível. Foram escolhidas sete espécies para o treinamento com aprendizado por transferência sobre 2800 imagens e 700 imagens para validação e teste sobre os modelos DenseNet201, InceptionResNetV2, MobileNetV2, ResNet50, ResNet152V2, e Xception. A metodologia aplicada pode ser observada na Figura 24. Foi avaliado os resultados de acurácia, precisão, revocação e *F1-score*. O estudo mostra que os modelos MobileNetV2 e Xception obtiveram os maiores valores de resultados em todas as métricas. É concluído que o modelo MobileNetV2 é o modelo superior em comparação aos demais, além disso, pode se concluir também que mesmo com um conjunto pequeno, o modelo obteve altos resultados para reconhecimento de pássaros.

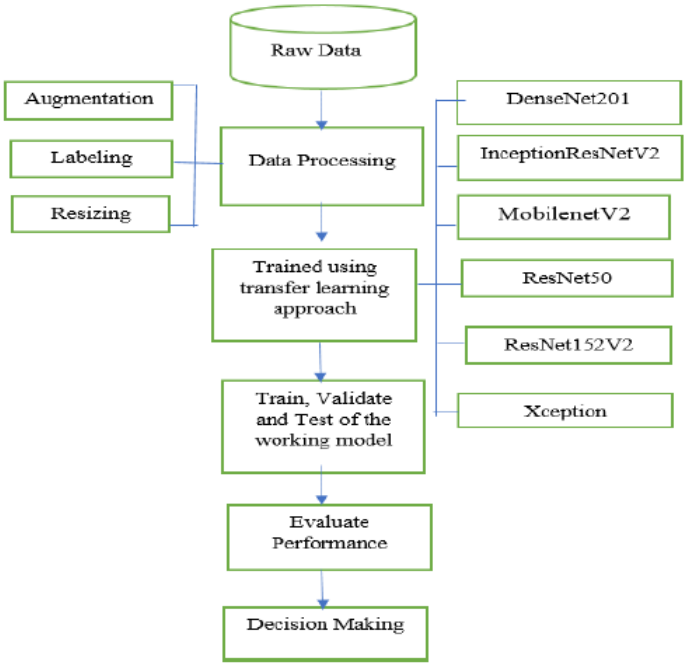
Em outro trabalho, Adami, Ojo e Giordano (2021) apresentam uma solução que envolve computação de borda e de nuvem e visão computacional para a expulsão segura de animais como javalis e cervos em áreas agrícolas de fazendas. O sistema de visão computacional interage com o módulo em borda por dispositivos especializados em processamento de aprendizado profundo (Intel Movidius Neural Compute Stick (NCS) e NVIDIA Jetson Nano) acoplados em um Raspberry Pi Model 3 B+, a visão geral do sistema pode ser observada na Figura 25. Os detectores de objetos em tempo real utilizados são o YoloV3 e o YoloV3-Tiny (versão mais leve do YoloV3). É avaliado as métricas de revocação, Average Precision (AP) e mean Average Precision (mAP) de ambos os modelos. Também é avaliado a performance em taxa de quadros por segundos (FPS) dos modelos implementados com e sem os dispositivos especiais para redes neurais na borda. Conforme os experimentos, o YoloV3 obteve melhor performance geral com 82.5% de mAP, sendo superior a sua versão Tiny que atingiu 62.4% de mAP. Contudo, em questão de FPS, o modelo Tiny em conjunto com o Nvidia Jetson obteve melhor resultado, com 15 FPS alcançados, em contra-partida, o modelo YoloV3 só obteve 4 FPS em seu melhor resultado (com o Jetson Nano). Conclui-se que o tipo de rede influência na performance de detecções na computação na borda, mesmo em tarefas soft-real time.

Para identificação e detecção de rebanhos de rinocerontes-brancos, girafas, gnus e zebras,

regressão, para mais detalhes de seu funcionamento acesse (URL): <https://towardsdatascience.com/k-nearest-neighbors-knn-explained-cbc31849a7e3#:~:text=K-nearest%20neighbors%20(kNN)%20is%20a%20supervised%20machine%20learning,that%20comes%20from%20real%20life.&text=kNN%20works%20similarly.,the%20data%20points%20around%20it.>

[10] O RF é um método de aprendizagem de conjunto para classificação e regressão com uso de múltiplas árvores de decisão, para mais detalhes acesse (URL): <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>

Figura 24 – Metodologia para reconhecimento de aves utilizando redes neurais convolucionais com aprendizado por transferência.

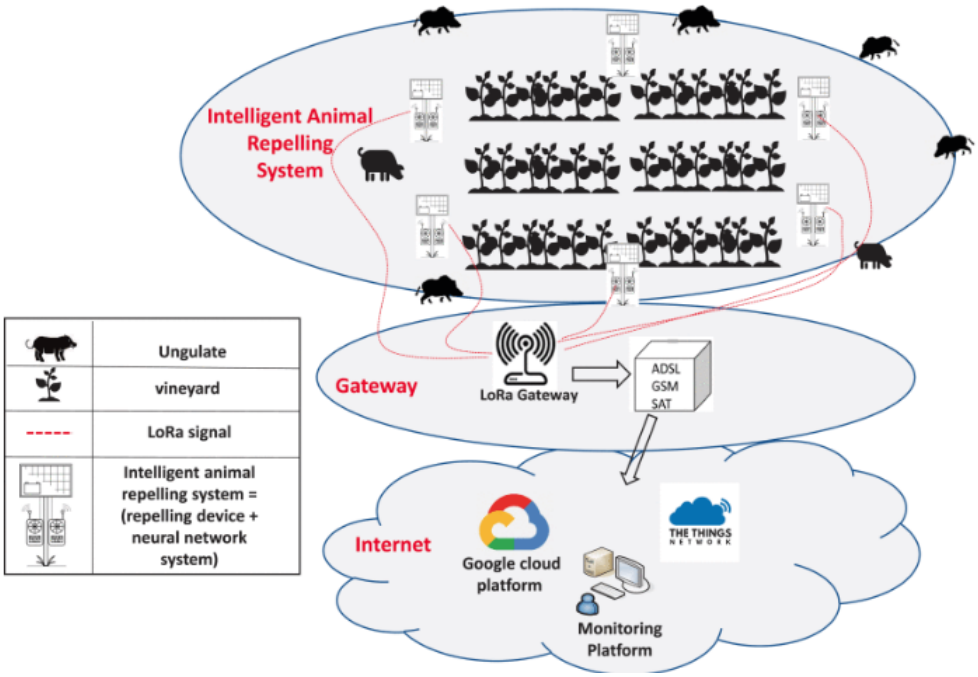


Fonte: Biswas *et al.* (2021).

o trabalho de Petso *et al.* (2021) utilizaram detectores baseados em YOLO para detecções via imagens capturadas por drones. O desafio é detectar animais com imagens aéreas, já que pode haver animais com camuflagem com o meio ambiente. Além disso, a implementação teve como objetivo principal, criar um mecanismo de monitoramento com visão computacional para manter em segurança os rinocerontes contra o risco de caça. Para tal, é construído um conjunto de imagens com imagens aéreas com imagens em diferentes altitudes e feito o treinamento, validação e avaliação dos modelos de detecção YoloV3 e YoloV4, como apresentado na Figura 26. É demonstrado que o YoloV4 obteve 13% a mais de performance nas detecções em tempo real. Também, em comparações das detecções dos animais para cada altitude, até 40 metros de distância dos animais do chão, ambos os detectores tiveram acurácias superiores a 97%, no entanto, após 50 metros de distância até 130 metros, o YoloV4 conseguiu manter suas precisões acima dos resultados do YoloV3, provando ser um detector mais eficaz e eficiente para sistemas de monitoramento de animais com risco na savana africana.

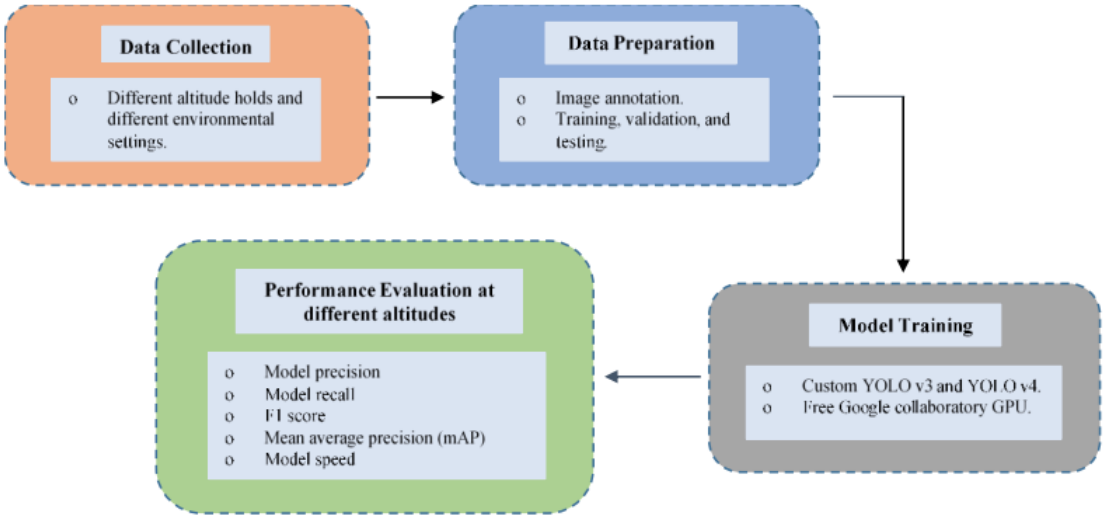
Os oito trabalhos descritos acima servem de influência direta para a criação da proposta deste trabalho. A Tabela 2 apresenta as principais comparações com os trabalhos citados. Cada um contribuído com ideias que podem ser aplicadas para a criação de mecanismos de detecção de animais para rodovias com redes neurais. Os trabalhos de Elias *et al.* (2017) e Adami, Ojo e Giordano (2021) apresentaram soluções com dispositivos IoT e uma arquitetura completa para o monitoramento dos animais que conseguiu bons resultados. O uso destes dispositivos IoT também é abordado neste trabalho, porém aplicado em um problema e cenário diferente

Figura 25 – Visão geral da proposta utilizando visão computacional com computação de nuvem e IoT para detecção de animais e expulsão em áreas agricolas.



Fonte: Adami, Ojo e Giordano (2021).

Figura 26 – Metodologia do processo de comparação dos detectores YoloV3 e YoloV4 em imagens aéreas.



Fonte: Petso *et al.* (2021).

e executando outros tipos de algoritmos de classificação/detecção. Outra característica que inspirou a criação do conjunto de imagens deste trabalho é a criação utilizando parcialmente o GoogleImages e uso de aumento de dados sobre o conjunto. Já os trabalhos de Song e Lin (2018) e Biswas *et al.* (2021) demonstraram grande eficácia no treinamento de modelos com animais em extinção. Suas principais contribuições são o uso de um pipeline com o uso de aprendizado por transferência na etapa de treinamento pode ser interessante em um modelo especializado

para detecções de animais.

O trabalho desenvolvido por Arruda *et al.* (2018) aborda a detecção de animais em extinção no bioma Pantanal. Alguns destes animais também fazem parte deste trabalho. Um dos tópicos interessantes mostrado é a abordagem de testes executada para validar o sistema, que propõe testar o modelo desenvolvido com animais reais *in loco*. Neste trabalho, pela dificuldade de prever e subsequente encontrar os animais nas pistas, uma das etapas de testes/validações pode ocorrer com imagens de zoológicos ou parques ambientais, além de testes com vídeos coletados na Internet.

Já os trabalhos de Schneider, Taylor e Kremer (2018) e Petso *et al.* (2021) criaram comparativos de redes neurais baseadas em arquitetura YOLO, auxiliando neste trabalho a criar a metodologia de comparativo dos detectores propostos utilizando métricas comumente avaliadas da área. Diferentemente dos dois trabalhos, este procura utilizar as versões da arquitetura YOLO que não foram abordadas, uma vez que algumas versões são recentes e há poucos trabalhos envolvendo.

Por fim, o trabalho de Antônio *et al.* (2019) trouxe uma arquitetura completa para detecção dos animais com uso de um fluxo de aprendizado de máquina tradicional, porém não utilizando redes convolucionais para a extração de características dos objetos, algo que comumente vem sendo utilizado na literatura, inclusive neste trabalho. Por sua abordagem também criar um conjunto de imagens sintético e próprio, reforça a criação e uso de um *dataset* especializado para o tema.

No geral, os trabalhos de linha de base abordam diversas técnicas e conjuntos de imagens. As técnicas abordadas variam destes algoritmos de aprendizado de máquina clássicos (KNN e *Random Forest*) até o uso de redes neurais convolucionais, variando de redes de um ou mais estágios. A diferencial deste trabalho é o uso de um conjunto de imagens, aberto e gratuito, especializado em animais com risco de extinção, construído com imagens livres da internet. Além disso, também é feito o comparativo de arquiteturas YOLO recentes, que ainda não foram testadas para o cenário de monitoramento animal em rodovias inteligentes com suporte a computação de borda para aplicação de inteligência artificial.

Sem considerar os trabalhos que envolvem a arquitetura YOLO (SCHNEIDER; TAYLOR; KREMER, 2018; PETSO *et al.*, 2021; ADAMI; OJO; GIORDANO, 2021), as técnicas utilizadas pelos trabalhos semelhantes possuem desvantagens perante sua velocidade de execução em comparação com YOLO, uma vez que a preocupação das primeiras redes convolucionais como Inception, VGGNet, DenseNet, ResNet e entre outras, era a maior precisão ou acurácia possível na classificação e detecção, sempre trazendo maiores larguras e complexidade para as camadas finais da rede, tornando os modelos pesados e não otimizados para dispositivos de borda, tendo ainda como foco inicial de implantação em ambientes com recursos computacionais abundantes, como computadores laboratoriais ou sistemas em nuvem. Por outro lado, tendo em vista as arquiteturas com preocupação na velocidade de inferência, o trabalho de Schneider, Taylor e

Kremer (2018) apresentou o comparativo na transição de detectores de dois estágios para de detectores de somente um estágio (YOLO), comprovando a superioridade da técnica. Além disso, a cada versão de YOLO, há um melhoramento de velocidade e também redução de complexidade das redes, portanto, as arquiteturas anteriores utilizadas pelos trabalhos anteriores não constam como versões estado-da-arte da arquitetura. Este trabalho utilizou das arquiteturas YOLO recentes ainda não averiguadas para detecção animal, tendo visto comprovar a viabilidade do BRA-Dataset para as formulações de detectores para o monitoramento animal.

Tabela 2 – Comparativo entre os trabalhos correlatos com o presente trabalho.

Trabalho	Computação de borda	*Dataset*	Técnica
(ELIAS *et al.*, 2017)	X	Próprio (Sintético)	Inception-v3
(SONG; LIN, 2018)		ImageNet	MobileNet, Inception-v3
(ARRUDA *et al.*, 2018)		ImageNet	VGGNet, FastR-CNN
(SCHNEIDER; TAYLOR; KREMER, 2018)		RTC, GSSS	FastR-CNN, YoloV2
(ANTôNIO *et al.*, 2019)		Próprio (Sintético)	KNN, *Random Forest*
(BISWAS *et al.*, 2021)		Próprio	DenseNet201, InceptionResNetV2, MobileNetV2, ResNet50, ResNet152V2, Xception
(ADAMI; OJO; GIORDANO, 2021)	X	Próprio	YoloV3, YoloV3-Tiny
(PETSO *et al.*, 2021)		Próprio	YoloV3, YoloV4
Este trabalho	**X**	**BRA-Dataset**	**YoloV4, YoloV5, Scaled-YoloV4, YoloR YoloX, YoloV7**

Fonte: Elaborada pelo autor.

CAPÍTULO

4

DETECÇÃO DE ANIMAIS COM RISCO DE EXTINÇÃO UTILIZANDO ARQUITETURAS *YOU ONLY LOOK ONCE* (YOLO) PARA RODOVIAS INTELIGENTES COM SUPORTE A COMPUTAÇÃO DE BORDA

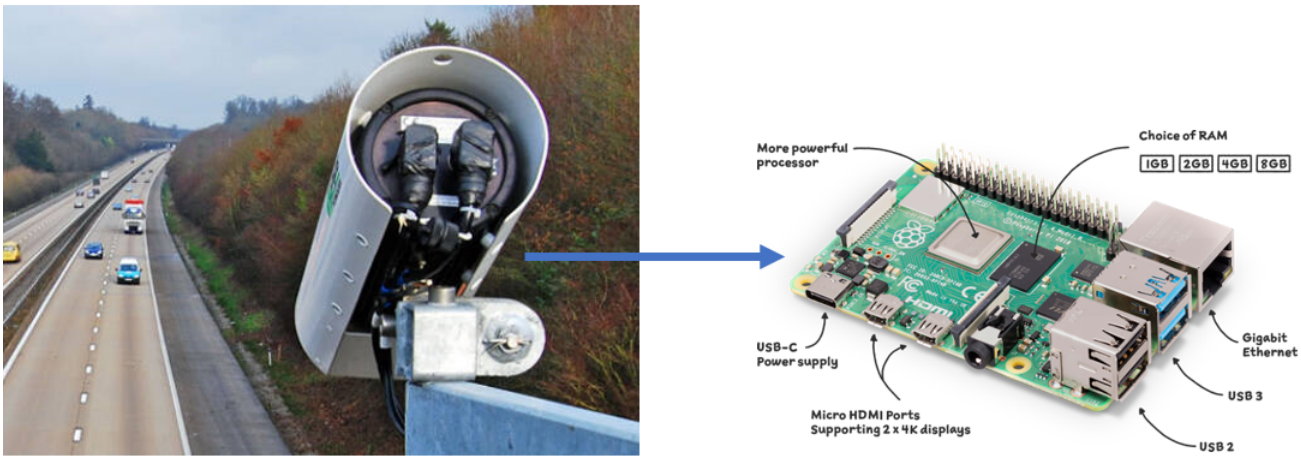
Neste capítulo é apresentado o processo, bem como os materiais e a metodologia de desenvolvimento e construção dos modelos customizados para a detecção de animais utilizando o conjunto de dados proposto. O objetivo deste trabalho com a implementação de modelos de detecção em tempo real é superar as dificuldades existentes no país para a proteção da fauna em ambientes rodoviários, com uso de uma solução tecnológica aplicável as situações e ambientes reais, como componente de sistemas inteligentes de monitoramento em rodovias. Em paralelo, também espera-se fornecer subsídios técnicos e teóricos para o avanço da área de detecção de animais no Brasil.

4.1 Visão geral

No cenário de monitoramento de rodovias, o mais comum é o uso de câmeras acopladas a estações estáticas direcionadas as pistas. Para a execução de algoritmos de processamento de imagem e detecção de objetos, o ambiente onde se encontra a câmera de captura deve estar acoplada com um dispositivo que contém um micro-processador ou um mini-computador, com poder computacional limitado, como exemplificado na Figura 27. Neste trabalho é feita a construção de mecanismos de detecção para estes cenários de uso.

Com o cenário elencado anteriormente, os sistemas gerais de detecção por meio de visão computacional implicam em três módulos, conforme apresentado na Figura 28. O primeiro

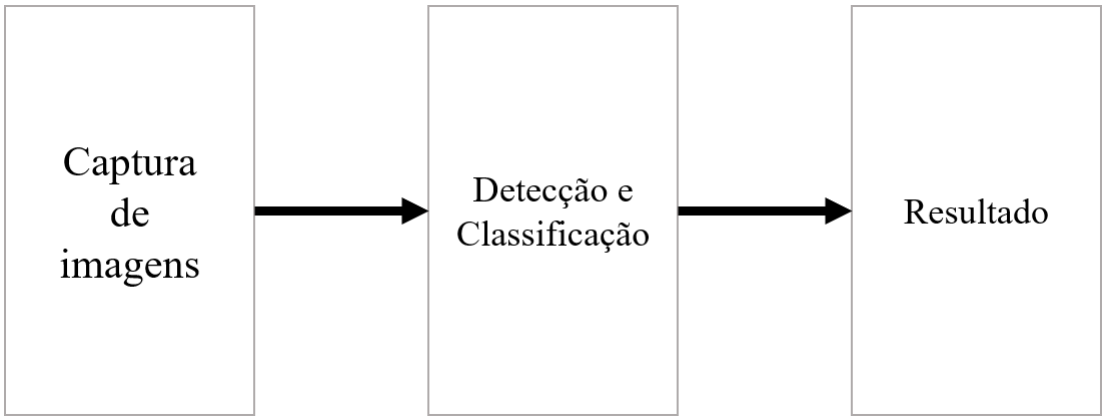
Figura 27 – Cenário de aplicação de sistemas de detecção propostos



Fonte: Elaborada pelo autor.

módulo é referente a tarefa de captura de imagens por meio de câmeras especializadas na estrutura rodovia existente. Logo após, é executada a tarefa de detecção e classificação dos objetos nas cenas, no qual foi proposto um mecanismo que usufrui da arquitetura YOLO, objetivando fornecer detectores com alta velocidade de inferência para possíveis aplicabilidades em tempo real. Além disso, que implementam técnicas de aumento de dados nativos e atuais. Para a especialização dos algoritmos que seguem tal arquitetura, é preciso efetuar o treinamento com imagens de domínio. Tendo em vista que o não há um conjunto de imagens especificamente para animais em extinção brasileiros, este projeto também propõe a construção do BRA-Dataset, que visa superar essa lacuna por meio de uma metodologia de aquisição de imagens gratuitas e de forma automatizada, propondo a viabilização do uso dos detectores escolhidos. Ao final, o último módulo consiste em apresentar o resultado da detecção e classificação, demarcando na imagem original os objetos (no caso, os animais) juntamente com a classe e o porcentual de confiança daquela classificação.

Figura 28 – Módulos gerais de sistemas de detecção



Fonte: Elaborada pelo autor.

Para desenvolver o mecanismo inicial foi necessário criar um conjunto de imagens de animais em extinção, que é utilizado para auxiliar os modelos em questão no entendimento de características. Os detalhes e informações sobre a construção e características do conjunto de dados pode ser compreendido na próxima sessão.

4.2 *Brazilian Road's Animals* (BRA-Dataset)

Na literatura, há uma falta de conjuntos de dados de imagem a respeito de animais que mais sofrem com acidentes no Brasil. Portanto, é um problema que impacta negativamente na construção de possíveis sistemas de monitoramento com detecção. Visando superar esta dificuldade, foi criado o conjunto de imagens *Brazilian Road's Animals*[1] (BRA-Dataset) (FERRANTE *et al.*, 2022). Este conjunto tem como foco, abordar as classes de animais de médio/grande porte com maiores índices de atropelamento no Brasil.

Essencialmente, é buscado a criação de um conjunto de dados de imagens de treinamento e teste com um número alto de imagens para cada classe que compõe o conjunto, de forma que tenha uma proporção na distribuição para cada classe. A quantidade pode variar de objetos de interesse e problema, em um cenário com poucas imagens, este fator pode influenciar na tarefa de treinamento, onde poucas amostras de características relevantes para os objetos não permitem uma generalização eficiente no modelo, para evitar este problema, a estratégia escolhida, é efetuar a captura do maior número possível de imagens para todas as classes.

4.2.1 Classes

As cinco classes do BRA-Dataset (que correspondem aos animais com mais riscos de acidentes (CYMBALUK, 2018; Erika, 2021)) são: Anta (*Tapirus terrestris*), Jaguarundi (*Herpailurus yagouaroundi*), Lobo Guará (*Chrysocyon brachyurus*), Onça Parda (*Puma concolor*) e Tamanduá Bandeira (*Myrmecophaga tridactyla*). Estes animais frequentam matas selvagens próximas às estradas e acabam sendo atropelados, devido ao atordoamento da visão causado pelos faróis dos veículos no período noturno e do barulho, que assusta estes animais ao meio da travessia.

A **Anta** (Figura 29) é um mamífero de grande porte, onde pode medir entre 1,70 a 2,00 metros de comprimento e pode pesar até 300 kg. Ela é comumente frequente nos biomas Pantanal e Mata-Atlântica brasileira e seus hábitos são solitários. É geralmente vista durante o período noturno (SUBIRá *et al.*, 2018).

Outra classe é o **Jaguarundi** (Figura 30). É um felino de médio porte com aparência bastante semelhante a um gato, porém distinta e pode variar de espécies com coloração escura (em biomas de floresta) e clara (em biomas secos). É ativo durante o dia e habitam planícies de

[1] O conjunto pode ser acesso pelo link do repositório: <https://github.com/GabrielFerrante/BRA-Dataset>

Figura 29 – Exemplos de instâncias da classe Anta

Fonte: Elaborada pelo autor.

florestas tropicais úmidas, onde, apesar de serem trepadores e saltadores, passam a maioria do tempo no chão (SUBIRá *et al.*, 2018).

Figura 30 – Exemplos de instâncias da classe Jaguarundi

Fonte: Elaborada pelo autor.

O **Lobo-guará** (Figura 31) é um animal típico do bioma Cerrado e é o maior canídeo da América do Sul, podendo atingir até um metro de altura. Essa espécie é tímida, solitária e

muito amistosa. É avistado normalmente em grandes campos no entardecer e no período noturno, acostumado a cruzar estradas onde muitas vezes é atropelado (SUBIRá *et al.*, 2018).

Figura 31 – Exemplos de instâncias da classe Lobo-guará

Fonte: Elaborada pelo autor.

A **Onça Parda** (Figura 32) é o segundo maior felino do Brasil, podendo chegar a um comprimento de 1,55 metros. Além disso, é o mamífero terrestre com a maior distribuição geográfica no ocidente. O seu ambiente natural inclui desde os locais desérticos, com o clima subártico, tropical e até mesmo as florestas densas, também comum em fazendas, áreas agrícolas e rodovias devido ao desmatamento (SUBIRá *et al.*, 2018).

Os **Tamanduás-bandeiras** (Figura 33) são animais que possuem um hábito terrestre e solitário. Nos casos das fêmeas com seus filhotes, durante o período de amamentação, e da época de reprodução, os tamanduás não ficam sozinhos. Podem ter atividade ao longo do dia e da noite, dependendo da temperatura e da chuva. Pode chegar a medir até 1,20 metros de comprimento e pode ser facilmente reconhecido pelo seu tamanho e focinho cilíndrico. No Brasil, os tamanduás-bandeira estão presentes em todos os biomas. E um dos maiores motivos pela redução da espécie é o crescimento da malha rodoviária (SUBIRá *et al.*, 2018).

4.2.2 Criação

Para construir o conjunto (etapa 1A), foi utilizada a seguinte abordagem apresentada na Figura 34. Foi utilizado o Google-Imagens como motor de busca. O uso do GoogleImages foi escolhido, pois, o acesso aos animais em parques ecológicos no momento da criação estava

Figura 32 – Exemplos de instâncias da classe Onça Parda

Fonte: Elaborada pelo autor.

limitado devido às questões sanitárias da pandemia e a captura destes animais nas rodovias são inviáveis, pois não há previsão de quando e onde eles aparecerão. Foi pesquisado por cada animal individualmente e para cada resultado, é aplicado dois *scripts*. O primeiro foi feito na linguagem Javascript, que espera o usuário realizar a rolagem das imagens resultantes da busca até o ponto em que o usuário ache necessário, assim, o *script* faz uma listagem das URLs de cada imagem entre o intervalo entre o ponto com o início da página, ao final, ele grava estas URLs em um arquivo de texto (etapas (1) e (2)). O outro foi construído em Python para realizar requisições de *download* para estas URLs listadas, separando para cada animal, uma pasta exclusiva para suas imagens (etapas (3) e (4)).

Após a execução dos *scripts* e com as imagens já baixadas, foi necessário efetuar uma limpeza nestas imagens. A limpeza dos dados (etapa (5)) é importante devido à possibilidade de ter sido baixadas algumas relacionadas ao nome do animal além do próprio animal, como, por exemplo, imagens de marcas de produtos, de objetos que possuem nomes semelhantes, localizações ou pessoas. Após essa limpeza nos dados, restaram 403 imagens de Antas com 478 rótulos, 311 imagens de Jaguarundis com 341 rótulos, 340 imagens de Lobos Guarás com 410 rótulos, 476 imagens e 495 rótulos de Onças Pardas e 293 imagens com 336 rótulos de Tamanduás Bandeira. Totalizando 1823 imagens com 2060 rótulos, com uma média de 364 de imagens por classe aproximadamente (a distribuição de imagens por classe pode ser observada na Figura 35). A Tabela 3 apresenta a média e o desvio padrão das dimensões das imagens. Devido à grande variedade de fontes, as dimensões não são padronizadas, tendo como desvio padrão para largura de até 828 pixels (na classe Jaguarundi) e 547 para altura (também na classe

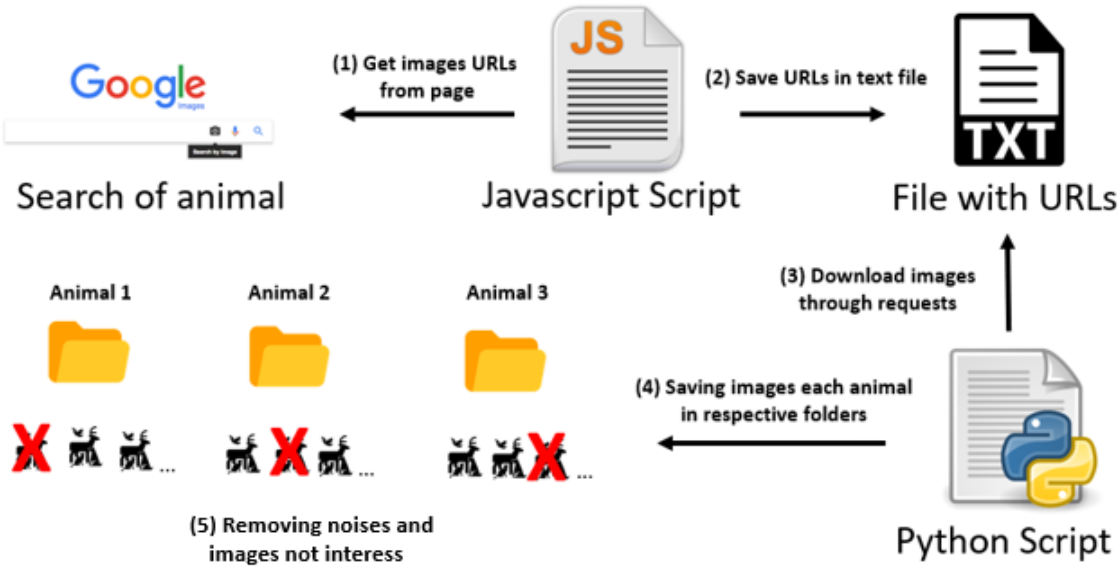
Figura 33 – Exemplos de instâncias da classe Tamanduá-Bandeira

Fonte: Elaborada pelo autor.

Jaguarundi).

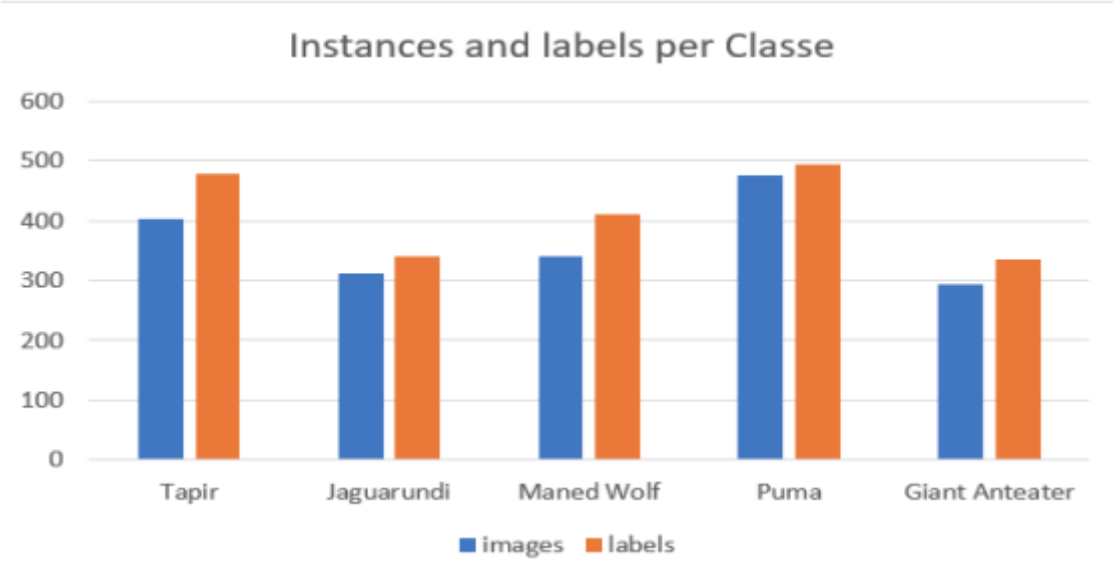
Com as imagens obtidas, é necessário realizar a tarefa de rotulação dos objetos nas imagens para posteriormente ser efetuado o treinamento do modelo supervisionado de detecção. A rotulação de imagens, no contexto de detecção de objetos, consiste em marcar para cada

Figura 34 – Fluxograma da criação do BRA-Dataset



Fonte: Ferrante *et al.* (2022).

Figura 35 – Gráfico de barras sobre o número de instâncias e número de rótulos por classe



Fonte: Ferrante *et al.* (2022).

Tabela 3 – Média (AVG) e Desvio Padrão (STD) das larguras (W) e alturas (H) em pixels das imagens

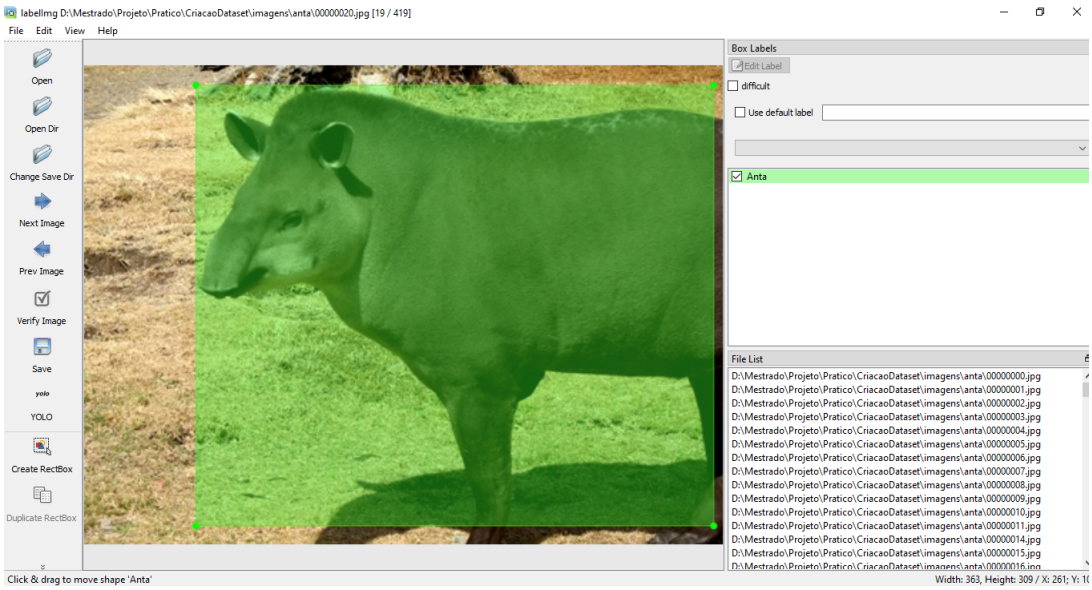
Classe	W AVG	H AVG	W STD	H STD
Tapir	887	619	620	436
Jaguarundi	873	640	828	547
Maned Wolf	926	649	646	462
Puma	900	647	509	400
Giant Anteater	901	621	566	393

Fonte: Elaborada pelo autor.

imagem, onde está o objeto de interesse com uma caixa delimitadora e qual é a classe do objeto em questão. Neste projeto, foi rotulado as imagens conforme os formatos YOLO darknet e PASCAL VOC, ambos os formatos são difundidos pela comunidade de aprendizado de máquina para imagens. A Figura 36 exibe a utilização do programa LabelImg para a rotulação de uma imagem. Basta selecionar os objetos e logo após, selecionar a classe correspondente daquela caixa delimitadora criada. Na versão rotulada em YOLO foi criado um arquivo de texto para cada imagem, contendo as seguintes propriedades: Coordenada X (inicio do retângulo no eixo X da imagem); Coordenada Y (inicio do retângulo no eixo Y da imagem); Largura e Altura do retângulo; Classe do animal detectado.

Algumas imagens podem ter mais de um animal, então para cada animal detectado, terá sua rotulação em linhas separadas no arquivo de texto. Para a versão rotulada em PASCAL VOC, para cada imagem foi criado um arquivo de extensão *Extensible Markup Language* (XML), contendo os seguintes elementos: O diretório da imagem; nome da imagem; caminho do diretório; tamanho de largura, altura e número de dimensões; se a imagem está segmentada ou não; um ou

Figura 36 – Rotulação de uma imagem



Fonte: Elaborada pelo autor.

mais objetos contendo as informações de classe pertencente e coordenadas da caixa delimitadora. Na Figura 36 observa-se que a caixa delimitadora é a área na cor verde sobre a Anta. Essa criação é manual e feita para cada objeto. No canto inferior direito temos as listas das imagens e acima temos a classe daquela caixa com cor verde.

O BRA-Dataset tem como escopo somente imagens adquiridas por meio da metodologia proposta, portanto, o conjunto não teve como foco o uso de imagens com visão noturna ou infravermelho, mas não se limitou a tais imagens, tendo instâncias desse tipo por consequência da busca generalizada. Além disso, a etapa de limpeza dos dados tem como foco manter somente imagens do animal real, portanto, não há a remoção de imagens com ruídos, baixa qualidade ou com poses de animais desfavoráveis.

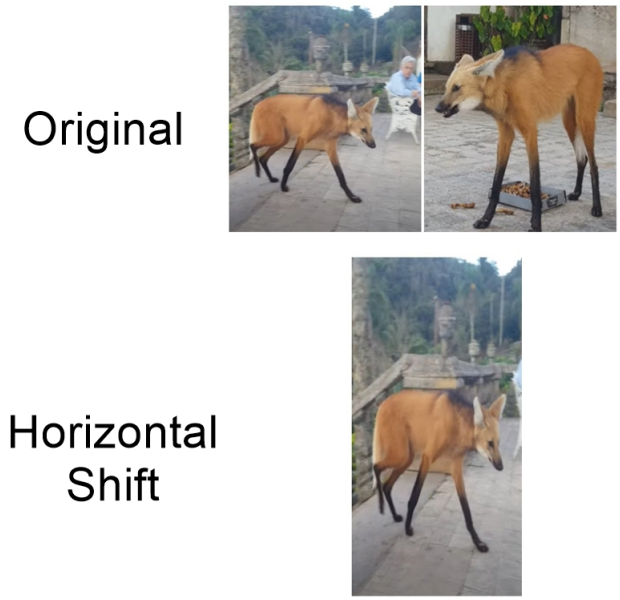
4.3 Aumento de dados

O aumento de dados é a estratégia de acrescentar dados a um conjunto de dados pequeno, tendo como base os dados existentes, para então criar novos dados. Na área de processamento de imagens, as imagens geradas por essa técnica são variações das imagens originais do conjunto. Essas variações são feitas com a aplicação de técnicas de manipulação de imagem, como, por exemplo, transformações geométricas, morfológicas e de agregação e entre outras (MAHARANA; MONDAL; NEMADE, 2022).

Seguindo a metodologia proposta, a etapa 1B foi a aplicação de aumento de dados sobre o BRA-Dataset. Aplicar o aumento de dados foi necessário visto que modelos treinados com poucos dados são pouco generalistas devido a pouca quantidade de amostras de imagens e ajuda a evitar o problema de *overfitting* sobre conjuntos de dados pequenos, como o BRA-Dataset. O aumento

de dados foi realizado somente sobre os dados de treinamento, que foram 80% dos dados do conjunto total, totalizando 1458 imagens. As seguintes técnicas foram aplicadas: *Horizontal Shift, Vertical Shift, Horizontal Flip, Vertical Flip* e *Rotation*. Todas são transformações geométricas clássicas e simples. Também é utilizado indiretamente as técnicas MixUp, CutMix, ColorJitter, Multi-scale, Moisaic e RandomHorizontalFlip, técnicas inclusas na *bag of freebies* (ZHANG *et al.*, 2019) que já são implementadas nas camadas iniciais de convolução de alguns modelos. O aumento de dados escolhido atua como uma forma de regularização (uma técnica para evitar o *overfitting*). Com o aumento de dados, o modelo tem acesso a uma quantidade maior de informações variadas, o que pode impedir que ele se concentre demais em recursos específicos presentes em um conjunto limitado de imagens de treinamento. Ao final, foi obtido um conjunto de treino aumentado com 8748 (incluindo as imagens originais), mas algumas imagens se corromperam durante o processo, portanto o conjunto ficou com 8407 imagens válidas. As Figuras 37, 38, 39, 40 e 41 apresentam exemplos da aplicação das técnicas escolhidas.

Figura 37 – Exemplo de aplicação da técnica *Horizontal Shif* no BRA-Dataset

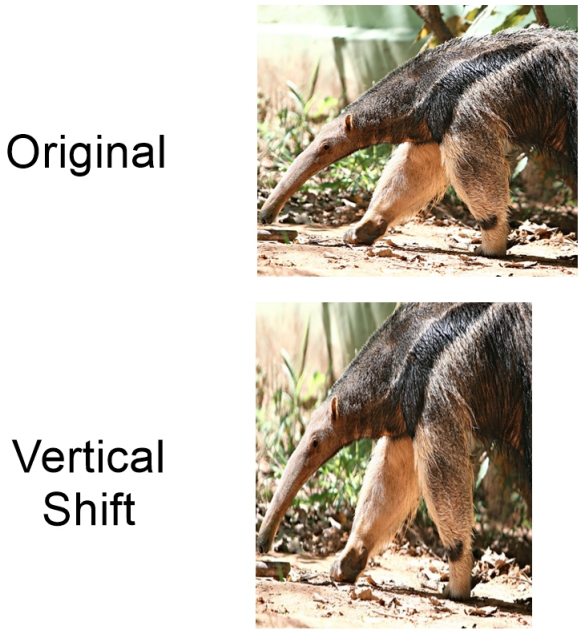


Fonte: Elaborada pelo autor.

4.4 Detecção dos modelos de classificação baseados em YOLO

Com a formulação do BRA-Dataset e aplicação das técnicas de aumento de dados, é proposto uma metodologia de avaliação das diversas arquiteturas recentemente criadas do YOLO. Além disso, há o fornecimento de uma padronização de testes e configurações que são aplicáveis a todos os modelos, tornando a avaliação igual e com mesmas condições.

Figura 38 – Exemplo de aplicação da técnica *Vertical Shif* no BRA-Dataset



Fonte: Elaborada pelo autor.

Figura 39 – Exemplo de aplicação da técnica *Horizontal Flip* no BRA-Dataset

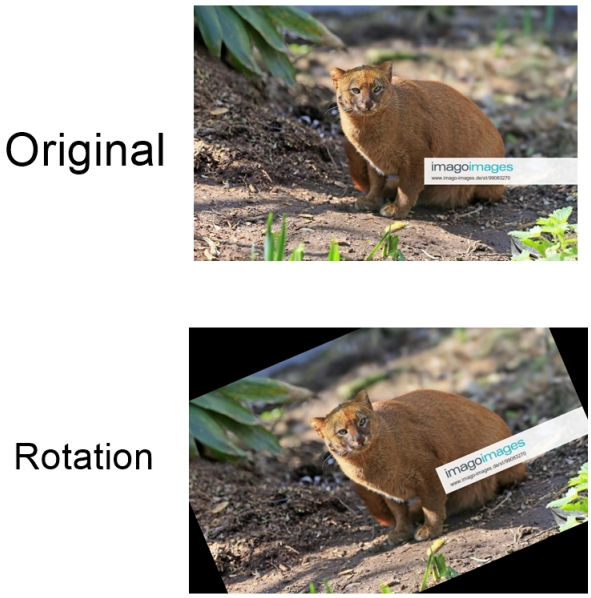
Fonte: Elaborada pelo autor.

O método de detecção foi realizado com três etapas fundamentais, como pode ser visto no fluxograma (Figura 42). A primeira etapa foi a criação/aquisição de um conjunto de dados rotulado com imagens de animais (1A). Além disso, esta etapa contempla a aplicação de técnicas clássicas de aumento de dados sobre o próprio conjunto (1B). Ambas etapas foram descritas anteriormente. Com as imagens do conjunto criado, foi iniciada a etapa de treinamento dos

Figura 40 – Exemplo de aplicação da técnica *Vertical Flip* no BRA-Dataset

Fonte: Elaborada pelo autor.

Figura 41 – Exemplo de aplicação da técnica *Rotation* no BRA-Dataset

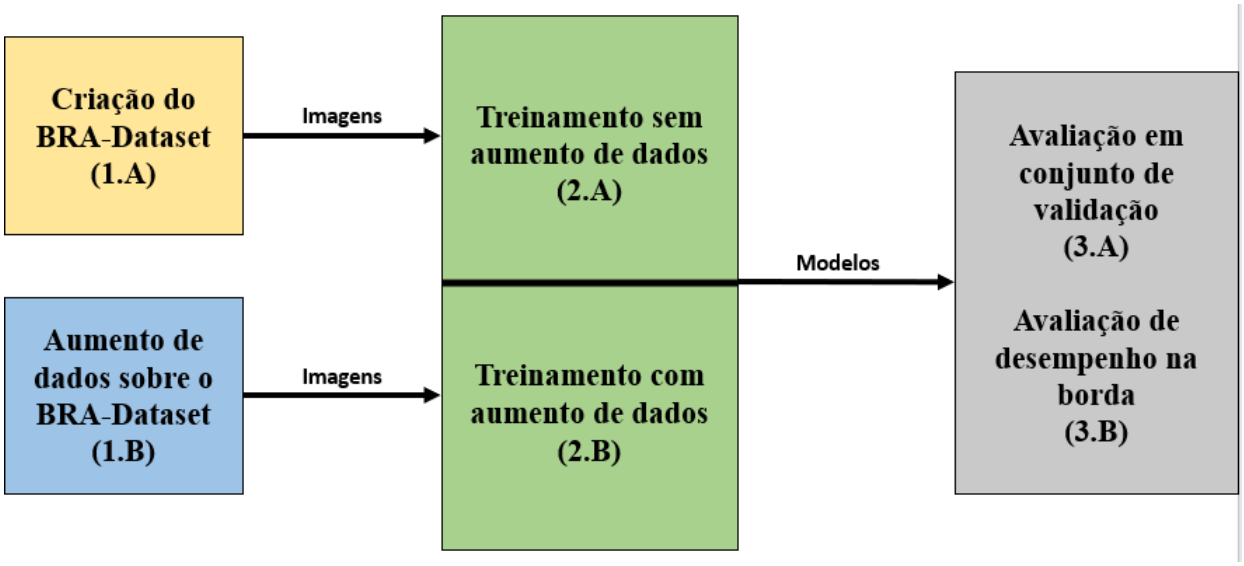


Fonte: Elaborada pelo autor.

modelos com (2A) e sem aumento de dados (2B) e utilizando aprendizado por transferência. Ao final, os modelos criados foram validados com os dados de validação e vídeos (3A), utilizando GPU e vídeos sobre o processamento dos dispositivos de computação de borda (3B).

A etapa de treinamento (2A e 2B) visam realizar o treino dos modelos, utilizando a parte de treinamento do BRA-Dataset original (1458 imagens) e posteriormente com aumento

Figura 42 – Fluxograma geral da proposta



Fonte: Elaborada pelo autor.

de dados (8407 imagens). As duas sub-etapas só possuem como diferença a quantidade de dados, os protocolos de treinamento seguem os mesmos parâmetros. Com isso, foi definido parâmetros iguais de treinamento para todos os modelos, conforme apresentado pela Tabela 4, os hiper-parâmetros de treinamento e seus valores. Todos os treinamentos foram executados utilizando GPU

Tabela 4 – Hiper-parâmetros de treinamento dos modelos utilizados.

Parâmetro	valor
Épocas	100
Batch	32
Max batches	10.000
Learning Rate	0.001
Tamanho da imagem de entrada	416 x 416
Kernels de convolução	20

Fonte: Elaborada pelo autor.

O primeiro hiper-parâmetro é o número de épocas de treinamento. Uma época é quando um conjunto de dados inteiro é executado através da rede neural apenas uma vez para a modificação dos pesos. À medida que o número de épocas aumenta, mais vezes os pesos são alterados na rede neural. O número de épocas deve ser um número equilibrado para não haver o problema de sobre-ajuste (*overfitting*). O valor de 100 épocas foi definido inicialmente observando com alguns treinamentos preliminares nas quais, a centésima época era onde a curva de perda começava a regredir, tendendo ao sobre-ajuste. É importante ressaltar, que os treinamentos com o BRA-Dataset original tem maiores tendências ao problema de sobre-ajuste, devido a poucas imagens

por classe. Isso pode ser afirmado considerando a quantidade de imagens por classe de outros conjuntos normalmente utilizados para comparativos. O segundo hiper-parâmetro é o número de *batch*, que representa a quantia de partes de seu conjunto de dados que serão carregados na memória da GPU para execução e depende da quantidade de memória para ser definido. Quanto maior o valor, mais imagens vão ser lidas pela GPU, aumentando a velocidade de treinamento.

Logo após, é configurado o parâmetro *max_batches*, que é o número máximo de *batchs* para executar no treinamento. O valor escolhido foi de 2000 vezes a quantidade de classes do BRA-Dataset, é um valor padrão e controla quando o treino estará completado, limitando a quantidade de vezes que cada batch é executado por época. O próximo parâmetro é a taxa de aprendizado (*Learning Rate*) por época. Ele determina o tamanho da passo do algoritmo otimizador em cada época enquanto se move em direção a um mínimo global de uma função de perda[2] na decida do gradiente[3]. Também é definido as dimensões das imagens de treinamento para 416 x 416. O tamanho da imagem influência tal como o tamanho do *batch* na velocidade do treinamento. Imagens de entrada com dimensões grandes (*High Definition* (HD), FULL HD e superior) exigem mais memória da GPU. Portanto, as dimensões foram escolhidas para obter maior velocidade do treinamento. Por fim, o último parâmetro definido foi da quantidade de filtros de convolução (*kernels*) aplicados nas camadas de convolução das redes. Foi definido um total de 20 filtros por camada para a captura das características.

Para melhorar a desempenho da velocidade de aprendizado do modelo durante o treinamento, foi utilizado a estratégia de aprendizado por transferência com modelos pré-treinados sobre o conjunto MS COCO[4]. Esta estratégia permitiu que os novos modelos em questão possam aprender os primeiros filtros de convolução de forma rápida, pois os modelos pré-treinados fornecem um mapeamento de seus treinamentos, que podem ser utilizado por outros modelos de arquitetura YOLO. Alguns modelos permitem que possa ser observado algumas amostras de *batchs* formados para treino, como a Figura 43 apresenta.

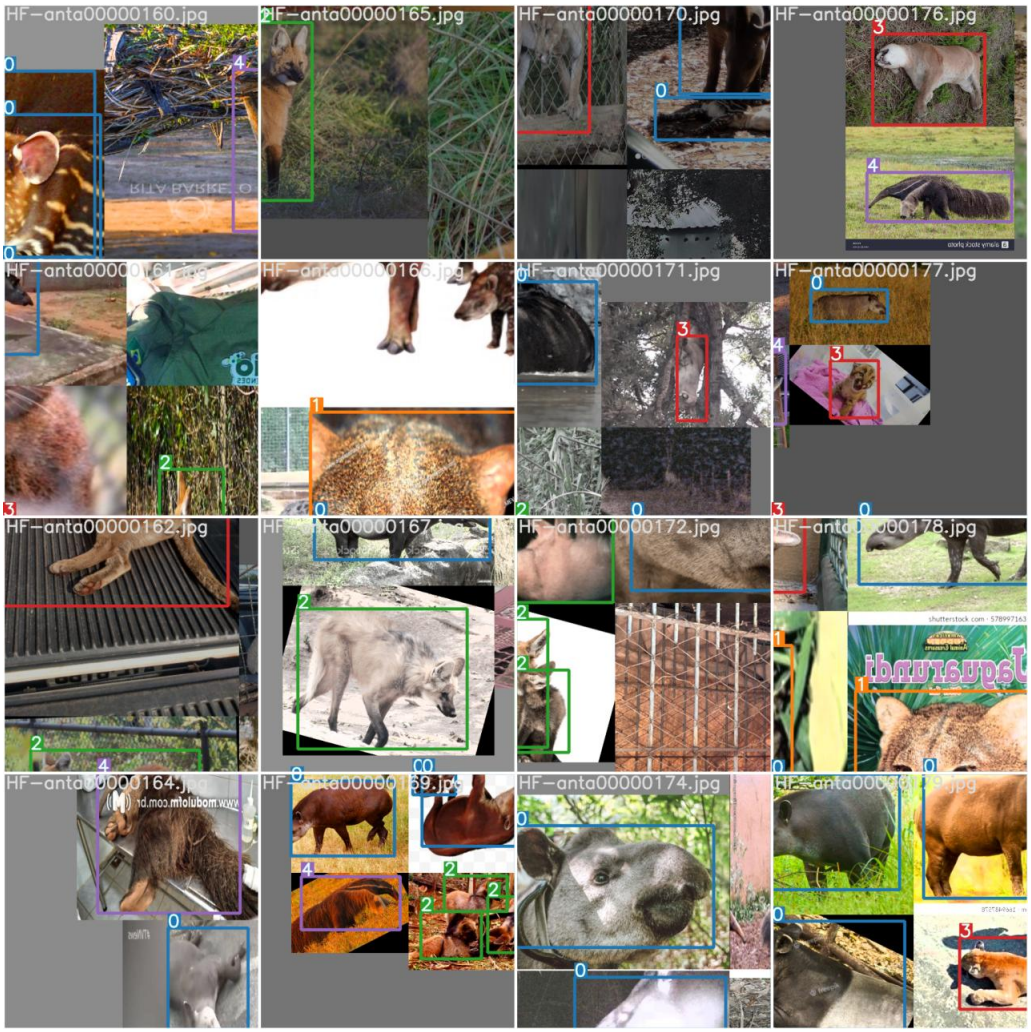
As arquiteturas e suas versões treinadas podem ser observados na Tabela 5. Foram utilizadas as funções de ativação padrões. Algumas arquiteturas por serem construídas tendo como linha de base outras, algumas funções também foram aproveitadas, exceto o YoloV4 e YoloR. Somente a versão YoloV5 foi treinada com pesos menores (Nano e Small) devido

[2] É uma função que compara as pontuações das classificações corretas, afim de determinar o quão satisfeita ela está com o resultado. Valores muito altos indicam uma maior tendência ao erro de classificações, portanto, uma função de perda sempre está associada há um algoritmo de otimização, para procurar seu mínimo valor. Para mais detalhes, acesse (URL): <https://medium.com/neuronio-br/principais-conceitos-por-tras-do-machine-learning-a4b942d5d309>

[3] A descida do gradiente é uma ferramenta padrão para otimizar funções complexas iterativamente em um *software*, de modo a sempre encontrar o mínimo da função (no contexto de redes neurais, é buscar o mínimo da função de perda). Para mais detalhes, acesse (URL): <https://www.deeplearningbook.com.br/aprendizado-com-a-descida-do-gradiente/>

[4] *Common Objects in Context* (COCO) é um conjunto de imagens, referência para tarefas de detecção e segmentação de imagens. Amplamente utilizado para comparativos entre detectores de objetos, possuindo 80 classes e mais de 330 mil imagens e 200 mil rótulos. (LIN *et al.*, 2014)

Figura 43 – Exemplo de um *batch* de treinamento com a aplicação de aumento de dados de agregação automático.



Fonte: Elaborada pelo autor.

que nem todas as arquiteturas forneciam pesos leves. Além disso, manter estes pesos leves do YoloV5, permite observar a discrepância de seus desempenhos com modelos medianos e largos. Portanto, os modelos leves do YoloV5 foram utilizados como "representantes"de modelos leves em geral. No geral, são treinadas as versões com redes menores e versões com redes grandes e complexas (por exemplo, YoloV5-N, YoloV5-S, YoloV5-M , YoloX-M, YoloV5-L, YoloX-L, Scaled-YoloV4-P5, YoloV5-X, YoloX-X, Scaled-YoloV4-P6 e YoloV7-X). Os modelos com nomenclatura contendo “N”, “S”, “M”, “L” e “X”, referem-se à profundidade e complexidade da rede, como “Nano”, “Pequeno”, “Médio”, “ Large"e "eXtreme", respectivamente. Para Scaled-YoloV4, o “p5, p6” no final do nome do modelo indica o número de camadas de escala de imagem. Mas, para YoloR, a versão padrão contém “p6” na nomenclatura, mas não se refere à profundidade.

Tabela 5 – Modelos e suas versões escolhidas para treinamento.

Arquitetura	Versão	*Loss Function* Padrão
YoloV4	Original	Mish
Scaled-YoloV4	P5, P6	LeakyReLU, Mish
YoloV5	Nano, Small, Medium, Large, Extra large	SiLU
YoloR	P6	Sigmoid
YoloX	Medium, Large e Extra large	SiLU
YoloV7	Original, Extra large	ReLU, SiLU

Fonte: Elaborada pelo autor.

4.4.1 Avaliação sobre o BRA-Dataset

A etapa de avaliações (3A e 3B) tem por objetivo avaliar os modelos criados, para averiguar se o modelo possuí generalização para novos dados. Um bom desempenho é percebido quando tanto a tarefa de detecção quanto a tarefa de classificação são alcançadas, ou seja, as caixas delimitadoras são desenhadas corretamente entorno do animal e a rotulação (classificação) dos animais detectados são os corretos, procurando evitar o problema de falsos positivos e falsos negativos. Para avaliar, é usado as métricas de precisão, revocação e média da *Average Precision* (mAP). Nas quais serão discutidas no capítulo 5.

Ao final da etapa de avaliação, será indicado a versão da arquitetura YOLO com melhor desempenho nos cenários utilizando GPU e sobre o dispositivo de computação de borda, referente as relações de custo computacional. No próximo capítulo, é informado com mais detalhes os procedimentos, métricas e infraestrutura de avaliação e testes.

CAPÍTULO

5

AVALIAÇÃO

Este capítulo apresenta as métricas definidas e resultados obtidos das etapas de avaliação (3A e 3B) apresentadas anteriormente. Essa avaliação tem por objetivo verificar quais entre os modelos do YOLO apresenta um melhor desempenho para o cenário proposto. Um bom desempenho é definido quando durante a aplicação do detector, tanto a tarefa de detecção quanto a tarefa de classificação são alcançadas, ou seja, as caixas delimitadoras são previstas corretamente entorno do animal e a rotulação (classificação) dos animais detectados são os certos.

5.1 Métricas de avaliação

Foram utilizadas as métricas de precisão, revocação e média da *Average Precision* (mAP). Para entendimento de cada métrica, é necessário entender o conceito por trás de uma matriz de confusão (Figura 44). A matriz de confusão é comumente utilizada na área de aprendizado de máquina e profundo e permite observar a relação de acertos e erros de um modelo comparando o que foi detectado/classificado com o valor real. É dividida em quatro valores que demonstram a eficácia de seu modelo de inteligência artificial. O primeiro valor é o número de verdadeiros positivos (VP), ou seja, os acertos de valores positivos, no caso deste trabalho, um valor positivo é quando um animal foi detectado e classificado corretamente. O segundo valor é o falso negativo (FN), que representa o número de erros que modelo teve em situações onde a presença dos animais são reais. O terceiro valor é sobre a quantia de falsos positivos (FP), que significam que a detecção detectou objetos inexistentes no ambiente. E por último, há o número de verdadeiros negativos (VN), que simboliza as detecções não realizadas em ambientes que não possuem nenhum objeto, ou seja, um acerto.

Seguindo a relação de valores da matriz de confusão, a métrica de precisão (P) (Equação 5.1), que verifica dentre todas as detecções que o modelo diz haver um objeto de determinada classe, quantas de fato estão corretas. A precisão é bastante utilizada para situações com ênfase

Figura 44 – Matriz de Confusão

		Detectada	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Fonte: Rodrigues (2021).

em mitigar falsos positivos (RODRIGUES, 2021). A revocação (*Recall*) (Equação 5.2) é uma métrica bastante associada a precisão, pois tem como foco situações onde os falsos negativos têm uma grande impacto no problema, ao contrário da precisão. Ela analisa dentre todos os cenários onde era previsto a detecção de um objeto, quantas foram assertivas pelo modelo (RODRIGUES, 2021).

$$P = \frac{VP}{VP + FP} \tag{5.1}$$

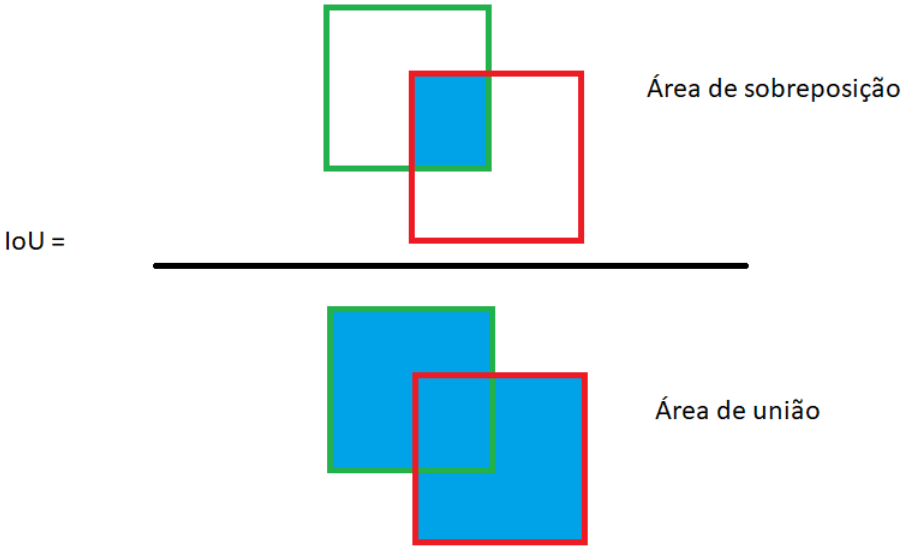
$$Recall = \frac{VP}{VP + FN} \tag{5.2}$$

Partindo para métricas mais específicas da área de detecção objetos, temos a intersecção sobre união (*Intersection Over Union*) (IoU), que compara as caixas delimitadoras corretas em determinada imagem com as caixas delimitadoras criadas pelo sistema de detecção. A Equação 5.3 apresenta como é calculado. A área de sobreposição (AS) é dividida pela área de união (AU). A área de sobreposição é a área em comum entre a caixa delimitadora prevista e a caixa delimitadora de verdade. Já a área de união é a mistura das áreas das duas caixas. Um exemplo ilustrado é apresentado na Figura 45, onde a caixa de cor verde representa a caixa delimitadora verdadeira e a caixa vermelha a detectada. O perímetro azul são as respectivas áreas da Equação 5.3. Quanto maior o valor do IoU, mais próxima à caixa delimitadora prevista está de ser igual à caixa correta do objeto (ROSEBROCK, 2016). Ela é utilizada para avaliar os acertos e erros na tarefa de detecção, permitindo a criação da matriz de confusão que será utilizada como base para as outras métricas.

$$IoU = \frac{AS}{AU} \tag{5.3}$$

Outra métrica específica da área de detecção de objetos é a métrica de precisão média. A Equação 5.4 apresenta a média da *Average Precision* , onde Q é o número total de imagens de seu conjunto de dados. q se refere a um dado do conjunto, AP sendo uma equação que recebe q

Figura 45 – Exemplo ilustrativo da métrica de intersecção sobre a união



Fonte: Elaborada pelo autor.

para calcular sua precisão média. Para cada precisão média calculada para cada dado, é feito a soma e depois a divisão pela quantidade de elementos do conjunto.

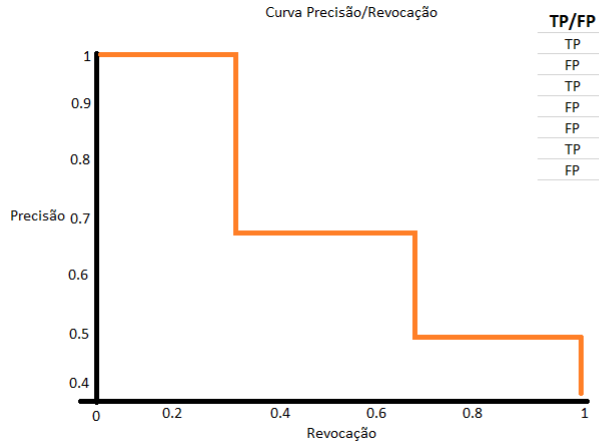
$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{5.4}$$

Para calcular o AP Equação 5.5, é preciso primeiro calcular a curva de precisão e revocação (PR). A curva PR é gerada com o valor da precisão interpolada de cada imagem, a precisão e a revocação. A precisão interpolada Equação 5.6 é calculada em cada nível de revocação da curva r, recuperando a precisão máxima medida para o r em questão definida como $(\tilde{r})$. Interpolar a curva PR reduz o impacto das pequenas variações na classificação das detecções (*outliers*). Tendo o valor da precisão interpolada de cada imagem, é feito a curva com o eixo vertical representando a precisão e o eixo horizontal representando a revocação, ambos com o intervalo de valores de 0 a 1. Para os valores de revocação, são fragmentados em 11 partes (Equação 5.5), que são depois somados para achar o valor de AP. Um exemplo de curva PR com 7 detecções (3 verdadeiras positivas e 4 falsas positivas), onde suas precisões, precisões interpoladas e revocações justificam a curva observada na Figura 46. Com o AP de cada detecção, é feito a média dos AP, assim chegando ao valor do mAP (TAN, 2021).

$$AP = \frac{1}{11} \sum_{Recall_i} Pinterp(Recall_i) = 1 \tag{5.5}$$

$$Pinterp(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r}) \tag{5.6}$$

Figura 46 – Exemplo ilustrativo da curva PR de 7 detecções



TP/FP	Precisão	Revocação	Precisão Interporlada
TP	1/1 = 1	1/3 = 0.33	1
FP	1/2 = 0.5	1/3 = 0.33	1
TP	2/3 = 0.67	2/3 = 0.67	0.67
FP	2/4 = 0.5	2/3 = 0.67	0.67
FP	2/5 = 0.4	2/3 = 0.67	0.67
TP	3/6 = 0.5	3/3 = 1	0.5
FP	3/7 = 0.43	3/3 = 1	0.5

Fonte: Tan (2021).

A avaliação do desempenho do dispositivo de borda com os modelo criados utilizou a métrica de *Frames per Second* (FPS), que calcula a quantidade de imagens por segundo que consegue exibir/mostrar. Esta métrica é importante para determinar se o detector possui uma fluidez no dispositivo e não gere travamentos ou lentidão que podem comprometer um serviço de monitoramento em tempo real, seja ele em movimento ou não. Também foi avaliado o consumo de CPU e memória RAM para a execução dos algoritmos.

Neste trabalho, a métrica dada mais importância é a revocação, que, com valores altos, a detecção de falsos negativos é maior. A revocação é mais relevante que a precisão neste trabalho, pois se um animal estiver na pista e o detector não detectar, é mais prejudicial a sistemas de suporte do que um modelo que tenha mais enfase na detecção de falsos positivos (precisão), pois mesmo se um animal não estiver e o modelo apontar que houve incidência de um, os prejuízos de averiguação por parte dos responsáveis do tratamento animal serão mínimos, exceto o tempo gasto. Porém, se um falso negativo ocorre (revocação baixa), o recurso de tratamento deste animal na pista não é utilizado pela falta da informação da ocorrência. Portanto, o animal continua sendo um desconhecido no monitoramento e possivelmente uma suposta vítima de acidentes.

Com as métricas listadas, a fase de avaliação é formada pela aplicação de testes em determinados cenários. Devido à dificuldade em encontrar os animais em questão no contexto do problema, foi proposta a avaliação em laboratório. Duas avaliações foram executadas. A primeira utilizando os modelos treinados sem e com aumento de dados e aplicados no conjunto de validação do BRA-Dataset, com 364 imagens, utilizando GPU dedicada (etapa 3A) para a aquisição das métricas de precisão, revocação, mAP e FPS. Na segunda avaliação (etapa 3B), foi utilizado somente os modelos treinados com aumento de dados para serem aplicados nos dispositivos de computação de borda, a fim de coletar as métricas de FPS, consumo de CPU e memória. O dispositivo escolhido foi um Raspberry Pi 4, com 1GB de memória *random access memory* (RAM). Para avaliação na borda, foi utilizado vídeos dos animais gravados no Parque

Ecológico de São Carlos. Os vídeos também foram aplicados com os modelos utilizando a GPU, a fim de ter uma linha de base do desempenho utilizando GPU e dispositivos de borda.

Os vídeos contêm não só os animais em campo de visão limpo e sem oclusão em seus recintos, mas também contendo alguns desafios de visão computacional. O primeiro é a oclusão, temos dois vídeos com os animais obstruídos por árvores e vegetação, atrapalhando a detecção (Figura 47), além disso, a pose do animal também foi desafiador, uma vez que os modelos foram treinados com a maioria dos dados com os animais na pose frontal e de perfil, mas não de costas ou sentado, por exemplo. Outro desafio obtido durante as gravações é o da camuflagem do animal no cenário e a distância, como pode ser visto na Figura 48. Em situações reais de monitoramento de estradas e em qual bioma se encontra o animal, suas características, principalmente de pelagem, pode gerar falsos negativos. Também testamos os modelos em um vídeo com resolução baixa e cenário chuvoso Figura 50 e em um vídeo gravado em um recinto onde o animal está atrás de uma tela protetora de vidro, gerando muitos reflexos para a câmera.

Figura 47 – *Frame* de uma Anta atrás de uma árvore.

Fonte: Elaborada pelo autor.

5.2 Parâmetros e infraestrutura computacional da avaliação

Como parâmetros de testes, em todas as avaliações, é executado os modelos com limiar de confiança e limiar de IoU de 50%. Além disso, os dados de validação são 20% do BRA-Dataset, tendo em torno de 365 imagens, reservados antes da execução do treinamento dos modelos. Também é configurado para que os modelos não apliquem aumento de dados nos dados de validação, dado que algumas implementações permitem o uso para dados de validação e teste. Abaixo, é apresentado na Tabela 6, as configurações de estrutura de *hardware* para efetuação dos experimentos.

Figura 48 – *Frame* de um Tamanduá-Bandeira camuflado no recinto e distante.

Fonte: Elaborada pelo autor.

Figura 49 – *Frame* de um vídeo com baixa qualidade com uma Anta atravessando uma rodovia em clima chuvoso.

Fonte: Elaborada pelo autor.

Tabela 6 – Configurações de *hardware* para execução da etapa de treinamento e avaliação do projeto

Componente	especificação
Processador	Intel® Core™ i7-10700 (2.9GHz até 4.8GHz, cache de 16M, octa-core)
Placa de vídeo	NVIDIA® GeForce® RTX™3060 com 12GB de GDDR6
Memória RAM	16GB com frequência 2933MHz
Armazenamento	SSD 256GB

Fonte: Elaborada pelo autor.

Figura 50 – *Frame* de um vídeo de uma Onça em um recinto com tela protetora de vidro.

Fonte: Elaborada pelo autor.

CAPÍTULO

6

RESULTADOS

Os modelos criados através dos conjunto de dados com e sem aumento de dados foram avaliados nos dois cenários propostos (execução dos detectores sobre GPU e Raspberry), como mencionado no capítulo anterior, resultando em dados de desempenho perante as métricas já mencionadas. Este capítulo, portanto, apresenta os resultados quantitativos de cada teste, sendo feita uma análise dos resultados, indicando quais arquiteturas obtiveram sucesso ou falhas sobre as detecções desejadas.

6.1 Resultados para avaliação em conjunto de validação com GPU

É possível observar em todas as tabelas dos resultados obtidos que alguns resultados estão com uma flag (*), representando modelos que tiveram suas métricas influenciadas negativamente e por poucos dados, ocasionando altos valores, dado a um leve sobre-ajuste. É um risco que pode acontecer quando se treina modelos supervisionados com poucos dados, devido a pouca variação de características. Esse problema foi identificado observando resultados excepcionais e fora da realidade de modelos de aprendizado de máquina, como, por exemplo, as métricas alcançando valores muito próximas a 100%.

6.1.1 Desempenho dos modelos treinados sem aumento de dados.

Na Tabela 7 é apresentado os resultados dos modelos sem aumento de dados sobre o conjunto de validação. Na grande maioria, os modelos foram afetados negativamente pelos poucos dados fornecidos para treino. A Tabela 8 mostra os resultados específicos para as cinco classes do BRA-Dataset nos modelos treinados sem aumento de dados.

No geral, os modelos fornecidos e com o BRA-Dataset sem aumento permitiram realizar detecções nas imagens do conjunto, porém, quando comparado aos resultados após o aumento

Tabela 7 – Resultados gerais de Precisão, Revocação e mAP@50 para os modelos de detecção treinados sem aumento de dados

Modelo	Precisão	Revocação	mAP@50
YoloV4*	0.96	0.96	97.4
Scaled-YoloV4-p5*	0.98	0.96	95.9
Scaled-YoloV4-p6*	0.97	0.96	96.2
YoloV5-N*	0.97	0.93	96.9
YoloV5-S*	0.98	0.93	96.9
YoloV5-M*	0.97	0.93	96.7
YoloV5-L	0.94	0.91	83.5
YoloV5-X	0.95	0.90	84.3
YoloR-p6*	0.93	0.96	98
YoloX-M	0.65	0.72	65.8
YoloX-L	0.66	0.73	66.8
YoloX-X	0.67	0.72	67.0
YoloV7*	0.72	0.58	65.8
YoloV7-X	0.79	0.70	77.4

Fonte: Elaborada pelo autor.

de dados (próxima subseção), os resultados são discrepantes, ou seja, houve uma "piora"nos modelos, apontando que o sobre-ajuste foi regulado, significando que os modelos que foram afetados diminuíram seus resultados nas métricas e que em suma, se tornaram mais generalistas.

6.1.2 Desempenho dos modelos treinados com aumento de dados.

Logo abaixo, as Tabela 9 e Tabela 10 apresentam respectivamente os resultados gerais e específicos por classe obtidos dos modelos de detecção treinados com as técnicas de aumento de dados sobre os conjunto de validação. É observado haver somente resultados mais consensuais com a realidade de modelos de aprendizado de máquina, não alcançando valores exorbitantes. Além disso, para a classe Anta, em alguns modelos alcançaram 100% de precisão. Para verificar se não houve o sobre-ajuste durante o treinamento, os resultados da validação deveriam ser menores que com o sobre-ajuste, e de fato foram. Por outro lado, a classe Onça-Parda no geral possui os menores resultados em ambos os testes com o conjunto de validação e com uma investigação precisa sobre o conjunto de treinamento, é observado que existe uma abundância de imagens com poses do animal sentado, deitado ou imagens com somente a face do animal, que impacta diretamente na coleta das características da classe, onde a classificação é prejudicada.

No geral, em sistemas de detecção animal em rodovias, existe um grande valor em saber qual é o nível de revocação dos modelos, uma vez que essa métrica enfatiza o problema de ter muitos falsos negativos, ou seja, animais que na realidade estão na rodovia, o modelo não detecta ou classifica. Uma revocação baixa, pode ocorrer de poucas detecções em situações positivas e influenciar diretamente, por exemplo, na contagem de um animal, dificultando estimativas

populacionais dessa classe na região de uma rodovia em específico ou o não alerta de um animal em determinado trecho de rodovia, ocasionando o não registro da classe no local e também um possível resgate.

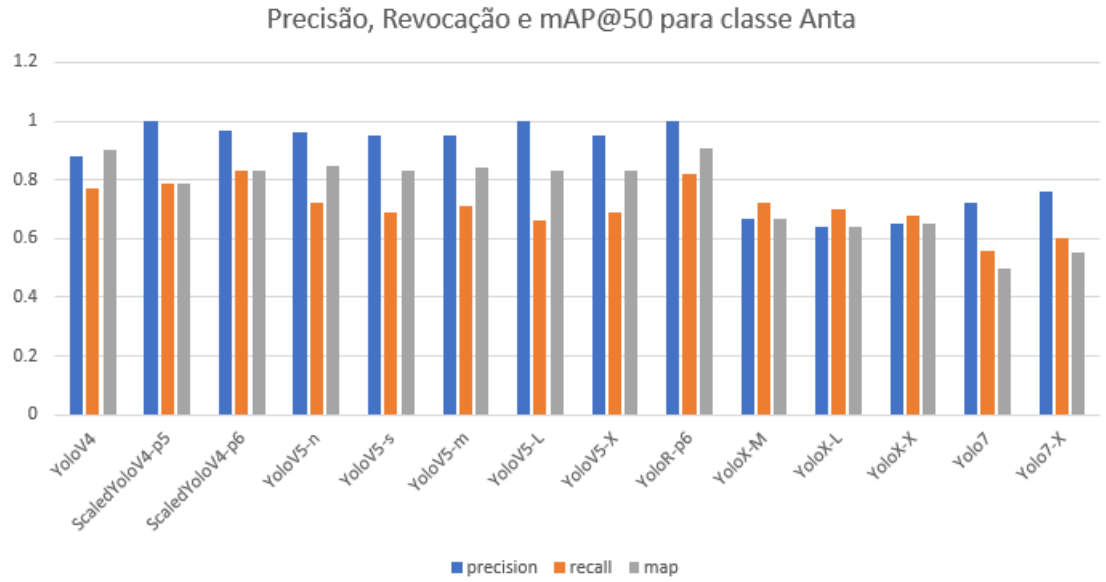
Considerando somente os resultados após a aplicação de aumento de dados, é observando exclusivamente a métrica revocação, o modelo que obteve o melhor desempenho geral contra falsos negativos é o Scaled-YoloV4-p6 e o pior é o recente YoloV7. O fato de Scaled-YoloV4 obter os melhores resultados em geral pode ser atribuído à capacidade de manipular diferentes escalas de uma mesma imagem, permitindo obter melhores características, no entanto, esta capacidade reflete a baixa velocidade de execução em comparação com as outras versões. Quanto ao YoloV7, observa-se que devido à sua capacidade de aplicar o aumento de dados nativos usando técnicas de combinação e agregação, ele fornece, em cenários com outras técnicas mais básicas, as representações de características de objetos são sobrepostas, ou seja, a arquitetura é sensível ao uso de ampliação de dados que não fazem parte da bag-of-freebies.

Para a classe Tamanduá-Bandeira, a maior revocação foi alcançado pela arquitetura YoloR. O fato do YoloR ter obtido melhor desempenho para a classe Tamanduá-bandeira reflete a utilização de camadas de unificação de conhecimento, onde podem existir animais da mesma classe com características diferentes (Tamanduá-bandeira preto e marrom, lobo-guará jovem e adulto, jaguarundi com três variações de cores). A estratégia de representação única é uma boa alternativa nesses casos. Para a classe Jaguarundi o detentor da maior revocação foi obtido ao peso YoloV5-X. No caso da classe Jaguarundi, a versão mais pesada do YoloV5 (“x”) proporcionou um resultado um pouco acima das arquiteturas Scaled-YoloV4 e YoloR, que já haviam apresentado bons resultados medianos para outras classes. Isso se deve à característica do YoloV5 de lidar bem com objetos grandes, pois quando é analisado o BRA-Dataset como um todo, muitas imagens desta classe apresentam o animal na maior parte da imagem, ou seja, o animal ocupa o maior espaço de pixels nas imagens. Por outro lado, essa característica das imagens podem dificultar a validação em imagens com o animal pequeno e longe da câmera.

Já nas classes Lobo-Guará, Onça-Parda, e Anta as versões do YoloV4-Scaled alcançaram os maiores desempenhos em revocação. No quesito precisão, o melhor desempenho no geral foi obtido pelo modelo YoloV4-Scaled-p5. Para a métrica mAP, o modelo clássico do YoloV4 ficou em primeiro lugar, seguido do modelo YoloR. Abaixo nas Figuras 51, 52, 53, 54, 55 apresentam os gráficos de barras do desempenho de cada arquitetura para cada classe.

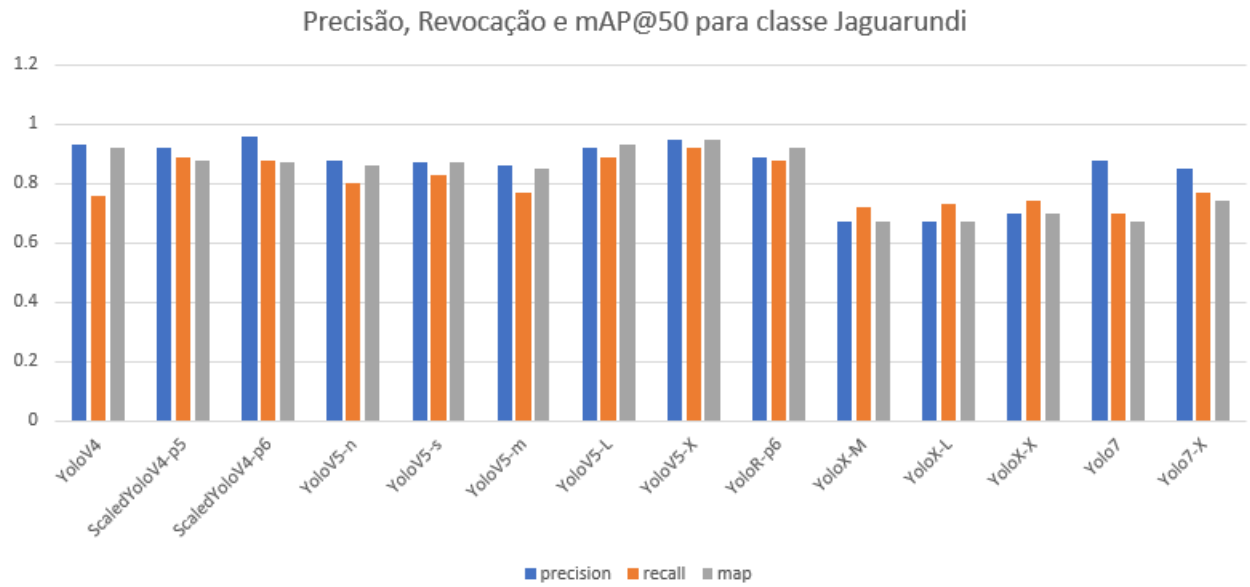
Já a arquitetura com menor desempenho foi o YoloV7, alcançando o valor de 0.56 para revocação na classe Onça-Parda. De outra forma, a melhor revocação foi obtida para a classe Jaguarundi com o modelo o YoloV5-X. Em todas as classes as menores precisões foram obtidas pela arquitetura YoloX, diferentemente das arquiteturas *scaled* do YoloV4 e YoloR que obtiveram altos valores.

Figura 51 – Precisão, Revocação e mAP@50 para classe Anta utilizando os modelos treinados com aumento de dados.



Fonte: Elaborada pelo autor.

Figura 52 – Precisão, Revocação e mAP@50 para classe Jaguarundi utilizando os modelos treinados com aumento de dados.

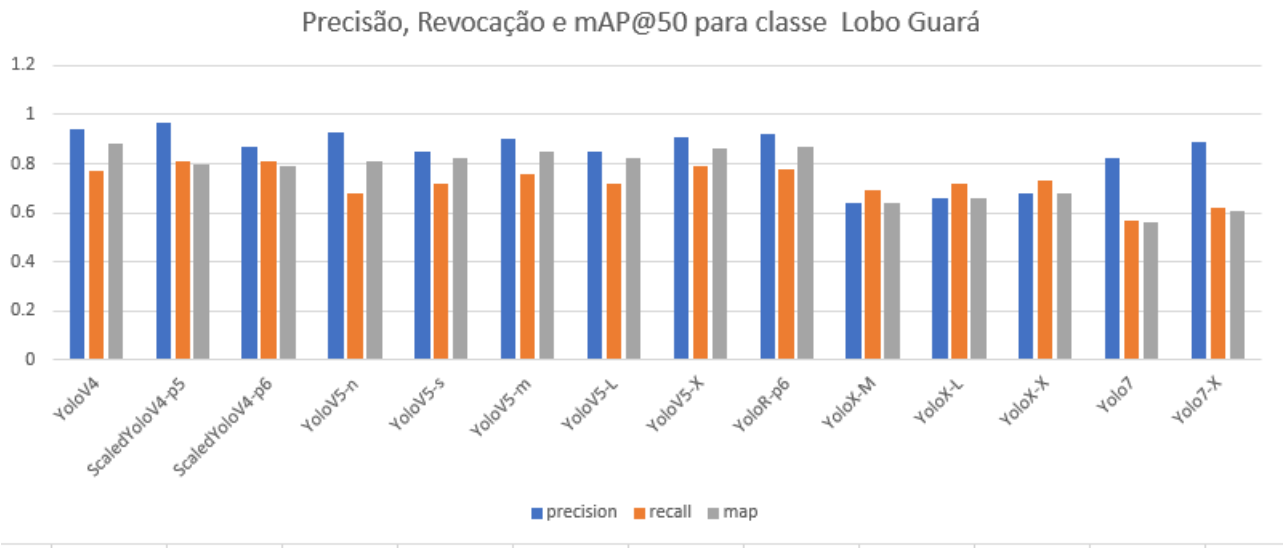


Fonte: Elaborada pelo autor.

6.2 Comparação entre GPU e dispositivos de computação de borda

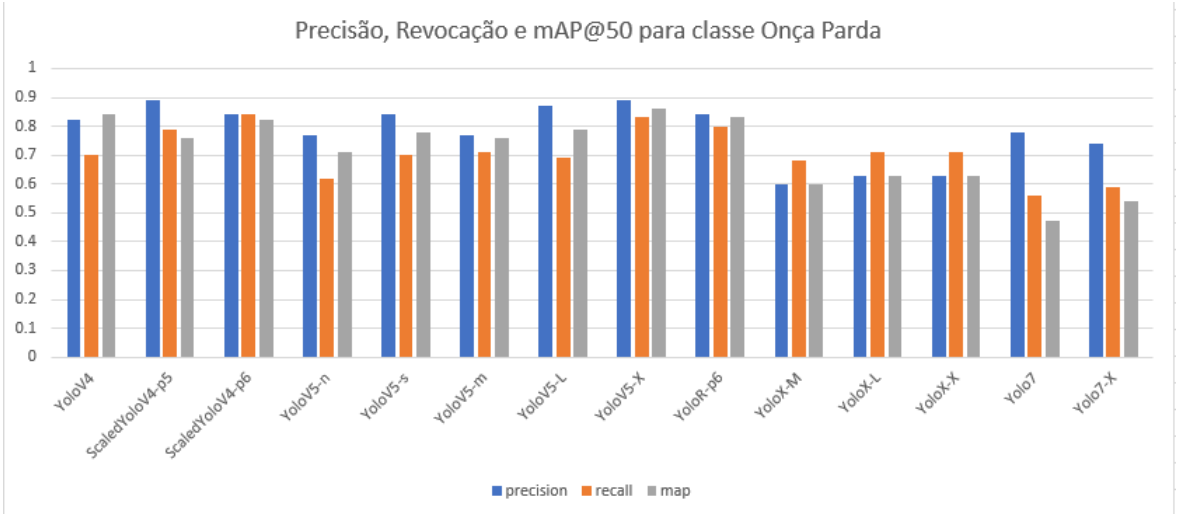
Os testes com vídeos sobre a GPU foram avaliados sobre a média de FPS para cada um dos modelos com aumento de dados (modelos úteis). A Figura 56 apresenta os desempenhos dos

Figura 53 – Precisão, Revocação e mAP@50 para classe Lobo-Guará utilizando os modelos treinados com aumento de dados.



Fonte: Elaborada pelo autor.

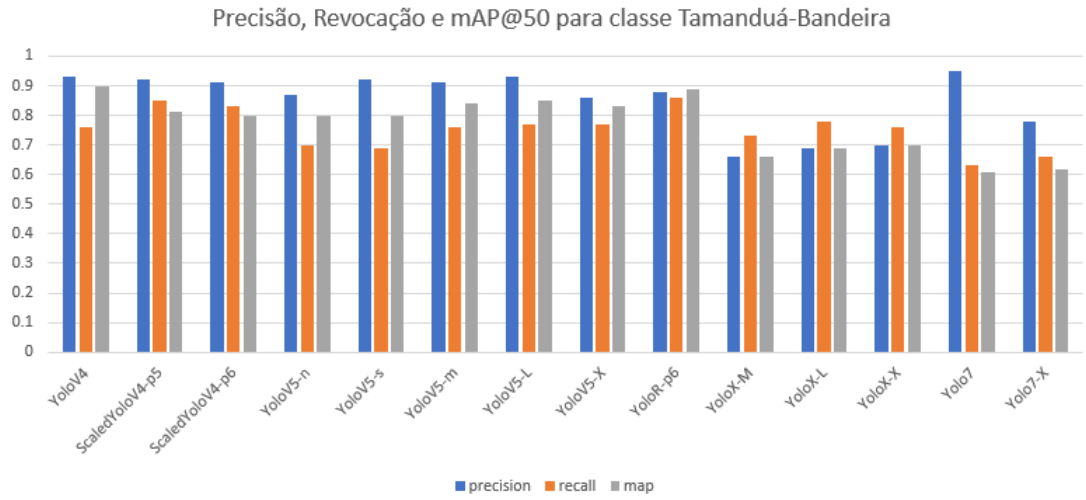
Figura 54 – Precisão, Revocação e mAP@50 para classe Onça-Parda utilizando os modelos treinados com aumento de dados.



Fonte: Elaborada pelo autor.

modelos sobre testes de inferência. No geral, modelos com redes menos complexas e profundas são mais rápidos do que as versões profundas. Em contrapartida, a precisão é maior em modelos complexos. O modelo mais rápido é a versão nano do YoloV5, essa versão é a mais leve e é uma ótima alternativa para dispositivos móveis e computação de borda limitada. Os modelos YoloV5-N e YoloV5-S foram os únicos modelos menores treinados visando apresentar a diferença entre modelos médios e largos, sendo usados como teto de desempenho. Já modelos mais pesados exigiram mais processamento e não alcançaram grandes desempenhos, porém são possíveis de ser usados, destacam-se os modelos YoloV7 e YoloV7X que mesmo sendo complexos atingiram FPS

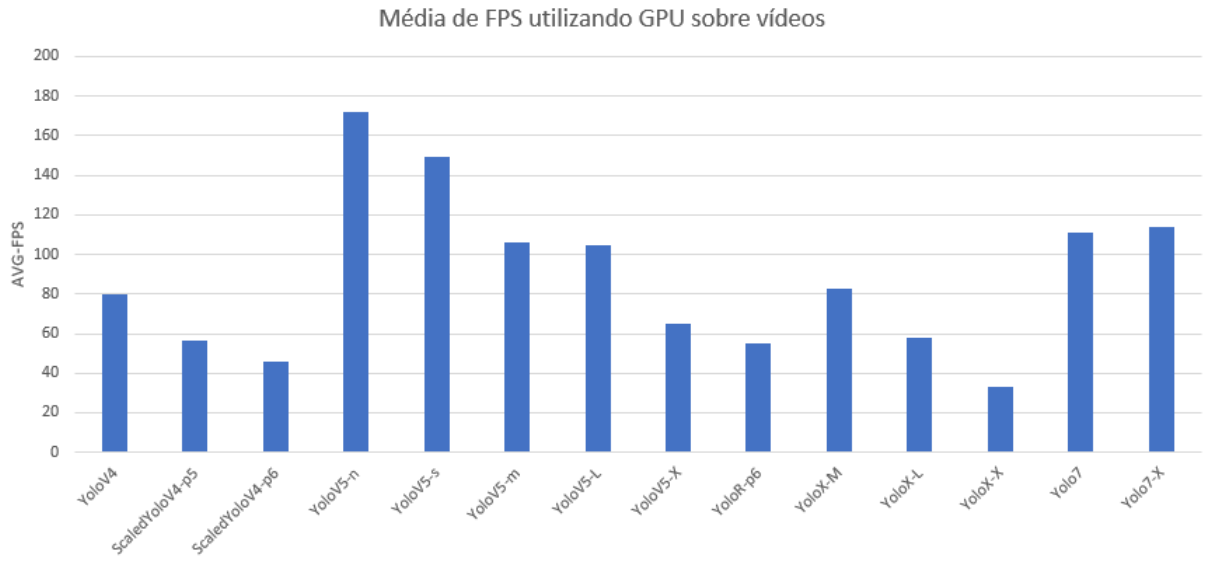
Figura 55 – Precisão, Revocação e mAP@50 para classe Tamanduá-Bandeira utilizando os modelos treinados com aumento de dados.



Fonte: Elaborada pelo autor.

maior que as outras redes médias e largas. A arquitetura do YoloV4 alcançou um desempenho razoável mesmo em comparações com arquiteturas lançadas posteriormente. No geral, mesmo com o YOLO de menor performance (YoloX-X) ainda foi possível obter uma velocidade de execução fluída para inferência.

Figura 56 – Desempenho em FPS sobre GPU dos modelos.



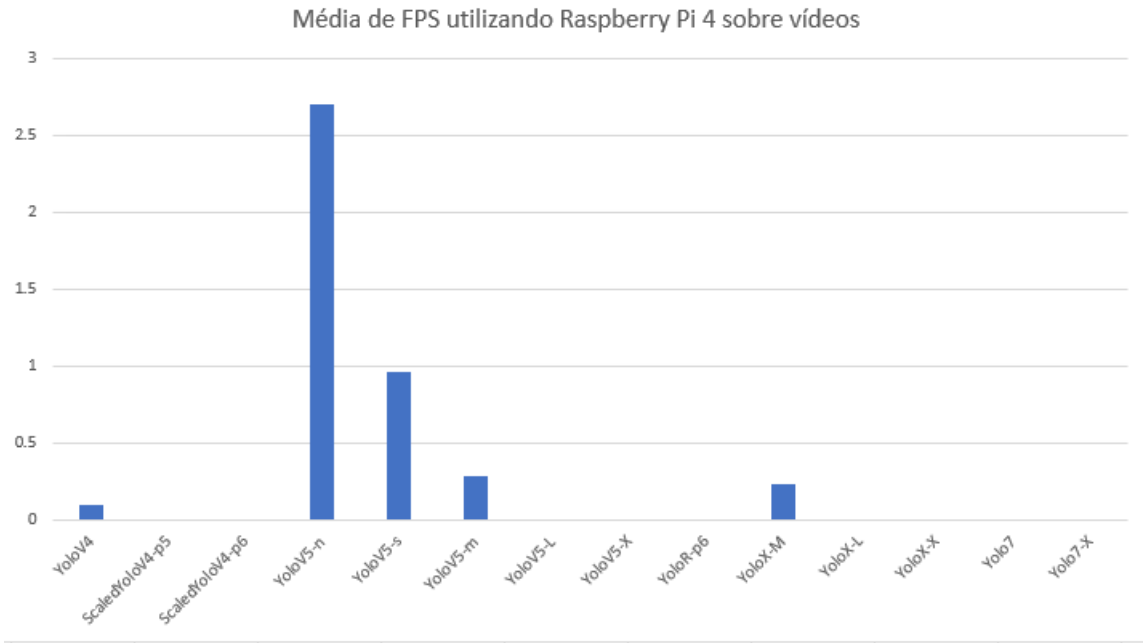
Fonte: Elaborada pelo autor.

Quando os modelos são executados nos dispositivos de computação de borda, o cenário muda drasticamente. Devido a suas limitações de *hardware*, a implantação de I.A na borda sempre foi um desafio. O Raspberry Pi 4 escolhido é a versão mais limitada dentre as variantes, com somente 1GB de memória RAM, portanto, foi testado o dispositivo no seu limite. Para

tal, o vídeo de entrada no dispositivo foi redimensionado para 416x416, pois vídeos com altas dimensões ocuparia muita memória, além dos próprios modelos e do sistema operacional e outros processos em execução, como bibliotecas e *drivers*. Mesmo com a entrada reduzida, muitos modelos consumiram muito além da capacidade de memória do Raspberry, sendo impossível executar, principalmente os mais complexos. Somente cinco modelos conseguiram ser carregados e executados, como apresenta a Figura 57.

O desempenho máximo alcançado foi de 2.7 FPS com o modelo Nano do YoloV5, seguido da versão Small (0.96) e Medium (0.29), o modelo médio do YoloX alcançou 0.23 FPS, somente acima do modelo YoloV4 original, com 0.1 FPS. É possível observar que no geral os modelos mal alcançaram 3 FPS, tendo a sua fluidez de detecção prejudicada em todos os testes. No aspecto de consumo de memória apresentado pela Figura 58, o limiar de consumo que fez a maioria dos modelos não ser executados é de acima de 700 megabyte (MB) (colunas em vermelho). Os modelos que menos utilizam memória foram Nano, Small e Medium do YoloV5. Isso pode ser um indicativo de que essa arquitetura em especial tem vantagens na borda, e que se o dispositivo tivesse mais recursos, os modelos dessa arquitetura no geral utilizam pouco recursos. Já o consumo de CPU durante a execução (Figura 59) foi similar entre os modelos do YoloV5 e YoloX médio, acima de 88% de consumo até 100%. Porém, o modelo YoloV4 utilizou somente 50%, isso se deve devido ao modelo ser construído em linguagem C e não utilizando a linguagem Python e *frameworks*, que deixam a execução mais lenta.

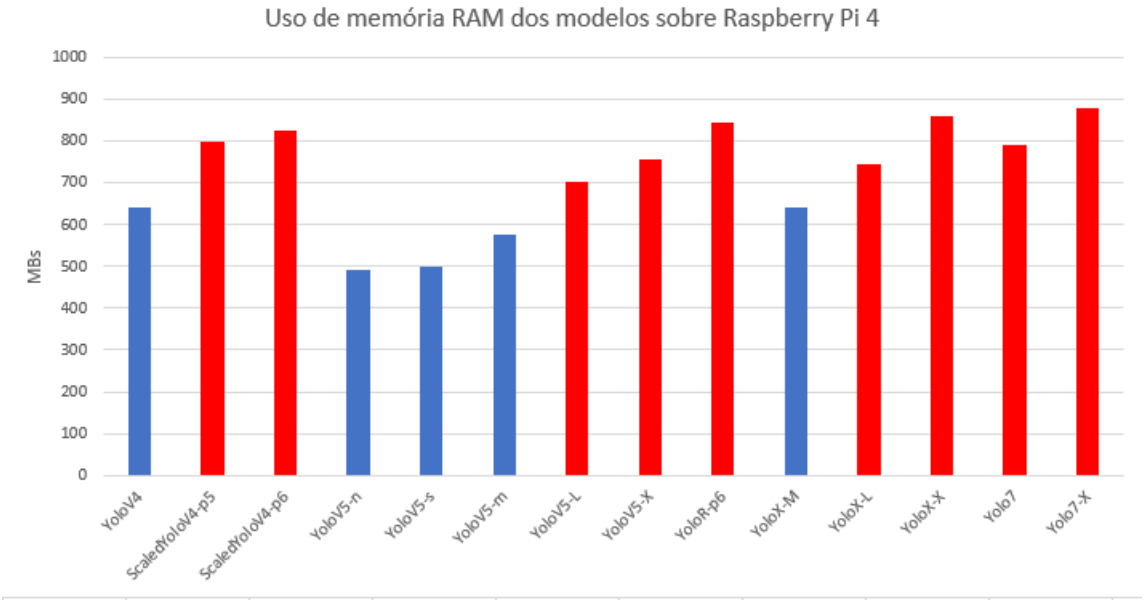
Figura 57 – Média de FPS utilizando Raspberry Pi 4 sobre vídeos.



Fonte: Elaborada pelo autor.

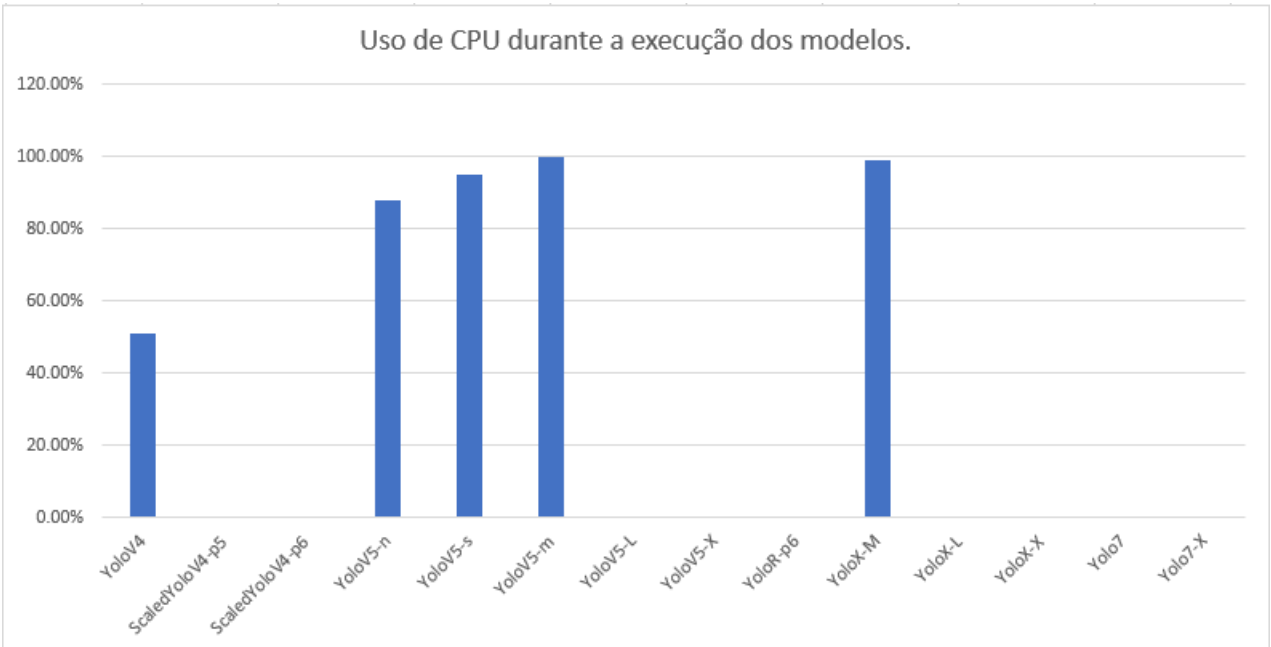
Conforme os resultados perante os testes sobre a infraestrutura em Raspberry, a arquitetura YoloV5-N apresentou maior velocidade de execução, em relação ao custo-computacional.

Figura 58 – Uso de memória RAM dos modelos sobre Raspberry Pi 4.



Fonte: Elaborada pelo autor.

Figura 59 – Uso de CPU dos modelos sobre Raspberry Pi 4.



Fonte: Elaborada pelo autor.

Portanto, é o mais performático para a computação de borda. No aspecto de desempenho em revocação (métrica mais relevante), as arquiteturas Scaled-YoloV4 (p5 e p6) e YoloR foram superiores as demais, evidenciando, em cenários ótimos de detecção, serem as melhores no quesito detecção de objetos. No próximo capítulo são apresentadas as conclusões referentes as hipóteses formuladas e conclusões gerais sobre o estudo referente ao estado de detecção de animais no Brasil.

Tabela 8 – Resultados específicos por classe de Precisão, Revocação e mAP@50 respectivamente para os modelos de detecção treinados sem aumento de dados

Modelo	**Tamanduá-Bandeira**	**Jaguarundi**	**Lobo-Guará**	**Onça Parda**	**Anta**
YoloV4*	0.97 | 0.94 | 96.8	0.93 | 0.97 | 96.8	0.96 | 0.91 | 95.8	0.98 | 0.96 | 98.8	0.97 | 0.99 | 98.8
Scaled-YoloV4-p5*	1.00 | 0.92 | 92.2	0.97 | 0.98 | 98.4	0.97 | 0.93 | 93.4	1.00 | 0.95 | 95.5	0.98 | 1.00 | 99.5
Scaled-YoloV4-p6*	1.00 | 0.94 | 94.5	0.97 | 0.97 | 97.5	0.96 | 0.96 | 96.2	0.97 | 0.94 | 94.5	0.97 | 0.98 | 98.5
YoloV5-N*	0.99 | 0.91 | 95.6	0.95 | 0.94 | 96.2	0.97 | 0.93 | 96.1	0.97 | 0.90 | 97.1	0.96 | 0.98 | 99.4
YoloV5-S*	0.98 | 0.91 | 94.7	0.97 | 0.94 | 97.4	0.97 | 0.93 | 96.0	0.98 | 0.91 | 96.8	0.98 | 0.97 | 99.4
YoloV5-M*	0.99 | 0.92 | 95.9	0.97 | 0.92 | 97.0	0.97 | 0.92 | 95.7	0.95 | 0.92 | 95.7	0.98 | 0.97 | 99.4
YoloV5-L	0.90 | 0.82 | 85.8	0.87 | 0.82 | 88.5	0.81 | 0.78 | 82.0	0.72 | 0.73 | 78.7	0.86 | 0.70 | 82.6
YoloV5-X	0.94 | 0.79 | 89.2	0.87 | 0.86 | 91.0	0.88 | 0.75 | 83.3	0.76 | 0.74 | 77.2	0.86 | 0.76 | 80.9
YoloR-p6*	0.95 | 0.94 | 97.4	0.93 | 0.97 | 98.4	0.94 | 0.95 | 96.3	0.89 | 0.96 | 98.3	0.96 | 1.00 | 99.5
YoloX-M	0.69 | 0.74 | 69.6	0.69 | 0.76 | 69.7	0.64 | 0.70 | 64.0	0.61 | 0.71 | 61.6	0.64 | 0.70 | 64.0
YoloX-L	0.72 | 0.77 | 72.4	0.69 | 0.77 | 69.9	0.66 | 0.72 | 66.4	0.59 | 0.69 | 59.7	0.65 | 0.69 | 65.2
YoloX-X	0.71 | 0.76 | 71.5	0.69 | 0.73 | 69.8	0.66 | 0.71 | 66.2	0.64 | 0.71 | 64.1	0.66 | 0.70 | 66.7
YoloV7*	0.75 | 0.58 | 68.7	0.78 | 0.72 | 78.0	0.79 | 0.65 | 70.1	0.59 | 0.47 | 51.3	0.70 | 0.50 | 60.9
YoloV7-X	0.81 | 0.70 | 80.9	0.85 | 0.77 | 82.7	0.79 | 0.76 | 78.7	0.59 | 0.66 | 66.5	0.93 | 0.59 | 78.1

Fonte: Elaborada pelo autor.

Tabela 9 – Resultados gerais de Precisão, Revocação e mAP@50 para os modelos de detecção treinados com aumento de dados

Model	**Precisão**	**Revocação**	**mAP@50**
YoloV4	0.89	0.75	89.4
Scaled-YoloV4-p5	0.94	0.83	81.4
Scaled-YoloV4-p6	0.91	0.84	82.5
YoloV5-N	0.88	0.71	80.9
YoloV5-S	0.89	0.72	82.4
YoloV5-M	0.88	0.74	83.4
YoloV5-L	0.91	0.75	84.8
YoloV5-X	0.91	0.80	87.0
YoloR-p6	0.90	0.83	88.8
YoloX-M	0.65	0.71	65.3
YoloX-L	0.66	0.73	66.5
YoloX-X	0.67	0.72	67.7
YoloV7	0.83	0.60	56.7
YoloV7-X	0.80	0.65	61.8

Fonte: Elaborada pelo autor.

Tabela 10 – Resultados específicos por classe de Precisão, Revocação e mAP@50 respectivamente para os modelos de detecção treinados com aumento de dados

Modelo	Tamanduá-Bandeira	Jaguarundi	Lobo-Guará	Onça Parda	Anta
YoloV4	0.93 | 0.76 | 90.8	0.93 | 0.76 | 92.7	0.94 | 0.77 | 88.6	0.82 | 0.70 | 84.5	0.88 | 0.77 | 90.6
Scaled-YoloV4-p5	0.92 | 0.85 | 81.9	0.92 | 0.89 | 88.9	0.97 | 0.81 | 80.5	0.89 | 0.79 | 76.3	1.00 | 0.79 | 79.5
Scaled-YoloV4-p6	0.91 | 0.83 | 80.0	0.96 | 0.88 | 87.6	0.87 | 0.81 | 79.4	0.84 | 0.84 | 82.2	0.97 | 0.83 | 83.4
YoloV5-N	0.87 | 0.70 | 80.0	0.88 | 0.80 | 86.5	0.93 | 0.68 | 81.4	0.77 | 0.62 | 71.3	0.87 | 0.70 | 80.0
YoloV5-S	0.92 | 0.69 | 80.3	0.87 | 0.83 | 87.6	0.85 | 0.72 | 82.3	0.84 | 0.70 | 78.1	0.95 | 0.69 | 83.4
YoloV5-M	0.91 | 0.76 | 84.6	0.86 | 0.77 | 85.5	0.90 | 0.76 | 85.5	0.77 | 0.71 | 76.7	0.95 | 0.71 | 84.7
YoloV5-L	0.93 | 0.77 | 85.5	0.92 | 0.89 | 93.4	0.85 | 0.72 | 82.2	0.87 | 0.69 | 79.5	1.00 | 0.66 | 83.3
YoloV5-X	0.86 | 0.77 | 83.7	0.95 | 0.92 | 95.3	0.91 | 0.79 | 86.4	0.89 | 0.83 | 86.0	0.95 | 0.69 | 83.5
YoloR-p6	0.88 | 0.86 | 89.2	0.89 | 0.88 | 92.4	0.92 | 0.78 | 87.4	0.84 | 0.80 | 83.8	1.00 | 0.82 | 91.1
YoloX-M	0.66 | 0.73 | 66.7	0.67 | 0.72 | 67.0	0.64 | 0.69 | 64.5	0.60 | 0.68 | 60.2	0.67 | 0.72 | 67.8
YoloX-L	0.69 | 0.78 | 69.8	0.67 | 0.73 | 67.6	0.66 | 0.72 | 66.6	0.63 | 0.71 | 63.4	0.64 | 0.70 | 64.7
YoloX-X	0.70 | 0.76 | 70.5	0.70 | 0.74 | 70.8	0.68 | 0.73 | 68.6	0.63 | 0.71 | 63.3	0.65 | 0.68 | 65.0
YoloV7	0.95 | 0.63 | 61.9	0.88 | 0.70 | 67.5	0.82 | 0.57 | 56.1	0.78 | 0.56 | 47.7	0.72 | 0.56 | 50.1
YoloV7-X	0.78 | 0.66 | 62.8	0.85 | 0.77 | 74.8	0.89 | 0.62 | 61.4	0.74 | 0.59 | 54.0	0.76 | 0.60 | 55.8

Fonte: Elaborada pelo autor.

CAPÍTULO

7

CONCLUSÃO

Esta dissertação de mestrado apresentou um comparativo das recentes arquiteturas do YOLO para detecção de objetos para o ambiente rodoviário, visando fornecer subsídios técnicos e teóricos na detecção eficaz de animais silvestres em extinção, que possa ser implementado em um ambiente inteligente e alto conectivo, podendo auxiliar na mitigação de acidentes envolvendo animais em rodovias. Também é fornecido como produto do trabalho, um conjunto de dados de imagens, o BRA-Dataset, que aborda cinco classes de animais de médio e grande porte com risco de extinção, validando o potencial de uma metodologia fácil e ágil de criação de base de dados.

Foi obtido como conclusões perante os resultados e hipóteses formuladas, que no geral, a execução em tempo-real dos modelos aplicados a dispositivos de computação de borda com baixo poder computacional ainda é um desafio, visto que os modelos mais complexos e robustos em detecção tiveram dificuldades de serem executados, obtendo baixas velocidades de inferência e alto consumo de memória, inviabilizando suas implementações. Já os modelos em suas versões menos complexas e não tão acuradas, permitiram a execução em tempo real no Raspberry, mas ainda com baixas taxas de quadros-por-segundo. Portanto, ainda há necessidade de melhoramento das arquiteturas para um processamento mais eficiente sobre os recursos disponíveis nos dispositivos. Em incremento, também há a comprovação referente a viabilidade da metodologia para a construção de conjuntos de imagens de forma rápida e gratuita através de mecanismos de busca de imagens. O BRA-Dataset trouxe a possibilidade de treinamento das arquiteturas YOLO. Porém, conclui-se que apesar da eficácia e velocidade do método para a criação, não há garantias da qualidade das imagens, no qual ainda necessitam de um olhar humano especializado para remoção de imagens não interessantes. Além disso, o critério de limpeza do *dataset* relacionado a manter somente imagens de animais realistas, excluiu outras variáveis que influenciam no treinamento dos modelos e na qualidade do mesmo, como ruídos, imagens com baixa resolução e com animais em poses e ângulos desfavoráveis e que não reflitam a realidade da aplicação alvo (tendo como base o cenário estipulado de implantação de um possível sistema).

No aspecto de desempenho em detecções sobre as problemáticas de detecção dos cenários propostos, nenhum modelo utilizado pôde superar, demonstrando que ainda tais problemas de oclusão, reflexão, distâncias longínquas e camuflagem do animal afetam negativamente, com falsos negativos. Em situações convencionais, os testes do conjunto de validação demonstraram que o modelo Scaled-YoloV4 obteve os melhores resultados na mitigação de falsos negativos (melhor recordação), mostrando maior eficiência na detecção de animais brasileiros ameaçados de extinção. O modelo Scaled-YoloV4 também teve a maior precisão, enquanto o modelo YoloV4 teve o maior mAP@50. Em termos de FPS médio na inferência de vídeo, o modelo YoloV5-N foi o de melhor desempenho, devido sua largura de rede ser menor. Pode-se concluir que as arquiteturas YoloV4, Scaled-YoloV4, YoloV5 e YoloR fornecem desempenhos relevantes para a criação de sistemas de detecção de animais em tempo-real e que as técnicas de aumento de dados são eficazes e eficientes para treinar essas arquiteturas, mesmo com um conjunto de dados de domínio limitado. Em comparação, os detectores YoloV7 e YoloX tiveram resultados abaixo do esperado no conjunto de validação, devido aos seus filtros de convolução especializados para resoluções de imagem mais altas, sendo menos eficientes em entradas menores que HD. Em termos de FPS, a arquitetura desacoplada dos modelos YoloX reduziu drasticamente o desempenho, enquanto o modelo YoloV7 manteve alta velocidade mesmo com versões grandes e complexas.

Conclui-se que o uso de arquiteturas YOLO em dispositivos de borda com pouca memória RAM ainda é um desafio para redes grandes e complexas, mesmo em versões recentes. Ao contrário do desempenho dos modelos na GPU, a velocidade de execução da inferência dos modelos ainda é muito baixa, inviabilizando a implantação em ambientes computacionais com menos recursos ou que o poder computacional para calculo matricial seja baixo.

Para trabalhos futuros, é possível reavaliar o BRA-Dataset e considerar possíveis acréscimos de novas classes seguindo estatísticas de acidentes informadas pelos veículos de divulgação, como o ICMBio e CBEE. Além disso, também aumentar o acervo para as classes existentes, e realizar a filtragem de imagens por outros critérios, como qualidade de imagem e pose de animais por exemplo. Também exploraremos novas técnicas emergentes de aumento de dados para comparações futuras, como técnicas para simular cenários desfavoráveis para detecção, a fim de fornecer uma amostra maior de imagens para treinamento. Outras contribuições que podem ser aderidas, é a inclusão da implementação de outras arquiteturas de detecção de estágio único para comparação com aquelas baseadas no YOLO, o que pode expandir nossa compreensão dos desafios da detecção de animais. Em adição, avaliar os modelos e arquiteturas em dispositivos de inteligência artificial especializados para computação de ponta (por exemplo, Nvidia Jetson Family, dispositivos FPGA e outros) pode fornecer informações valiosas sobre a aplicação prática da detecção remota em tempo real e ajudar a avaliar o consumo de memória e processamento, podendo fornecer uma análise custo-benefício para implantações em cenários reais. Adicionalmente, avaliar os modelos em outros cenários de oclusão pode ajudar a enfrentar os desafios de detecção de animais em diferentes ambientes ao redor do mundo e fornecer a

outros pesquisadores uma melhor compreensão das tecnologias disponíveis em seu ambiente local.

Especificamente para o cenário brasileiro de pesquisas em detecção automática de animais, observa-se com a execução deste trabalho, lacunas referentes a vários aspectos do cenário ecológico no Brasil. Em primeiro lugar, o campo de pesquisa para o tema de acidentes envolvendo animais utilizando artifícios da ciência da computação para preposições de soluções computacionais é pouco fomentado, visto que apesar do problema ser recorrente e com amplo conhecimento da sociedade e governos, existe um baixo número de artigos científicos tratando do cenário interno brasileiro. Em contra-partida, estudos referentes a ecologia em estradas é altamente fomentado e que permite o embasamento para a criação de pesquisas, portanto, os motivos para o baixo referencial teórico de sistemas de computação inteligente para o auxílio de monitoramento das estradas se devem a fatores financeiros de implementação. Em segundo lugar, o campo de detecção de objetos e processamento de imagens já possui mecanismos modernos para fornecer subsídios tecnológicos para diversas problemáticas e cenários, sendo assim, visualizando tais técnicas, acredita-se que o emprego da visão computacional é o próximo passo para novos sistemas de computação urbana, em especial, o monitoramento de rodovias e cidades inteligentes, auxiliando não só em tarefas de segurança de condutores e animais, mas também no melhor entendimento de fenômenos ecológicos de fauna em setores urbanos e florestais, além de ampliar a intersecção de estudos das áreas biológicas e de computação.

7.1 Contribuições intelectuais

Como contribuições intelectuais, houve a publicação de artigos científicos em conferências da área de análise de dados de Big Data, processamento de imagem e reconhecimento de padrões.

A primeira contribuição é a publicação de um artigo que teve como objetivo apresentar lacunas da área de detecção animal para ambientes de computação urbana, bem como apresentar o estado-da-arte dos mecanismos utilizados para a detecção animal em diversos problemas ambientais e de conflito humano-animal. Além disso, o trabalho permitiu o embasamento teórico e técnico para o desenvolvimento da proposta desta dissertação.

- *Understanding the state of the Art in Animal detection and classification using computer vision technologies* na conferência *IEEE International Conference on Big Data (Big Data)* (FERRANTE *et al.*, 2021).

A segunda contribuição é referente a publicação de um artigo em conferência para a divulgação do BRA-Dataset, fornecendo informações detalhadas de sua construção e características, instâncias e métodos de rotulação, além de testes de desempenho do conjunto em algumas das arquiteturas YOLO trabalhadas durante o projeto.

- *Brazilian Road's Animals (BRA): An Image Dataset of Most Commonly Run Over Animals* na conferência *35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (FERRANTE *et al.*, 2022).

E por fim, a terceira contribuição consiste na publicação de um artigo de avaliação das arquiteturas apresentadas neste trabalho perante ao BRA-Dataset, visando divulgar os resultados e conclusões desta pesquisa e ampliar o conhecimento sobre o cenário de detecção animal e dificuldades relacionadas aos modelos utilizados.

- *Evaluating YOLO architectures for detecting road killed endangered Brazilian animals* no periódico *Nature Scientific Reports* (FERRANTE *et al.*, 2024).

REFERÊNCIAS

ACHANTA, R.; SHAJI, A.; SMITH, K.; LUCCHI, A.; FUA, P.; SüSSTRUNK, S. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 34, n. 11, p. 2274–2282, nov. 2012. ISSN 0162-8828, 2160-9292. Disponível em: <http://ieeexplore.ieee.org/document/6205760/>. Citado na página 51.

ADAMI, D.; OJO, M. O.; GIORDANO, S. Design, Development and Evaluation of an Intelligent Animal Repelling System for Crop Protection Based on Embedded Edge-AI. **IEEE Access**, v. 9, p. 132125–132139, 2021. ISSN 2169-3536. Citado nas páginas 54, 55, 56, 57 e 59.

AKABANE, A.; IMMICH, R.; PAZZI, R.; MADEIRA, E.; VILLAS, L. Distributed Egocentric Betweenness Measure as a Vehicle Selection Mechanism in VANETs: A Performance Evaluation Study. **Sensors**, v. 18, n. 8, p. 2731, ago. 2018. ISSN 1424-8220. Disponível em: <http://www.mdpi.com/1424-8220/18/8/2731>. Citado na página 25.

AKABANE, A. T.; IMMICH, R.; BITTENCOURT, L. F.; MADEIRA, E. R.; VILLAS, L. A. Towards a distributed and infrastructure-less vehicular traffic management system. **Computer Communications**, v. 151, p. 306–319, fev. 2020. ISSN 01403664. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S014036641930218X>. Citado na página 25.

AKABANE, A. T.; IMMICH, R.; MADEIRA, E. R. M.; VILLAS, L. A. iMOB: An Intelligent Urban Mobility Management System Based on Vehicular Social Networks. In: **2018 IEEE Vehicular Networking Conference (VNC)**. Taipei, Taiwan: IEEE, 2018. p. 1–8. ISBN 9781538694282. Disponível em: <https://ieeexplore.ieee.org/document/8628436/>. Citado na página 25.

AL-OMAIR, O. M.; HUANG, S. A comparative study on detection accuracy of cloud-based emotion recognition services. In: SPML '18: PROCEEDINGS OF THE 2018 INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING AND MACHINE LEARNING, 18., 2018, Xangai, China. Nova Iorque, Estados Unidos: ACM, 2018. p. 142–148. Citado na página 37.

ALAM, M.; FERREIRA, J.; FONSECA, J. (Ed.). **Intelligent Transportation Systems**. Cham: Springer International Publishing, 2016. v. 52. (Studies in Systems, Decision and Control, v. 52). ISBN 9783319281810 9783319281834. Disponível em: <http://link.springer.com/10.1007/978-3-319-28183-4>. Citado na página 25.

AN, H.; JUNG, J.-i. Design of a Cooperative Lane Change Protocol for a Connected and Automated Vehicle Based on an Estimation of the Communication Delay. **Sensors**, v. 18, n. 10, p. 3499, out. 2018. ISSN 1424-8220. Disponível em: <http://www.mdpi.com/1424-8220/18/10/3499>. Citado na página 25.

ANDRADE, R. d. O. **As máquinas que tudo veem**. FAPESP, 2019. Disponível em: <https://revistapesquisa.fapesp.br/as-maquinas-que-tudo-veem/>. Citado na página 37.

ANTôNIO, W. H. S.; SILVA, M. D.; MIANI, R. S.; SOUZA, J. R. A Proposal of an Animal Detection System Using Machine Learning. **Applied Artificial Intelligence**, v. 33, n. 13, p. 1093–1106, nov. 2019. ISSN 0883-9514, 1087-6545. Disponível em: <https://www.tandfonline.com/doi/full/10.1080/08839514.2019.1673993>. Citado nas páginas 52, 53, 57 e 59.

ARRUDA, M. d. S. de; SPADON, G.; RODRIGUES, J. F.; GONçALVES, W. N.; MACHADO, B. B. Recognition of Endangered Pantanal Animal Species using Deep Learning Methods. In: **2018 International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2018. p. 1–8. ISSN: 2161-4407. Citado nas páginas 50, 52, 57 e 59.

BAI, K.; LIU, S.; YI, Y. High speed and energy efficient deep neural network for edge computing. In: **Proceedings of the 4th ACM/IEEE Symposium on Edge Computing**. New York, NY, USA: Association for Computing Machinery, 2019. (SEC '19), p. 347–349. ISBN 9781450367332. Disponível em: <https://doi.org/10.1145/3318216.3363453>. Citado na página 45.

BISWAS, A. A.; RAHMAN, M. M.; RAJBONGSHI, A.; MAJUMDER, A. Recognition of Local Birds using Different CNN Architectures with Transfer Learning. In: **2021 International Conference on Computer Communication and Informatics (ICCCI)**. [S.l.: s.n.], 2021. p. 1–6. ISSN: 2329-7190. Citado nas páginas 54, 55, 56 e 59.

BLAIECH, A. G.; KHALIFA, K. B.; VALDERRAMA, C.; FERNANDES, M. A.; BEDOUI, M. H. A Survey and Taxonomy of FPGA-based Deep Learning Accelerators. **Journal of Systems Architecture**, v. 98, p. 331–345, set. 2019. ISSN 13837621. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1383762118304156>. Citado na página 45.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. **arXiv:2004.10934 [cs, eess]**, abr. 2020. ArXiv: 2004.10934. Disponível em: <http://arxiv.org/abs/2004.10934>. Citado nas páginas 39 e 40.

Bogusław Cyganek. **Object Detection and Recognition in Digital Images: Theory and Practice**. Oxford, UK: John Wiley & Sons Ltd, 2013. ISBN 9781118618387 9780470976371. Disponível em: <http://doi.wiley.com/10.1002/9781118618387>. Citado na página 37.

CARION, N.; MASSA, F.; SYNNAEVE, G.; USUNIER, N.; KIRILLOV, A.; ZAGORUYKO, S. End-to-end object detection with transformers. **CoRR**, abs/2005.12872, 2020. Disponível em: <https://arxiv.org/abs/2005.12872>. Citado na página 38.

CBEE. **Dados de atropelamento no Brasil**. 2023. Disponível em: <https://sistemaurubu.com.br/dados/>. Citado nas páginas 27 e 28.

CHEN, C.; SEFF, A.; KORNHAUSER, A.; XIAO, J. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. In: **2015 IEEE International Conference on Computer Vision (ICCV)**. Santiago, Chile: IEEE, 2015. p. 2722–2730. ISBN 9781467383912. Disponível em: <http://ieeexplore.ieee.org/document/7410669/>. Citado na página 25.

CLAPPIS, A. M. **Uma introdução as redes neurais convolucionais utilizando o Keras**. 2019. Disponível em: <https://medium.com/data-hackers/uma-introdu%C3%A7%C3%A3o-as-redes-neurais-convolucionais-utilizando-o-keras-41ee8dcc033e>. Citado na página 36.

CNT. **Painel CNT de Acidentes Rodoviários**. 2021. Disponível em: <https://cnt.org.br/painel-acidente>. Citado na página 26.

CYMBALUK, F. **Animais na pista**. 2018. Disponível em: <https://www.uol/noticias/especiais/animais-na-pista.htm>. Citado nas páginas 26 e 63.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. ImageNet: A large-scale hierarchical image database. In: **2009 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2009. p. 248–255. ISSN: 1063-6919. Citado na página 50.

ELIAS, A. R.; GOLUBOVIC, N.; KRINTZ, C.; WOLSKI, R. Where's the Bear? - Automating Wildlife Image Processing Using IoT and Edge Cloud Systems. In: **2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)**. [S.l.: s.n.], 2017. p. 247–258. Citado nas páginas 49, 50, 55 e 59.

Erika. **Infográficos**. 2021. Disponível em: <https://sistemaurubu.com.br/infograficos/>. Citado na página 63.

EXPERT, I. **Detecção de Objetos com YOLO, Darknet, OpenCV e Python**. 2021. Disponível em: <https://iaexpert.academy/courses/deteccao-de-objetos-com-yolo-darknet-opencv-e-python/>. Citado nas páginas 35 e 39.

FACELI, K. **Inteligência artificial: uma abordagem de aprendizado de máquina**. Rio de Janeiro: Grupo Gen - LTC, 2011. OCLC: 854567299. ISBN 9788521618805. Disponível em: <http://site.ebrary.com/id/10707120>. Citado na página 32.

FANG, Y.; LIAO, B.; WANG, X.; FANG, J.; QI, J.; WU, R.; NIU, J.; LIU, W. You only look at one sequence: Rethinking transformer in vision through object detection. **CoRR**, abs/2106.00666, 2021. Disponível em: <https://arxiv.org/abs/2106.00666>. Citado na página 43.

FERRANTE, G. S.; NAKAMURA, L. H. V.; ANDRADE, F. R. H.; FILHO, G. P. R.; GRANDE, R. E. D.; MENEGUETTE, R. I. Brazilian Road's Animals (BRA): An Image Dataset of Most Commonly Run Over Animals. In: **2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. Natal, Brazil: IEEE, 2022. p. 246–251. ISBN 9781665453851. Disponível em: <https://ieeexplore.ieee.org/document/9991774/>. Citado nas páginas 63, 67, 68 e 100.

FERRANTE, G. S.; NAKAMURA, L. H. V.; SAMPAIO, S.; FILHO, G. P. R.; MENEGUETTE, R. I. Evaluating YOLO architectures for detecting road killed endangered Brazilian animals. **Scientific Reports**, v. 14, n. 1, p. 1353, jan. 2024. ISSN 2045-2322. Disponível em: <https://www.nature.com/articles/s41598-024-52054-y>. Citado na página 100.

FERRANTE, G. S.; RODRIGUES, F. M.; ANDRADE, F. R. H.; GOULARTE, R.; MENEGUETTE, R. I. Understanding the state of the Art in Animal detection and classification using computer vision technologies. In: **2021 IEEE International Conference on Big Data (Big Data)**. [S.l.: s.n.], 2021. p. 3056–3065. Citado nas páginas 28, 38 e 99.

FILHO, O. M.; NETO, H. V. (Ed.). **Processamento Digital De Imagens**. Rio de Janeiro, Brasil: Brasport, 1999. 410 p. ISBN 978-8574520094. Citado na página 37.

FORSYTH, D.; PONCE, J. **Computer vision: a modern approach**. Upper Saddle River, N.J. ; London: Prentice Hall, 2003. OCLC: ocm50100728. ISBN 9780130851987. Citado na página 37.

GANESH, P. **Types of Convolution Kernels : Simplified**. 2019. Disponível em: <https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37>. Citado na página 35.

GE, Z.; LIU, S.; LI, Z.; YOSHIE, O.; SUN, J. OTA: optimal transport assignment for object detection. **CoRR**, abs/2103.14259, 2021. Disponível em: <https://arxiv.org/abs/2103.14259>. Citado na página 42.

GE, Z.; LIU, S.; WANG, F.; LI, Z.; SUN, J. YOLOX: exceeding YOLO series in 2021. **CoRR**, abs/2107.08430, 2021. Disponível em: <https://arxiv.org/abs/2107.08430>. Citado nas páginas 41, 42 e 43.

GEEKSFORGEEKS. **CNN | Introduction to Pooling Layer**. 2019. Disponível em: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>. Citado na página 36.

GERAT, J.; SOPIAK, D.; ORAVEC, M.; PAVLOVICOVA, J. Vehicle speed detection from camera stream using image processing methods. In: **2017 International Symposium ELMAR**. Zadar: IEEE, 2017. p. 201–204. ISBN 9789531842259. Disponível em: <http://ieeexplore.ieee.org/document/8124468/>. Citado na página 25.

GERON, A. **Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems**. First edition. Beijing ; Boston: O'Reilly Media, 2017. OCLC: ocn953432302. ISBN 9781491962299. Citado na página 32.

GIRSHICK, R. Fast R-CNN. **arXiv:1504.08083 [cs]**, set. 2015. ArXiv: 1504.08083. Disponível em: <http://arxiv.org/abs/1504.08083>. Citado na página 38.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. **arXiv:1311.2524 [cs]**, out. 2014. ArXiv: 1311.2524. Disponível em: <http://arxiv.org/abs/1311.2524>. Citado na página 38.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. The MIT Press, 2016. OCLC: 1183962587. ISBN 9780262337373. Disponível em: <http://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=6287197>. Citado nas páginas 32 e 33.

HAO, J.; SUBEDI, P.; KIM, I. K.; RAMASWAMY, L. Characterizing Resource Heterogeneity in Edge Devices for Deep Learning Inferences. In: **Proceedings of the 2021 on Systems and Network Telemetry and Analytics**. New York, NY, USA: Association for Computing Machinery, 2020. (SNTA '21), p. 21–24. ISBN 9781450383868. Disponível em: <https://doi.org/10.1145/3452411.3464446>. Citado na página 45.

HERNANDEZ-JAYO, U.; IGLESIA, I. De-la; PEREZ, J. V-Alert: Description and Validation of a Vulnerable Road User Alert System in the Framework of a Smart City. **Sensors**, v. 15, n. 8, p. 18480–18505, jul. 2015. ISSN 1424-8220. Disponível em: <http://www.mdpi.com/1424-8220/15/8/18480>. Citado na página 25.

HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M.; ADAM, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. **arXiv:1704.04861 [cs]**, abr. 2017. ArXiv: 1704.04861. Disponível em: <http://arxiv.org/abs/1704.04861>. Citado na página 50.

HUSSAIN, S. A.; JAHROMI, B. S.; CETIN, S. Cooperative Highway Lane Merge of Connected Vehicles Using Nonlinear Model Predictive Optimal Controller. **Vehicles**, v. 2, n. 2, p. 249–266, mar. 2020. ISSN 2624-8921. Disponível em: <https://www.mdpi.com/2624-8921/2/2/14>. Citado na página 25.

ICMBIO. **Instituto Chico Mendes de Conservação da Biodiversidade - Espécies ameaçadas**. 2018. Disponível em: <https://www.icmbio.gov.br/portal/especies-ameacadas-destaque>. Citado na página 27.

IMMICH, R.; CERQUEIRA, E.; CURADO, M. Towards a QoE-driven mechanism for improved H.265 video delivery. In: **2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)**. Vilanova i la Geltru, Spain: IEEE, 2016. p. 1–8. ISBN 9781509019830. Disponível em: <http://ieeexplore.ieee.org/document/7528427/>. Citado na página 25.

____. Efficient high-resolution video delivery over VANETs. **Wireless Networks**, v. 25, n. 5, p. 2587–2602, jul. 2019. ISSN 1022-0038, 1572-8196. Disponível em: <http://link.springer.com/10.1007/s11276-018-1687-2>. Citado na página 25.

JOCHER, G.; CHAURASIA, A.; STOKEN, A.; BOROVEC, J.; NANOCODE012; KWON, T. Y.; MICHAEL, K.; FANG, J.; IMYHXY; LORNA; WONG, C.; YIFU), V, A.; MONTES, D.; WANG, Z.; FATI, C.; NADAR, J.; LAUGHING; UNGLVKITDE; TKIANAI; YXNONG; SKALSKI, P.; HOGAN, A.; STROBEL, M.; JAIN, M.; MAMMANA, L.; XYLIEONG. **ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations**. Zenodo, 2020. Disponível em: <https://zenodo.org/record/3908559>. Citado na página 41.

KHANAL, S. R.; BARROSO, J.; LOPES, N.; SAMPAIO, J.; FILIPE, V. Performance analysis of microsoft's and google's emotion recognition api using pose-invariant faces. In: DSAI 2018: 8TH INTERNATIONAL CONFERENCE ON SOFTWARE DEVELOPMENT AND TECHNOLOGIES FOR ENHANCING ACCESSIBILITY AND FIGHTING INFO-EXCLUSION, 8., 2018, Tessalônica, Grécia. Nova Iorque, Estados Unidos: ACM, 2018. p. 172–178. Citado na página 37.

KIM, M.; KIM, H. K.; LEE, S. H. A Distributed Cooperative Localization Strategy in Vehicular-to-Vehicular Networks. **Sensors**, v. 20, n. 5, p. 1413, mar. 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/5/1413>. Citado na página 25.

KLJUCARIC, L.; JOHNSON, A.; GEORGE, A. D. Architectural Analysis of Deep Learning on Edge Accelerators. In: **2020 IEEE High Performance Extreme Computing Conference (HPEC)**. [S.l.: s.n.], 2020. p. 1–7. ISSN: 2643-1971. Citado na página 45.

LEMLEY, J.; BAZRAFKAN, S.; CORCORAN, P. Deep Learning for Consumer Devices and Services: Pushing the limits for machine learning, artificial intelligence, and computer vision. **IEEE Consumer Electronics Magazine**, v. 6, n. 2, p. 48–56, abr. 2017. ISSN 2162-2256. Citado na página 37.

LI, S.; XU, L. D.; ZHAO, S. The internet of things: a survey. **Information Systems Frontiers**, v. 17, n. 2, p. 243–259, abr. 2015. ISSN 1572-9419. Disponível em: <https://doi.org/10.1007/s10796-014-9492-7>. Citado na página 45.

LI, X.; SHI, Y. Computer Vision Imaging Based on Artificial Intelligence. In: **2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)**. [S.l.: s.n.], 2018. p. 22–25. Citado na página 37.

LIN, T.; MAIRE, M.; BELONGIE, S. J.; BOURDEV, L. D.; GIRSHICK, R. B.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft COCO: common objects in context. **CoRR**, abs/1405.0312, 2014. Disponível em: <http://arxiv.org/abs/1405.0312>. Citado na página 74.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S. E.; FU, C.; BERG, A. C. SSD: single shot multibox detector. **CoRR**, abs/1512.02325, 2015. Disponível em: <http://arxiv.org/abs/1512.02325>. Citado na página 38.

MAHARANA, K.; MONDAL, S.; NEMADE, B. A review: Data pre-processing and data augmentation techniques. **Global Transitions Proceedings**, v. 3, n. 1, p. 91–99, jun. 2022. ISSN 2666285X. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S2666285X22000565>. Citado na página 69.

MARQUES, D. **Viadutos para animais silvestres começam a ser implantados no Brasil**. 2020. Disponível em: <https://brasil.mongabay.com/2020/10/viadutos-para-animais-silvestres-comecam-a-ser-implantados-no-brasil/>. Citado na página 26.

MENEGUETTE ROBSON A F LOUREIRO, A. R. E. D. G. I. **INTELLIGENT TRANSPORT SYSTEM IN SMART CITIES: aspects and challenges of vehicular networks... and cloud**. S.l.: SPRINGER, 2019. OCLC: 1086339834. ISBN 9783030066413. Citado na página 25.

MINICHINO, J.; HOWSE, J. **Learning OpenCV 3 computer vision with Python: unleash the power of computer vision with Python using OpenCV**. Second edition. Birmingham Mumbai: Packt Publishing, 2015. (Community experience distilled). ISBN 9781785283840. Citado nas páginas 33 e 34.

MITCHELL, T. M. **Machine Learning**. 1. ed. New York: McGraw-Hill, 1997. (McGraw-Hill series in computer science). ISBN 9780070428072. Citado na página 31.

NACIONAL, J. **Iniciativas em Minas Gerais ajudam na prevenção de atropelamento de animais nas estradas**. 2023. Publisher: G1. Disponível em: <https://g1.globo.com/jornal-nacional/noticia/2023/03/13/iniciativas-em-minas-gerais-ajudam-na-prevencao-de-atropelamento-de-animais-nas-estradas.ghtml>. Citado na página 26.

NOGUEIRA, J. G. **Intel lança o Neural Compute Stick 2, seu novo computador de bolso para IA**. 2018. Disponível em: <https://mundoconectado.com.br/noticias/v/7491/intel-lanca-o-neural-compute-stick-2-seu-novo-computador-de-bolso-para-ia>. Citado na página 46.

NVIDIA. **NVIDIA Jetson Nano para Aplicações AI e de Educação no Edge**. 2023. Disponível em: <https://www.nvidia.com/pt-br/autonomous-machines/embedded-systems/jetson-nano/>. Citado na página 47.

OSCO, L. P.; WU, Q.; LEMOS, E. L. de; GONçALVES, W. N.; RAMOS, A. P. M.; LI, J.; JUNIOR, J. M. **The Segment Anything Model (SAM) for Remote Sensing Applications: From Zero to One Shot**. 2023. Citado na página 38.

PADILLA, D. A.; VILLAVERDE, J. F.; MAGDARAOG, J. J. T.; OCONER, A. J. L.; RANJO, J. P. Vehicle and Weather Detection Using Real Time Image Processing Using Optical Flow and Color Histogram. In: **2019 5th International Conference on Control, Automation and**

Robotics (ICCAR). Beijing, China: IEEE, 2019. p. 880–883. ISBN 9781728133263. Disponível em: <https://ieeexplore.ieee.org/document/8813346/>. Citado nas páginas 25 e 36.

PANG, H.; TAN, K.-L. Authenticating query results in edge computing. In: **Proceedings. 20th International Conference on Data Engineering**. [S.l.: s.n.], 2004. p. 560–571. ISSN: 1063-6382. Citado na página 44.

PETSO, T.; JAMISOLA, R. S.; MPOELENG, D.; MMEREKI, W. Individual Animal and Herd Identification Using Custom YOLO v3 and v4 with Images Taken from a UAV Camera at Different Altitudes. In: **2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP)**. [S.l.: s.n.], 2021. p. 33–39. Citado nas páginas 55, 56, 57 e 59.

PUJARI, M. S. M. R. K. P. **Practical Convolutional Neural Networks**. Place of publication not identified: Packt Publishing, 2018. OCLC: 1035779157. ISBN 9781788392303. Disponível em: <https://proquest.safaribooksonline.com/9781788392303>. Citado na página 35.

QUADROS, C.; CERQUEIRA, E.; NETO, A.; RIKER, A.; IMMICH, R.; CURADO, M. A mobile QoE Architecture for Heterogeneous Multimedia Wireless Networks. In: **2012 IEEE Globecom Workshops**. [S.l.: s.n.], 2012. p. 1057–1061. ISSN: 2166-0077. Citado na página 25.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You Only Look Once: Unified, Real-Time Object Detection. **arXiv:1506.02640 [cs]**, maio 2016. ArXiv: 1506.02640. Disponível em: <http://arxiv.org/abs/1506.02640>. Citado nas páginas 38 e 40.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. **arXiv:1506.01497 [cs]**, jan. 2016. ArXiv: 1506.01497. Disponível em: <http://arxiv.org/abs/1506.01497>. Citado na página 38.

RICH, E.; KNIGHT, K. **Inteligencia artificial**. São Paulo (SP): Makron Books, 1994. OCLC: 816721738. ISBN 9788534601221. Citado na página 31.

RODRIGUES, V. **Métricas de Avaliação: acurácia, precisão recall... quais as diferenças?** 2021. Disponível em: <https://vitorborbarodrigues.medium.com/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-acur%C3%A1cia-precis%C3%A3o-recall-quais-as-diferen%C3%A7as-c8f05e0a513c>. Citado na página 78.

ROSA, J. **Fundamentos da Inteligência Artificial**. Rio de Janeiro: Editora LTC, 2011. 228 p. ISBN 9788521605935. Citado na página 31.

ROSEBROCK, A. **Intersection over Union (IoU) for object detection**. 2016. Disponível em: <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>. Citado na página 78.

____. **Deep learning for computer vision with Python: practitioner bundle**. 1. ed. [S.l.: s.n.], 2017. OCLC: 1057451360. ISBN 9781722487836. Citado na página 38.

SANTOS, T. T.; BARBEDO, J. G. A.; TERNES, S.; NETO, J. C.; KOENIGKAN, L. V.; SOUZA, K. X. S. D. Visão computacional aplicada na agricultura. In: **Agricultura digital: pesquisa, desenvolvimento e inovação nas cadeias produtivas.** Brasília, DF: Embrapa, 2020. p. 406. ISBN 978-65-86056-37-2. Original-date: 2020. Citado na página 46.

SATYANARAYANAN, M. The Emergence of Edge Computing. **Computer**, v. 50, n. 1, p. 30–39, jan. 2017. ISSN 1558-0814. Citado na página 44.

SCHNEIDER, S.; TAYLOR, G. W.; KREMER, S. Deep Learning Object Detection Methods for Ecological Camera Trap Data. In: **2018 15th Conference on Computer and Robot Vision (CRV)**. [S.l.: s.n.], 2018. p. 321–328. Citado nas páginas 51, 52, 57, 58 e 59.

SIMCHON, L.; RABINOVICI, R. Real-Time Implementation of Green Light Optimal Speed Advisory for Electric Vehicles. **Vehicles**, v. 2, n. 1, p. 35–54, jan. 2020. ISSN 2624-8921. Disponível em: <https://www.mdpi.com/2624-8921/2/1/3>. Citado na página 25.

SINGH, H. **Practical Machine Learning and Image Processing: For Facial Recognition, Object Detection, and Pattern Recognition Using Python**. Berkeley, CA: Apress, 2019. ISBN 9781484241486 9781484241493. Disponível em: <http://link.springer.com/10.1007/978-1-4842-4149-3>. Citado na página 34.

SONG, Y.; LIN, Z. Species recognition technology based on migration learning and data augmentation. In: **2018 5th International Conference on Systems and Informatics (ICSAI)**. Nanjing: IEEE, 2018. p. 1016–1021. ISBN 9781728101200. Disponível em: <https://ieeexplore.ieee.org/document/8599361/>. Citado nas páginas 50, 51, 56 e 59.

SUBIRá, R. J.; GALVãO, A.; CARVALHO, C. E. G. de; SOARES, A. H. S. d. B. **Livro Vermelho da Fauna Brasileira Ameaçada de Extinção volume II – mamíferos**. 1. ed. Brasília, DF: ICMBio/MMA, 2018. v. 2. Disponível em: <https://www.gov.br/icmbio/pt-br/centrais-de-conteudo/publicacoes/publicacoes-diversas/livro-vermelho/livro-vermelho-da-fauna-brasileira-ameacada-de-extincao-2018>. Citado nas páginas 63, 64 e 65.

SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. Rethinking the Inception Architecture for Computer Vision. **arXiv:1512.00567 [cs]**, dez. 2015. ArXiv: 1512.00567. Disponível em: <http://arxiv.org/abs/1512.00567>. Citado na página 49.

Introduction to edge computing. In: TAHERI, J.; Shuiguang Deng (Ed.). **Edge Computing: Models, technologies and applications**. Institution of Engineering and Technology, 2020. p. 3–25. ISBN 9781785619403 9781785619410. Disponível em: <https://digital-library.theiet.org/content/books/10.1049/pbpc033e_ch1>. Citado nas páginas 44 e 45.

TAN, M.; PANG, R.; LE, Q. V. EfficientDet: Scalable and Efficient Object Detection. **arXiv:1911.09070 [cs, eess]**, jul. 2020. ArXiv: 1911.09070. Disponível em: <http://arxiv.org/abs/1911.09070>. Citado na página 40.

TAN, R. J. **Breaking down Mean Average Precision (mAP)**. 2021. Disponível em: <https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52>. Citado nas páginas 79 e 80.

VIANA, I. B.; AOUF, N. Distributed Cooperative Path-Planning for Autonomous Vehicles Integrating Human Driver Trajectories. In: **2018 International Conference on Intelligent Systems (IS)**. Funchal - Madeira, Portugal: IEEE, 2018. p. 655–661. ISBN 9781538670972. Disponível em: <https://ieeexplore.ieee.org/document/8710544/>. Citado na página 25.

VOURGIDIS, I.; MAGLARAS, L.; ALFAKEEH, A. S.; AL-BAYATTI, A. H.; FERRAG, M. A. Use Of Smartphones for Ensuring Vulnerable Road User Safety through Path Prediction and Early Warning: An In-Depth Review of Capabilities, Limitations and Their Applications in Cooperative Intelligent Transport Systems. **Sensors**, v. 20, n. 4, p. 997, fev. 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/4/997>. Citado na página 25.

WANG, C.; LIAO, H. M.; YEH, I.; WU, Y.; CHEN, P.; HSIEH, J. Cspnet: A new backbone that can enhance learning capability of CNN. **CoRR**, abs/1911.11929, 2019. Disponível em: <http://arxiv.org/abs/1911.11929>. Citado na página 41.

WANG, C.; WANG, Y.; HAN, Y.; SONG, L.; QUAN, Z.; LI, J.; LI, X. CNN-based object detection solutions for embedded heterogeneous multicore SoCs. In: **2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)**. [S.l.: s.n.], 2017. p. 105–110. ISSN: 2153-697X. Citado na página 45.

WANG, C.; YEH, I.; LIAO, H. M. You only learn one representation: Unified network for multiple tasks. **CoRR**, abs/2105.04206, 2021. Disponível em: <https://arxiv.org/abs/2105.04206>. Citado nas páginas 41 e 42.

WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. Scaled-yolov4: Scaling cross stage partial network. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2021. p. 13029–13038. Citado na página 41.

____. **YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors**. arXiv, 2022. Disponível em: <https://arxiv.org/abs/2207.02696>. Citado nas páginas 43 e 44.

WANG, P.; GAO, S.; LI, L.; SUN, B.; CHENG, S. Obstacle Avoidance Path Planning Design for Autonomous Driving Vehicles Based on an Improved Artificial Potential Field Algorithm. **Energies**, v. 12, n. 12, p. 2342, jun. 2019. ISSN 1996-1073. Disponível em: <https://www.mdpi.com/1996-1073/12/12/2342>. Citado na página 25.

WEI, L.; CAPPELLE, C.; RUICHEK, Y.; ZANN, F. Intelligent Vehicle Localization in Urban Environments Using EKF-based Visual Odometry and GPS Fusion. **IFAC Proceedings Volumes**, v. 44, n. 1, p. 13776–13781, jan. 2011. ISSN 14746670. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1474667016458386>. Citado na página 25.

ZHANG, H.; LI, F.; LIU, S.; ZHANG, L.; SU, H.; ZHU, J.; NI, L. M.; SHUM, H.-Y. **DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection**. 2022. Citado na página 38.

ZHANG, Z.; HE, T.; ZHANG, H.; ZHANG, Z.; XIE, J.; LI, M. Bag of freebies for training object detection neural networks. **CoRR**, abs/1902.04103, 2019. Disponível em: <http://arxiv.org/abs/1902.04103>. Citado na página 70.

ZHU, X.; SU, W.; LU, L.; LI, B.; WANG, X.; DAI, J. Deformable DETR: deformable transformers for end-to-end object detection. **CoRR**, abs/2010.04159, 2020. Disponível em: <https://arxiv.org/abs/2010.04159>. Citado na página 38.

ÖZKAYA, O.; YILLIKCI, G. (Ed.). **Arduino Computer Vision Programming**. Birmingham, Inglaterra: Packt Publishing, 2015. 222 p. ISBN 978-1-78355-262-7. Citado na página 36.


ICMC
USP
SÃO CARLOS