MONITORAMENTO DE ANIMAIS DOMÉSTICOS COM REDES NEURAIS CONVOLUCIONAIS E TRANSFER LEARNING: UMA ABORDAGEM UTILIZANDO YOLO PARA CUIDAR DOS MEUS PETS

Alex Almeida do Amaral*
Raimundo Valter Costa Filho**

RESUMO

Nos últimos anos a visão computacional obteve avanços significativos, ampliando seu conhecimento e aplicações em diversas áreas. Um exemplo importante é o uso dessa tecnologia para melhorar o reconhecimento de diferentes tipos de animais. Nesse artigo é proposto um sistema de vigilância inteligente que pode identificar individualmente cada animal em um local específico e indicar claramente áreas perigosas ou inadequadas durante o monitoramento, garantindo a segurança de pessoas e dos animais monitorados. Nesse contexto, algoritmos de aprendizado profundo, como *Convolutional Neural Networks* (CNN), são usados para produzir modelos de aprendizado de máquina que podem detectar e identificar objetos em imagens digitais. O estudo utiliza o modelo *You Only Look Once* (YOLO) versão 8 e alcança uma precisão de $99,5\%$ no reconhecimento de animais, indicando sua eficácia no monitoramento. Além disso, uma comparação entre um modelo treinado partindo da inicialização aleatória dos pesos e outro baseado em *transfer learning* revela que o último supera em diversas métricas, evidenciando uma precisão de $99,5\%$, *recall* de $99,3\%$, *Mean Average Precision* (mAP)50 de $99,5\%$, e mAP50 $-$ 95 de $77,5\%$. Esses resultados ressaltam a vantagem do *transfer learning* na otimização do desempenho.

Palavras-chave:Aprendizagem Profunda. Redes Neurais. Reconhecimento de Imagem.

ABSTRACT

In recent years, computer vision has made significant advancements, expanding its knowledge and applications in various fields. An important example is the use of this technology to enhance the recognition of different types of animals. This article proposes an intelligent surveillance system that can individually identify each animal in a specific location and clearly indicate dangerous or unsuitable areas during monitoring, ensuring the safety of both people and the

* Graduando em Ciência da Computação, Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Aracati, CE, Brasil. E-mail: alex.almeida.amaral05@aluno.ifce.edu.br.
** Doutor em Engenharia de Teleinformática, Docente do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Aracati, CE, Brasil. E-mail: valter.costa@ifce.edu.br

monitored animals. In this context, deep learning algorithms, such as CNN, are used to produce machine learning models that can detect and identify objects in digital images. The study utilizes the YOLO model version 8 and achieves an accuracy of 99.5% in animal recognition, indicating its effectiveness in monitoring. Furthermore, a comparison between a model trained from random weight initialization and another based on *transfer learning* reveals that the latter outperforms across various metrics, demonstrating an accuracy of 99.5%, a *recall* of 99.3%, mAP50 of 99.5%, and mAP50 – 95 of 77.5%. These results highlight the advantage of *transfer learning* in optimizing performance.

Keywords: *Deep Learning*. *Neural Networks*. *Image Recognition*.

1 INTRODUÇÃO

O Brasil figura como o segundo país com maior número de cães e gatos como animais de estimação, ficando atrás apenas dos Estados Unidos. Essa realidade abrange cerca de 60% dos lares brasileiros, nos quais esses animais de companhia se integram. Conviver com esses *pets* oferece uma série de benefícios para os seres humanos, tais como a redução dos níveis de ansiedade, estresse e o enfrentamento da depressão (DOMINGUES et al., 2015).

Em seu trabalho, Carvalho (2013) descreve que a interação atual entre seres humanos e animais de estimação é influenciada pela modernização das cidades e pela prevalência da individualização na cultura ocidental. Ele destaca, adicionalmente, a perspectiva de que esses animais possam desempenhar papéis mais proeminentes dentro do ambiente doméstico.

A fuga de um animal de estimação causa sofrimento aos seus tutores devido ao forte vínculo emocional que se estabelece entre esses animais e seus cuidadores. Mesmo com precauções cuidadosas, é comum que os animais se percam de casa. Uma pesquisa realizada nos Estados Unidos em 2010 constatou que 15% dos tutores de gatos e cachorros perderam seus animais nos cinco anos anteriores ao estudo (WEISS; SLATER; LORD, 2012).

A proposta deste trabalho é desenvolver um sistema de monitoramento para animais de estimação baseado em inteligência artificial, capaz de reconhecer cada *pet* individualmente e auxiliar seus tutores nas tarefas rotineiras de vigia. O sistema propõe monitorar áreas restritas (porta de saída de casa ou um lugar que oferece risco) afim de garantir a segurança do *pet*, registrando violações em *logs* (abreviação do inglês *logbooks*), mantendo um diário que nos ajude a desvendar o paradeiro do *pet*.

O desafio inerente a este projeto reside na habilidade do sistema em detectar e classificar diversos *pets* de forma eficiente em uma aplicação específica. A seleção do algoritmo de detecção e classificação considerou duas necessidades cruciais derivadas da natureza da aplicação: a velocidade na detecção e a capacidade de aprendizado com poucos exemplos. Em cenários de monitoramento de vídeo, é comum lidar com taxas de amostragem de 7,5 *frames per second* (fps) (SANTOS; JUNIOR, 2023). Além disso, é essencial que o algoritmo demonstre a capacidade de

aprender a reconhecer um *pet* específico a partir de um número reduzido de exemplos.

Diante dessas exigências, optou-se pelo uso do algoritmo YOLO, dada sua eficiência na detecção em tempo real, aliada à capacidade de aprendizado com poucos dados. A oportunidade de empregar o *transfer learning* com o YOLO acrescenta um aspecto valioso ao projeto. Essa abordagem permitirá que o algoritmo aproveite conhecimentos prévios adquiridos em um conjunto de dados maior, facilitando assim a tarefa de reconhecimento de *pets* específicos entre outros animais de interesse.

Quanto à estrutura do artigo, a seção 2 faz uma breve revisão acerca da teoria utilizada, apresentando informações sobre conceitos relacionados à visão computacional e como eles se conectam com o projeto desenvolvido. Na seção 3, são abordados trabalhos relacionados, analisando estudos feitos por outros autores sobre o mesmo tema. A seção 4 descreve a metodologia utilizada para a aplicação proposta. A seção 5, apresenta-se tecnologias utilizadas, assim como os equipamentos empregados na implementação do sistema. Na seção 6 discorre acerca dos resultados alcançados após o treinamento da rede neural convolucional. Por fim, na seção 7 são apresentadas as conclusões e trabalhos futuros.

2 REFERENCIAL TEÓRICO

Para melhor compreensão sobre o tema abordado, esta seção apresentados conceitos sobre Inteligência Artificial (IA), *Deep Learning*, Visão Computacional e *Convolutional Neural Networks* (CNN), ferramentas utilizadas no sistema.

2.1 Inteligência Artificial

De acordo com Bellman (1957), a IA é o campo da ciência da computação que se concentra no desenvolvimento de sistemas e algoritmos que permitem que os computadores realizem tarefas que, quando executadas por seres humanos, requerem inteligência, como o reconhecimento de padrões, a resolução de problemas complexos e a tomada de decisões informadas.

Assim podemos dizer que a IA permite que as máquinas executem operações que envolvam processos complexos, tais como a capacidade de reconhecer padrões, resolver problemas para os quais não existe uma solução imediatamente óbvia e tomar decisões informadas com base nos dados e informações disponíveis.

Outro conceito segundo RUSSELL e NORVIG (2009), A IA refere-se à simulação de processos de inteligência humana por meio de sistemas computacionais, como aprendizado de máquina, redes neurais artificiais e algoritmos de Processamento de Linguagem Natural (PLN), para realizar tarefas complexas e adaptar-se a novas situações.

Notamos que criar sistemas que imitam ou simulam a capacidade de raciocínio e aprendizado humano por meio de tecnologias computacionais é uma das abordagens fundamentais de uma IA, esta subárea é conhecida como aprendizado de máquina. Redes Neurais Artificiais (RNA) é o algoritmo mais utilizado que desempenha papel proeminente nesse campo.

2.2 Aprendizado de máquina

Podemos dizer que aprendizado de máquina refere-se à capacidade de os sistemas aprenderem com dados e experiências. Em vez de uma programação explícita para executar uma tarefa específica, as máquinas aprendem a executar tarefas através da análise de dados e do reconhecimento de padrões. O objetivo é permitir que sistemas automatizados melhorem seu desempenho ao longo do tempo com base na experiência adquirida com os dados. Há vários tipos de tarefas de aprendizado de máquina, cada um com características distintas para a aprendizagem e o seu processamento de dados. Os principais tipos incluem:

- **Aprendizagem Supervisionada:** é uma abordagem na qual o modelo é treinado usando um conjunto de dados rotulados, ou seja, dados em que as saídas desejadas já são conhecidas. Isso permite que o modelo aprenda a mapear as entradas para as saídas corretas, tornando-o útil em tarefas de previsão e classificação (WITTEN, 2005).

- **Aprendizagem Não Supervisionada:** é uma abordagem que visa descobrir padrões e estruturas subjacentes nos dados sem a necessidade de rótulos ou categorias predefinidas. Essa técnica é comumente usada em tarefas de clusterização, redução de dimensionalidade e geração de representações latentes (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

- **Aprendizagem Semi-Supervisionada:** a ideia é rotular alguns dos exemplos no conjunto de exemplos não rotulados, os quais são posteriormente utilizados durante a fase de treinamento do classificador, frequentemente resultando em uma classificação mais precisa (BLUM; MITCHELL, 1998). O modelo é treinado usando os dados rotulados, onde as entradas já estão associadas a resultados conhecidos, posteriormente utiliza-se o modelo pré-treinado para explorar e aprender padrões mais amplos nos modelos ainda não rotulados. O objetivo é superar a limitação do custo e da dificuldade associados à rotulação extensiva de dados.

- **Aprendizagem por Reforço:** A aprendizagem por reforço é um paradigma computacional de aprendizagem em que um agente aprendiz procura maximizar uma medida de desempenho baseada nos reforços (recompensas ou punições) que recebe ao interagir com um ambiente desconhecido (RIBEIRO, 1999). Este aprendizado é inspirado na psicologia do comportamento, onde um agente aprende a agir de maneira a obter a máxima recompensa possível, ajustando seu comportamento com base nas experiências passadas. O aprendizado por reforço tem aplicações em diversas áreas, incluindo robótica, jogos, finanças, controle de tráfego e até mesmo em sistemas de recomendação.

- ***transfer learning*:** De acordo com Raffel et al. (2023), é uma técnica em que um modelo é primeiro pré-treinado em uma tarefa rica em dados antes de ser ajustado em uma tarefa posterior. Esta abordagem emergiu como uma técnica poderosa em vários campos, incluindo PLN e visão computacional. O conhecimento previamente adquirido em domínios de origem distintos, porém relacionados, podem aprimorar o desempenho dos

aprendizes nos domínios-alvo. Essa abordagem visa diminuir a necessidade de uma grande quantidade de dados específicos do domínio-alvo para o treinamento dos aprendizes-alvo. Esta técnica é interessante para não depender apenas de dados locais, a transferência de conhecimento aproveita informações valiosas de contextos relevantes para otimizar o processo de aprendizagem no domínio desejado (SHIN et al., 2016).

2.3 *Deep Learning*

Deep Learning é uma subárea do aprendizado de máquina que se concentra em redes neurais profundas, que são modelos com muitas camadas ocultas. Essas redes têm a capacidade de aprender representações hierárquicas de dados, o que as torna ideais para tarefas complexas de aprendizado e reconhecimento (GOODFELLOW; BENGIO; COURVILLE, 2016).

Segundo Schmidhuber (2015), o aprendizado profundo envolve a construção de redes neurais artificiais com múltiplas camadas ocultas. Cada camada oculta é composta por unidades de processamento que realizam transformações lineares e não lineares nos dados de entrada. Durante o treinamento, as conexões entre as unidades são ajustadas iterativamente para minimizar uma função de custo, permitindo que a rede aprenda representações complexas dos dados.

A propagação direta dos dados transporta até a camada de saída, onde são geradas as previsões ou resultados. A seleção da função de ativação na camada de saída depende da natureza da tarefa. Uma função *loss* avalia a diferença entre as previsões e os rótulos reais, o treinamento dessa classe de algoritmos consiste em minimizar essa métrica quanto avalia-se as amostras separadas para o treinamento.

O algoritmo *backpropagation* é fundamental, pois realiza ajustes nos pesos da rede a partir das discrepâncias identificadas pela função de cálculo do *loss*. Esse procedimento iterativo ocorre ao longo de múltiplas épocas de treinamento. A eficiência das redes profundas frequentemente está atrelada ao uso de grandes conjuntos de dados, sendo que o processo de treinamento pode demandar bastante processamento computacional.

Algoritmos de *deep learning*, destaque para visão computacional e PLN, tem desempenhado papel importante revolucionando áreas como educação, diagnóstico médico, indústria automobilística (veículos autônomos), jogos, previsão de séries temporais, etc. A capacidade das redes neurais profundas de aprender representações complexas e extrair informações valiosas dos dados as torna uma ferramenta versátil para uma ampla gama de aplicações (LECUN, 2015).

2.3.1 Visão Computacional

De acordo com Ballard (1982), visão computacional é a ciência responsável pela visão de uma máquina, pela forma como um computador enxerga o meio à sua volta, extraindo informações significativas a partir de imagens capturadas por câmeras de vídeo, sensores, *scanners*, entre outros dispositivos. Estas informações permitem reconhecer, manipular e pensar sobre os objetos que compõem uma imagem.

Esse campo da computação, fornece ao computador uma variedade de informações através da análise de dados visuais e a tomada de decisões com base nessas informações. O processo faz o uso das seguintes etapas:

- **Aquisição de Dados:** Captura de imagens ou vídeos usando câmeras ou sensores;
- **Pré-processamento:** Melhoramento das imagens, incluindo a remoção de ruído, ajuste de contraste e normalização de cores;
- **Extração de Características:** Identificação de elementos importantes nas imagens, como bordas, texturas, formas e cores;
- **Reconhecimento de Padrões:** Utilização de algoritmos para identificar objetos, rostos, caracteres ou outros padrões nas imagens;
- **Interpretação e Tomada de Decisões:** Com base nas características extraídas e nos padrões reconhecidos, os computadores podem interpretar o conteúdo visual e tomar decisões, como classificar objetos, detectar movimentos ou realizar ações específicas.

Todas as etapas permitem que os computadores entendam e atuem com base nas informações visuais, com aplicações em uma ampla gama de campos incluindo medicina, indústria e segurança. Dentre os algoritmos mais proeminentes nesse campo, certamente devemos ressaltar o papel da CNN.

2.3.2 Rede Neurais Convolucionais CNN

De acordo com LeCun (2015), as CNNs são uma classe de modelos de aprendizado profundo projetados para processar dados de forma semelhante à maneira como o cérebro humano extrai características visuais. Elas são compostas por múltiplas camadas, incluindo camadas de convolução, que aplicam filtros para identificar padrões locais, camadas de *pooling*, que reduzem a dimensionalidade preservando informações essenciais, e camadas totalmente conectadas que realizam tarefas de classificação ou regressão. O funcionamento das CNNs baseia-se na aprendizagem de representações hierárquicas de características, tornando-as extremamente eficazes na visão computacional, processamento de imagens e em muitas outras aplicações.

As unidades de convolução nas CNNs realizam uma operação de convolução que consiste em multiplicar os pesos dos filtros com as regiões locais dos dados de entrada. Os resultados são então somados e passados por uma função de ativação não linear, como a função *Rectified Linear Unit* (ReLU), introduzindo não linearidade no processamento. Essa abordagem permite que os neurônios extraiam características locais como bordas e texturas, tornando as CNNs eficazes na extração de características em imagens (ZEILER; FERGUS, 2014).

Os neurônios de convolução em CNNs aplicam operações de convolução a pequenas regiões dos dados de entrada usando filtros. Cada neurônio pondera essas regiões e aplica uma função de ativação, resultando na extração de características locais. A hierarquia de camadas

de neurônios de convolução permite a aprendizagem de características complexas, incluindo detecção de bordas, texturas e padrões em diferentes níveis de abstração (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

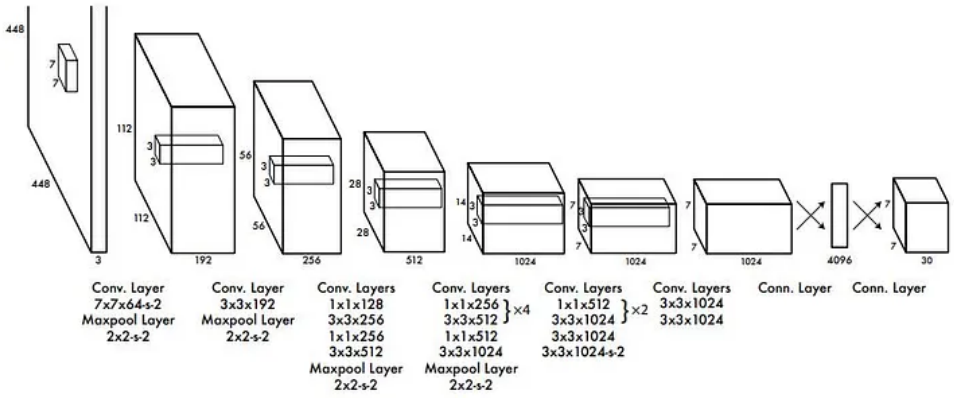
O treinamento de CNNs envolve o ajuste iterativo dos pesos dos neurônios por meio do algoritmo *backpropagation*. Durante o treinamento, a rede é exposta a um conjunto de dados de treinamento, e os pesos são atualizados para minimizar a função de perda, que quantifica o erro entre as previsões da rede e os rótulos reais. Esse processo de otimização é repetido por várias épocas até que a rede alcance um desempenho satisfatório na tarefa desejada (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.4 *You Only Look Once* (YOLO)

YOLO, acrônimo para "*You Only Look Once*", é uma abordagem inovadora para detecção de objetos em imagens que usa CNN como extratores de recursos. Criado por Joseph Redmon, o YOLO difere dos algoritmos anteriores, como *Region-based Convolutional Neural Networks* (R-CNN) ou *Faster* R-CNN, porque simplifica o processo, exigindo apenas uma única observação de imagem pela rede neural convolucional para que esta detecte os objetos contidos nela. Este recurso exclusivo fornece taxas de detecção superiores em comparação com métodos concorrentes sem sacrificar a precisão do YOLO, solidificando sua posição como uma abordagem eficiente e precisa para detecção de objetos (REDMON et al., 2015).

Essa abordagem destaca a capacidade do YOLO de considerar objetos em diferentes escalas, proporcionando uma detecção abrangente em diversas situações. Em termos de detecção de objetos, como as suas precursoras, a YOLO tem o objetivo de detectar e marcar os objetos nas imagens por meio de "*bounding boxes*"(caixas delimitadoras). Essas representam regiões retangulares ao redor dos objetos identificados, fornecendo uma estrutura que delimita visualmente a área em que o objeto está contido. Essas *bounding boxes* são fundamentais para calcular a exatidão da localização dos objetos detectados na imagem digital. A CNN YOLO, em sua primeira versão publicada por Redmon et al. (2015) está esquematizada na Figura 1.

Figura 1 – Arquitetura do YOLO.



Fonte: (REDMON; FARHADI, 2018).

2.4.1 Funcionamento do YOLO)

O algoritmo YOLO aborda a detecção de objetos primeiramente como um problema de regressão, pois intenciona marcar os objetos na imagem por meio das suas coordenadas cartesianas (x_{centro_objeto}, y_{centro_objeto}), $altura_{caixa}$, $largura_{caixa}$ e confiança de acerto da delimitação do objeto. Também é, em si, um problema de classificação, pois, além de detectar a posição do objeto, classifica-o em um rótulo conhecido (bicicleta, cachorro, carro, etc.).

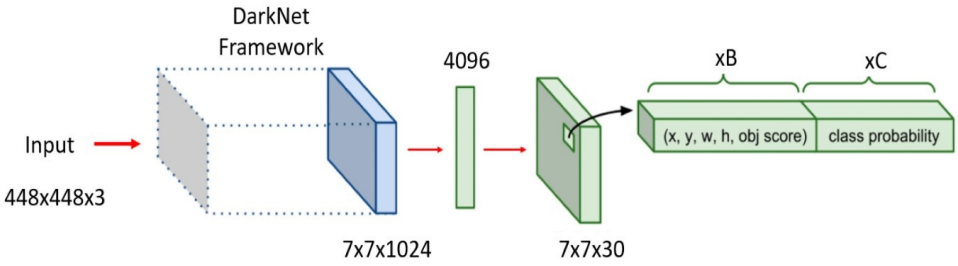
Inicialmente, o algoritmo recorta a imagem de entrada (dimensões quadradas e tamanho fixo) em uma grade (*grid*) de células, divisões de *S* (linhas/colunas totalizando *SxS* células). Sua primeira versão publicada por Redmon et al. (2015), utilizavam entrada de 428x428 *pixels* e as repartiam em *grids* de 7x7 linhas/colunas, versões mais recentes optam por grades contendo até 80×80 células e tamanhos de imagens de entrada de 640x640 *pixels*. A Figura 3 (a) toma o mesmo exemplo utilizado por Redmon et al. (2015), quando publicou a primeira versão da YOLO com uma *grid* de 7 linhas por 7 colunas.

Cada célula é encarregada de prever um conjunto de *B* caixas delimitadoras (*bounding boxes*) com suas informações de localidade x_{centro_objeto} e y_{centro_objeto}, percentuais em relação à célula onde está contida. Ainda para cada caixa, a rede prevê $altura_{caixa}$ e a $largura_{caixa}$ que delimita o objeto (medidas percentuais em relação à imagem completa) e a confiança que o modelo tem que aquela caixa realmente seleciona um objeto de interesse na imagem. Assim, para uma YOLO com grades de 7x7, com cada grade sendo responsável por sugerir *B* caixas delimitadoras, teremos uma saída com $7x7xB = 49B$ caixas delimitadoras. Sabendo que cada caixa contém 5 informações (x_{centro_objeto}, y_{centro_objeto}, $altura_{caixa}$, $largura_{caixa}$ e confiança), teremos $49Bx5 = 245B$ números na saída do algoritmo YOLO. Entender essa aritmética acerca do YOLO nos permite afirmar que o algoritmo é limitado a detectar, no máximo, *SxSxB* objetos em uma imagem, onde *S*: número de linhas/colunas na grade e *B*: número de caixas por célula da grade.

Adicionalmente, cada célula da grade contém associada a ela "*C*"valores que especulam qual a "classificação"do objeto centrado naquela célula. Imagine que a rede reconheça apenas duas classes ($C = 2$, exemplo: gatos e cachorros). Para cada célula, teremos a informação de classificação para presença de gato e cachorro. Normalmente esses "*C*"valores presentes em cada célula são sugeridos, em textos científicos, como probabilidades da célula conter um objeto reconhecível, entretanto é prudente compreender que essa medida está relacionada à percepção da rede de que a célula contém um objeto de interesse e não à probabilidade dele estar presente na célula. Assim sendo, tomando outro exemplo na Figura 1, para YOLO com divisões de grade de 7x7, onde cada grade contém $B = 2$ caixas delimitadoras e $C = 20$ classes reconhecíveis, tem-se um tensor de saída de $7x7x30$. Pode-se ver todo o processo na Figura 2.

No contexto da classificação das *bounding boxes* no YOLO, cada caixa delimitadora está associada a uma célula específica em uma grade, onde encontra-se seu centro. Com a informação "C", cada uma dessas células atua como um classificador, sugerindo não apenas a região onde localiza-se o centro da caixa, mas, também, a predição da classe do objeto contido nela. Assim,

Figura 2 – Arquitetura preliminar YOLOv1.

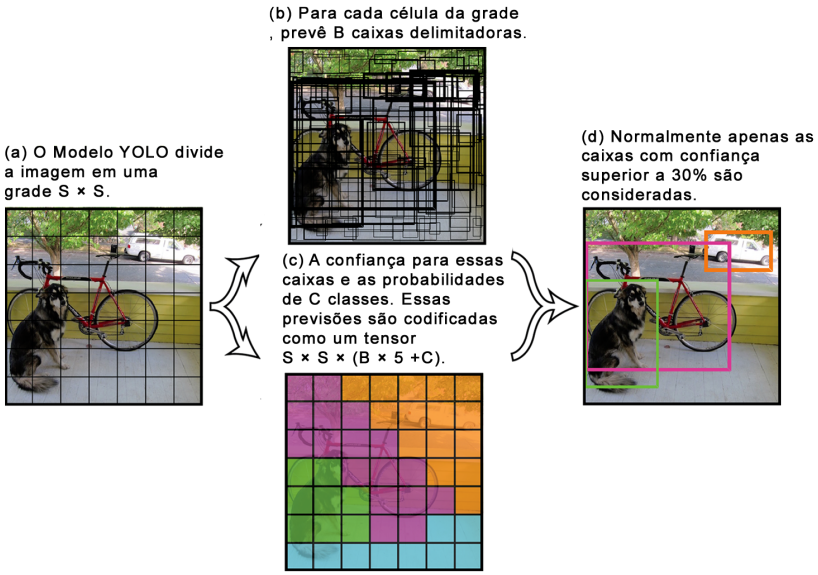


Esquema simplificado demonstrando as dimensões de entrada e as características visualizadas na saída para a YOLOv1 Fonte: (HUSSAIN, 2023).

ao multiplicarmos a confiança da célula em conter o objeto nas coordenadas já descritas e o maior valor de "C"daquela célula, a rede neural determine a confiança de um objeto detectado pertencer a uma classe específica a que ela foi treinada. Note, na Figura 3-c o centro de cada objeto repousa em uma célula classificada, respectivamente, como cachorro (verde), bicicleta (rosa) e carro (laranja).

A confiança do modelo em detectar e classificar um objeto por meio da caixa delimitadora é, então, uma integração entre as medidas de confiança de ter encontrado um objeto (Figura 3-b) e a predição da classe do objeto por meio das classificação de cada célula em que o objeto está (Figura 3-c). Essa integração produz um resultado final (Figura 3-d) que representa a confiança geral de que a caixa contenha um objeto de interesse e que ele possui a classificação apontada pelo modelo. Usando uma grade 7x7 e $B = 5$ caixas detectáveis por célula, são possíveis 245 caixas delimitadoras. A Figura 3 ilustra o processo completo da primeira versão da YOLO.

Figura 3 – Funcionamento do YOLO.



Etapas do experimento seminal de Redmon et al. (2015). Fonte: (REDMON; FARHADI, 2018)

Apesar do modelo sugerir a existência de várias caixas ($S \times S \times B$), a maioria delas apresenta pontuações de confiança muito baixas. Portanto, normalmente, apenas as caixas com pontuações finais consideradas suficientes são mantidas, essa pontuação é chamada de limiar

(*threshold*) que pode ser ajustado de acordo com a precisão desejada. O valor padrão do *threshold* pode variar dependendo da versão da YOLO utilizada. Por exemplo, para a YOLOv8, o *threshold* de confiança padrão é definido como 25%. Nesse contexto, apenas as detecções com pontuações finais superiores a 25% são consideradas (JOCHER; CHAURASIA; QIU, 2023a). As Figuras 3 (b) e (d) ilustram o processo de redução da quantidade de caixas delimitadoras para a YOLOv1.

Mesmo após a etapa de seleção das *bounding boxes* de interesse com confiança acima do *threshold*, é bem possível que um mesmo objeto na imagem seja detectado múltiplas vezes por diferentes células. Isso decorre especialmente do fato do centro do objeto estar próximo às bordas entre as células. Assim, com vistas a lidar com a redundância nas previsões, é utilizada a técnica *Non-Maximum Suppression* (NMS), ou seja, supressão não máxima. Essa estratégia assegura que, ao identificar *bounding boxes* sobrepostas, apenas a caixa com a maior confiança seja mantida, descartando as demais que possuem sobreposição significativa medida pela métrica *Intersection over Union* (IoU) (HOSANG; BENENSON; SCHIELE, 2017).

A IoU avalia quão sobrepostas estão duas ou mais caixas delimitadoras que se interceptam. Quanto maior o valor da IoU, maior é a sobreposição entre as regiões delimitadas por essas caixas e maior é a possibilidade das caixas estarem detectando o mesmo objeto na imagem. A aplicação da técnica NMS é crucial para evitar múltiplas detecções do mesmo objeto. Na Figura 4 (a), o cachorro é detectado pelas caixas verde (confiança: 98%), azul (confinança: 90%) e cinza (confiança: 82%). Tendo suas áreas sobrepostas além de um limiar, a caixa de maior confiança sobressairá frente as demais, Figura 4 (b). O limiar padrão para a IoU adotado pela técnica NMS pode variar de acordo com a versão do YOLO, para a versão 8, por exemplo, este valor é de 70% (JOCHER; CHAURASIA; QIU, 2023a). Para mais detalhes sobre a métrica IoU consulte a seção 2.6.6.

Figura 4 – Funcionamento do NMS.

(a) O modelo detecta múltiplas vezes o *pet* de interesse com caixas delimitadoras com IoU acima de um limiar, 70% para YOLOV8, por exemplo.

(b) A técnica NMS é aplicada e a caixa delimitadora com maior confiança é mantida.

Exemplificação da técnica NMS Fonte: (SINGH, 2020)

2.4.2 Evolução do YOLO

O YOLO, desde a publicação feita por Redmon et al. (2015), acumulou várias melhorias de desempenho em precisão e rapidez. Relaciona-se, a seguir, as diferentes versões:

- **YOLOv1 (2015):** A primeira versão do YOLO, baseada no *framework Darknet* escrito em C e *Compute Unified Device Architecture* (CUDA), apresentou duas subvariantes. A primeira arquitetura, composta por 24 camadas convolucionais, tinha a camada final conectada à primeira das duas camadas totalmente conectadas. Já a variante *'Fast YOLO'* consistia em apenas nove camadas convolucionais, cada uma hospedando menos filtros em comparação com a camada antecessora na rede (HUSSAIN, 2023).

  A arquitetura simplificada do YOLO, junto com sua regressão inovadora em uma única etapa para a imagem completa, proporcionou uma notável velocidade em comparação com os detectores de objetos existentes, permitindo desempenho em tempo real. No entanto, essa eficiência veio acompanhada de uma desvantagem, pois, apesar de sua rapidez, o YOLO apresentava um maior erro de localização em comparação com métodos de última geração, como o *Fast* R-CNN (GIRSHICK, 2015). Essa limitação tinha três principais causas: a capacidade restrita de detectar, no máximo, dois objetos da mesma classe que compartilhassem seu centro em uma mesma célula da grade, limitando sua aptidão para prever objetos próximos; a dificuldade em prever objetos com proporções de aspecto não vistas nos dados de treinamento; e a aprendizagem a partir de características mais grosseiras de objetos devido às camadas de subamostragem (TERVEN; CORDOVA-ESPARZA, 2023).

  Nesta versão, o limiar de confiança para que uma caixa delimitadora seja considerado como um objeto detectado pelo algoritmo é fixado em 20%. Enquanto isso, para o algoritmo de NMS, a configuração padrão do limiar de IoU ficou estabelecido em 50%, indicando a sobreposição mínima necessária para que duas ou mais detecções sejam consideradas como sendo do mesmo objeto. Esses hiperparâmetros podem ser consultados em (REDMON, 2024).

- **YOLOv2 (2016):** proporcionou maior precisão e velocidade na detecção de objetos quando comparado à versão anterior. Uma técnica-chave implementada para aprimorar o desempenho do modelo foi a *Batch Normalization* (normalização em lote), que normaliza as entradas de uma camada em lotes, mantendo a escala dos valores que propagam na rede (etapa *feedforward*) na mesma escala. Essa técnica acaba por facilitar o treinamento, resultando em uma convergência mais rápida e estável (HUSSAIN, 2023).

  Outra evolução interessante foi a substituição das camadas totalmente conectadas ao final da rede por **caixas âncora** (*anchor box*). Enquanto as camadas conectadas tinham o objetivo de prever diretamente as coordenadas dos objetos na imagem, as caixas âncora são como molduras retangulares padrões para cada classe de objeto, estrategicamente

escolhidas durante a fase de treinamento. Nesta fase, as **caixas âncora** são escolhidas com base no conjunto de dados, atuando como referências para capturar objetos de diferentes tamanhos e proporções. Durante a fase de inferência, o modelo ajusta essas caixas para melhor se adaptarem aos objetos na cena, resultando em uma detecção mais precisa e eficiente em comparação com a abordagem anterior. Enquanto as camadas totalmente conectadas tinham uma abordagem mais direta e uniforme, as caixas âncora oferecem uma maior flexibilidade, permitindo ao modelo detectar uma variedade de objetos de forma mais eficiente e adaptável, como podemos ver na Figura 5.

Figura 5 – Funcionamento das Caixas Âncoras, introduzidas na YOLOv1.

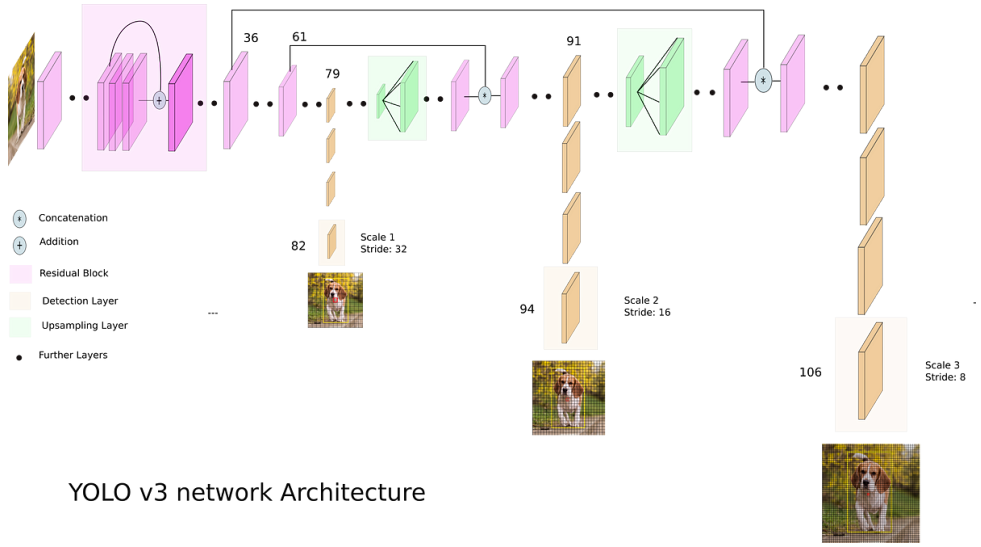
Fonte: (KO; KO; LEE, 2024)

Essas inovações, aliadas ao treinamento em imagens de maior resolução, capacitaram o YOLOv2 a detectar mais de 9 mil categorias de objetos de maneira mais eficaz, tornando-o uma ferramenta avançada para tarefas de visão computacional. Em resumo, melhorias como normalização em lote (*batch normalization*), caixas de âncora e ajustes nas camadas conectadas contribuíram para um YOLOv2 mais preciso, rápido e versátil (TERVEN; CORDOVA-ESPARZA, 2023). O limiar padrão de IoU para o algoritmo de NMS neste modelo é estabelecido em 50%, enquanto o de confiança é fixado em 20%, iguais à versão 1 (REDMON, 2024).

- **YOLOv3 (2018):** apresentado por Redmon e Farhadi (2018), introduziu uma série de melhorias notáveis, incluindo avanços significativos na precisão, resultando em detecções mais precisas quando comparado ao seu predecessor. No entanto, apesar desses ganhos em precisão, o YOLOv3 não se destaca por sua velocidade em comparação com versões anteriores. A principal inovação reside na abordagem de predição da imagem em três diferentes escalas, reduzindo as imagens em 32, 16 e 8 vezes, respectivamente. Essa abordagem visa manter a acurácia mesmo em tamanhos menores, superando a dificuldade de versões anteriores em lidar com objetos muito pequenos, como ilustrado na Figura 6. O modelo fixa o *threshold* de confiança em 25% e utiliza limiar padrão de 45% de IoU para NMS (ULTRALYTICS, 2024a).

- **YOLOv4 (2020):** Publicado por Bochkovskiy, Wang e Liao (2020), destaca-se como o estado da arte em detecção de objetos em tempo real, incorporando inovações notáveis.

Figura 6 – Arquitetura do YOLOv3.



Fonte: (NEWMAN et al., 2023).

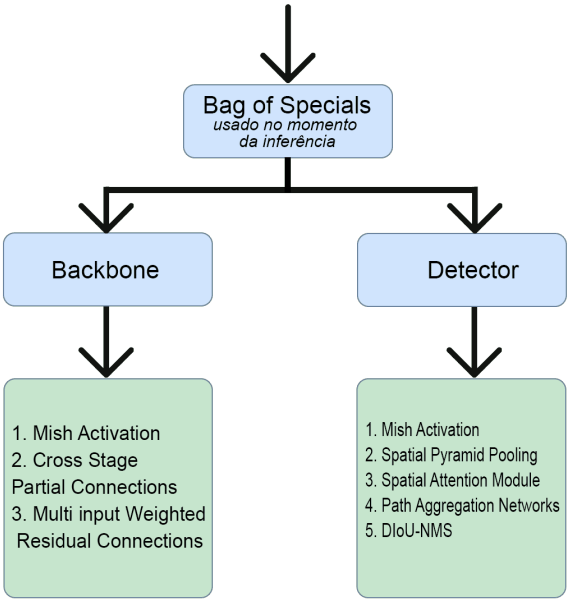
Uma dessas inovações é a técnica *Bag-of-Specials* (BoS), uma abordagem que envolve uma cuidadosa seleção de características especiais.

O BoS é uma técnica que processamento de imagens, vai além da extração de características visuais simples, como cores e formas. Ela utiliza uma CNN para identificar padrões relevantes, como objetos específicos, em imagens. Esses padrões são usados para criar regiões de interesse (*Region of Interest* (ROI)s), partes da imagem com informações relevantes. As ROIs são agrupadas em conjuntos usando técnicas de agrupamento, onde cada conjunto representa um **"especial"** contendo características similares. Esses **"especiais"** capturam aspectos únicos da imagem pertinentes à análise. A imagem é então representada como um histograma desses **"especiais"**, mostrando quantas vezes cada um aparece na imagem. Essa representação compacta e informativa pode ser aplicada em diversas áreas, como classificação de imagens e detecção de objetos. Podemos ver os diferentes métodos presentes no *Bag-of-Freebies* (BoF) na Figura 7.

A BoS incorpora o *backbone* da rede neural *CSPDarknet-53* e o agregador de características da rede neural *PANet*. Tais modificações não apenas aprimoram a precisão do modelo, mas também melhoram sua eficiência computacional, possibilitando uma execução mais rápida em *Graphics Processing Unit* (GPU) com menor consumo de memória.

O YOLOv4 introduziu a abordagem de treinamento conhecida como BoF, conforme discutido por Zhang et al. (2019). Essa abordagem compreende um conjunto estratégico de técnicas de treinamento destinadas a aprimorar a precisão global de um modelo de detecção de objetos, abrangendo tanto métodos convencionais, como *data augmentation*, quanto inovações específicas, como a técnica de mosaico. Além das transformações regulares, como ajuste de brilho e recorte, os desenvolvedores incorporaram práticas como *DropBlock*, uma forma de regularização que funciona através da seleção aleatória de regiões contíguas

Figura 7 – Diferentes métodos presentes no Bag of Specials.

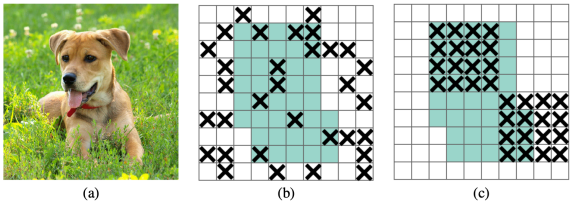


Fonte: (PATTANSHETTI; NIVADE, 2021).

em um mapa de características *(feature map)* e do descarte de todas as unidades dentro dessas regiões durante o treinamento, ou seja, é como remover aleatoriamente blocos de peças do tabuleiro, forçando a rede neural a aprender a jogar com um conjunto de peças incompleto. Isso a torna mais robusta e adaptável a diferentes situações, pois precisa aprender a lidar com a falta de informações. A aplicação de *DropBlock* estimula a rede a aprender características mais robustas e diversas, especialmente em camadas convolucionais, onde as unidades de ativação são espacialmente correlacionadas (GHIASI; LIN; LE, 2018), como é exemplificado na Figura 8. Essa estratégia visa prevenir o *overfitting*, um fenômeno no qual a rede se ajusta excessivamente aos dados de treinamento, mas falha em generalizar para novos dados. Além disso, foram implementadas técnicas como a suavização de rótulos de classe para fortalecer a capacidade de generalização do modelo. Uma contribuição notável é a inclusão da *Complete Intersection over Union* (CIoU), uma variação da IoU que leva em conta não apenas a sobreposição espacial entre as caixas delimitadoras previstas e verdadeiras, mas também considera a distância entre seus centros e as diferenças de aspecto. Detalhes adicionais sobre a CIoU podem ser encontrados na seção 2.6.6.

Outra técnica distintiva empregada no YOLOv4 é o *Self-Adversarial Training* (SAT), uma estratégia inovadora projetada para reforçar a robustez do modelo. Nesse método, introduzem-se perturbações intencionais nas imagens de treinamento por meio de ataques adversariais (*adversarial attacks*), simulando a ausência do objeto real. Essa abordagem desafia o modelo a manter sua precisão mesmo em face de situações enganosas, conforme discutido por (JIAO et al., 2023). Por exemplo, em uma imagem contendo um cachorro, é possível aplicar distorções leves para induzir o modelo a erroneamente pensar que não há

Figura 8 – Exemplo de Execução do *Dropblocks*.



(a) imagem de entrada para uma rede neural convolucional. As regiões verdes em (b) e (c) incluem as unidades de ativação que contêm informações semânticas na imagem de entrada. Eliminar ativações aleatoriamente não é eficaz na remoção de informações semânticas porque as ativações próximas contêm informações intimamente relacionadas. Em vez disso, eliminar regiões contínuas pode remover certas informações semânticas (por exemplo, cabeça ou pés) e, conseqüentemente, forçar as unidades restantes a aprender recursos para classificar a imagem de entrada.

Fonte: (GHIASI; LIN; LE, 2018).

um cachorro presente. Dessa forma, o modelo aprende a ser mais preciso mesmo quando confrontado com ambiguidades, assemelhando-se a um treinamento constante diante de desafios e truques.

Em suma, esta versão trouxe várias inovações, destacando-se não apenas por sua excepcional capacidade de detecção de objetos em tempo real, mas, também, pelo compromisso contínuo com avanços significativos em arquitetura, treinamento e robustez, proporcionando uma solução de ponta para diversas aplicações em visão computacional. As configurações padrão incluem uma limiar IoU de 50% para NMS e um limiar de confiança de 20% (BOCHKOVSKIY, 2024).

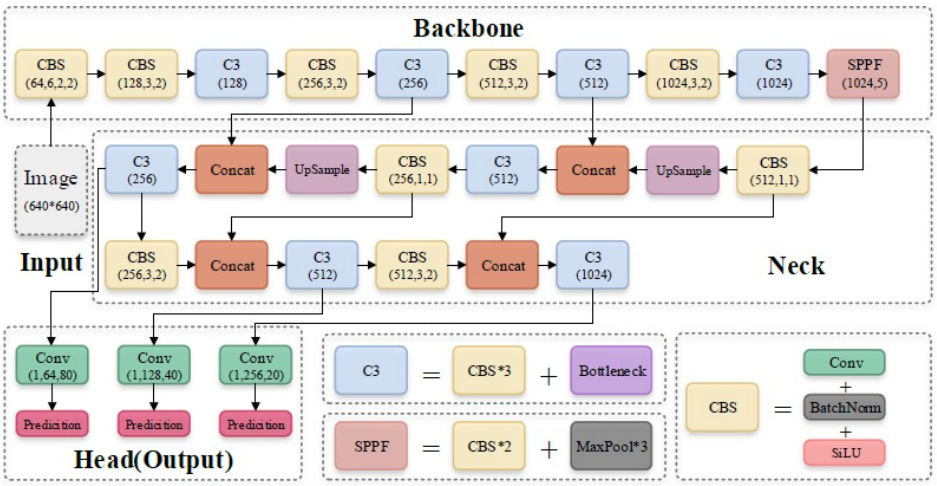
- **YOLOv5 (2020):** publicada por Jocher (2020) alguns meses após o YOLOv4, representa uma evolução notável na detecção de objetos em tempo real. Diferentemente do YOLOv4, o YOLOv5 foi desenvolvido com base no *framework PyTorch* escrito em python, tornando-o mais acessível para implementações práticas. Mantido ativamente pela *Ultralytics*, com mais de 250 colaboradores, o YOLOv5 destaca-se pela facilidade de uso, treinamento e implementação, incluindo uma versão móvel para *iOS* e Android.

  A arquitetura essencial do YOLOv5 envolve três pilares-chave: *Backbone* para extração de características, *Neck* focado na agregação de características e *Head* para gerar detecções. Semelhante ao YOLOv4, o YOLOv5 reúne e refina técnicas de visão computacional para melhorar o desempenho, como podemos ver na Figura 9.

  Destaca-se, também, o conceito de "aprendizado automático de caixas âncora", integrado ao *pipeline* do YOLOv5. Isso permite que a rede aprenda automaticamente as caixas âncora mais adequadas para o conjunto de dados específico, acelerando o processo de treinamento. O limiar de IoU para a NMS é de 45%, enquanto o *threshold* de confiança é de 25% (ULTRALYTICS, 2024b).

- **YOLOX** (2021)**:** publicado em julho de 2021 pela Megvii Technology (GE et al., 2021), representa um avanço significativo na detecção de objetos. Desenvolvido em *Pytorch* e construído sobre a base do YOLOV3 da *Ultralytics*, este modelo introduziu cinco mudanças principais em relação ao seu predecessor. Em primeiro lugar, adotou uma

Figura 9 – Estrutura de rede padrão do YOLOv5.



Fonte: (LIU et al., 2022).

arquitetura livre de âncoras, abandonando a abordagem de detecção baseada em âncoras que caracterizou versões anteriores do YOLO. Inspirado por detectores de objetos de ponta sem âncoras, como *CornerNet*, que representa um objeto por dois pontos-chave: o canto superior esquerdo e o canto inferior direito. Diferente de métodos tradicionais, ele não usa regressões para prever coordenadas da caixa delimitadora, mas sim prevê diretamente esses pontos através de uma única CNN. O *CornerNet* introduziu a camada de agrupamento de cantos, a qual auxilia a rede na localização precisa desses cantos, resultando em um modelo eficiente e preciso para tarefas de detecção de objetos (LAW; DENG, 2019), o *CenterNet*, que é outro detector de objetos, onde ele representa cada objeto por um conjunto de três pontos-chave: o ponto central e dois pontos adicionais para refinar a localização e tamanho do objeto. Essa abordagem, em comparação ao *CornerNet*, oferece maior precisão e capacidade de recuperação de objetos. O *CenterNet* utiliza uma única rede neural convolucional para extrair características da imagem e realizar previsões, além de realizar regressões para determinar propriedades adicionais do objeto, como tamanho, orientação e até mesmo pose (em detecção de pessoas) (DUAN et al., 2019), e o *Fully Convolutional One-Stage Object Detection* (FCOS), outro modelo de detecção de objetos que se diferencia por não utilizar caixas âncoras. Ao invés disso, ele realiza a previsão direta de objetos por pixel, similar à segmentação semântica. Essa abordagem torna o FCOSmais eficiente e preciso do que modelos tradicionais, além de simplificar o processo de detecção e reduzir a quantidade de hiperparâmetros a serem ajustados. O FCOS utiliza uma rede totalmente convolucional para processar a imagem em paralelo, possibilitando uma detecção rápida e precisa de diversos tipos de objetos em uma única imagem. O modelo é ideal para aplicações de visão computacional em diversas áreas, como robótica, veículos autônomos, segurança e monitoramento. (TIAN et al., 2019).

Outra inovação crucial foi a separação do *Head* do modelo em duas partes distintas: uma para tarefas de classificação e outra para tarefas de regressão. Essa abordagem melhorou o

Average Precision (AP) em $1,1$ pontos percentuais e acelerou a convergência do modelo. O YOLOX também aprimorou a atribuição de rótulos, adotando uma versão simplificada do problema de *Optimal Transport* (OT), chamada *Simple Optimal Transport Assignment* (simOTA), onde o modelo faz uso das caixas delimitadoras que são divididas em grupos positivos e negativos. As caixas delimitadoras do grupo positivo são consideradas boas previsões que delimitam um objeto, enquanto as caixas delimitadoras do grupo negativo são consideradas más previsões que não delimitam um objeto, resultando em um aumento adicional de $2,3$ pontos percentuais no AP.

Por fim, os autores incorporaram *augmentations* robustas, como *MixUP*, que combina duas imagens diferentes em uma única imagem. As duas imagens são combinadas de tal forma que partes de uma imagem são visíveis e partes da outra imagem também são visíveis. Além disso, os rótulos dessas duas imagens também são combinados de acordo (ZHANG et al., 2018), e *Mosaic*, que pega quatro imagens e as combina em uma, ele faz isso redimensionando cada uma das quatro imagens, costurando-as juntas e, em seguida, pegando um recorte aleatório das imagens costuradas para obter a imagem *Mosaic* final, eliminando a necessidade de pré-treinamento e aumentando o AP em $2,4$ pontos percentuais (KIMATA; NITTA; TAMAKI, 2022). Essas inovações culminaram em um modelo que alcançou resultados de ponta em 2021, oferecendo um equilíbrio otimizado entre velocidade e precisão na detecção de objetos. Os hiperparâmetros padrões do YOLOX para IoU para o algoritmo NMS é 65%, enquanto o limiar de confiança é de 25% (Ge, Zheng and Liu, Songtao and Wang, Feng and Li, Zeming and Sun, Jian, 2024).

- **YOLOv6 (**2022**):** Lançado pela equipe do Departamento de Visão de Inteligência Artificial da Meituan, o YOLOv6 incorpora diversas inovações significativas (LI et al., 2023). Uma contribuição destacada é a introdução do *EfficientRep*, uma arquitetura CNN otimizada para *hardware*, especialmente compatível com unidades de processamento gráfico (GPU). Inspirado na estrutura do *RepVGG* (WENG et al., 2023), o *EfficientRep* utiliza uma técnica de reparametrização estrutural, desvinculando a arquitetura durante o treinamento e a inferência. Isso resulta em um modelo simplificado e eficiente (DING et al., 2021), oferecendo um paralelismo superior em comparação com as iterações anteriores do YOLO.

  Além disso, o YOLOv6 adota a topologia Personal Area Network (PAN) para o *Neck* da rede, aprimorada com blocos *RepBlocks*. Esses blocos, fundamentais na arquitetura da rede convolucional *RepVGG*, consistem em uma sequência de convolução 3x3 seguida por uma função de ativação *ReLU*. O *CSPStackRep* é outra inovação incorporada, sendo uma versão reparametrizada do *CSP Block* utilizado na arquitetura YOLOv6. Esta adaptação visa melhorar a eficiência e o desempenho, especialmente para modelos maiores.

  O YOLOv6 implementa um *Head* desacoplado eficiente, seguindo a abordagem do YOLOX (TERVEN; CORDOVA-ESPARZA, 2023). Essa estratégia aprimorada no *Head* contribui para uma maior eficiência durante as operações de inferência, consolidando o

posicionamento do YOLOv6 como uma solução de vanguarda no campo da detecção de objetos.

O modelo adota a inovadora abordagem de aprendizado de alinhamento de tarefas do *Task-aligned One-stage Object Detection* (TOOD), uma técnica de detecção de objetos que realiza uma explícita harmonização entre a classificação e a localização dos objetos, fundamentada em aprendizado. Essa estratégia resulta em detecções de objetos mais precisas e confiáveis (FENG et al., 2021). Esse enfoque é ampliado com a incorporação de novas perdas de classificação e regressão, notavelmente o *loss* de classificação *Varifocal loss* (VFL). Derivada do *loss* focal, a VFL se destaca ao adotar uma abordagem assimétrica. Essa abordagem inclui a medida de IoU, além do *loss* de regressão *Scylla-Intersection over Union* (SIoU)/*Generalized Intersection over Union* (GIoU) (ZHANG et al., 2021).

A abordagem SIoU, fundamentada na métrica IoU, leva em consideração a regressão das caixas delimitadoras, abrangendo aspectos como forma, distância e desalinhamento na proporção de aspecto (ZHANG; ZHANG, 2023). Já a GIoU representa uma extensão da métrica IoU, incorporando a geometria das caixas delimitadoras, além da sobreposição. O cálculo da GIoU considera a menor caixa delimitadora que envolve ambas as caixas delimitadoras (prevista e real), juntamente com a interseção e união entre elas.

Uma estratégia de auto-destilação é empregada para aprimorar as tarefas de regressão e classificação, adicionando uma camada de sofisticação ao processo de treinamento do modelo. Essa abordagem contribui para a eficácia geral do modelo, reforçando sua capacidade de aprendizado e generalização em cenários diversos.

Finalmente, o YOLOv6 implementa um esquema de quantização para detecção, utilizando o *RepOptimizer*, onde a ideia central é a reparametrização do gradiente, que facilita o treinamento eficiente de modelos muito simples e amigáveis à quantização (DING et al., 2023), e a destilação por canal, resultando em um detector mais rápido e eficiente. Os hiperparâmetros padrões do YOLOv6 para o algoritmo NMS é de 45% de IoU, enquanto o limiar de confiança é 25% (MEITUAN, 2023).

- **YOLOv7 (2022):** apresentou mudanças substanciais em sua arquitetura e abordagem em relação ao YOLOv6. Duas reformas arquitetônicas destacadas foram implementadas. A primeira foi a introdução da *Extended Efficient Layer Aggregation Network* (E-ELAN) ou Rede de Agregação Eficiente Estendida, inspirada em avanços na eficiência de rede. Esta estratégia otimiza fatores como custo de acesso à memória, proporção de canais de entrada/saída e caminho de gradiente no *Backbone* do YOLOv7, aprimorando a precisão e velocidade de detecção (WANG; BOCHKOVSKIY; LIAO, 2022).

  A segunda reforma foi o dimensionamento composto do modelo, visando atender a uma variedade de requisitos de aplicação. Ao contrário de métodos convencionais independentes, o YOLOv7 adotou um mecanismo de dimensionamento composto, ajustando largura

e profundidade de forma coerente para redes baseadas em concatenação, mantendo uma estrutura ideal.

Além disso, foram implementadas estratégias de reparametrização para tornar a rede mais robusta, permitindo que segmentos regulassem suas próprias estratégias de reparametrização. A introdução de *Heads* auxiliares de nível grosseiro a fino buscou melhorar o treinamento, embora de maneira menos eficiente do que o *Head* principal.

Posteriormente, o YOLOv7 se destacou ao transcender outros detectores em termos de velocidade e precisão. Notavelmente, este modelo foi treinado de maneira exclusiva com o conjunto de dados MS *Common Objects In Context* (COCO), dispensando o uso de *backbones* pré-treinados. Uma inovação significativa foi a introdução do conceito de "BoF"(*Backbone Optimization Framework*), visando elevar a precisão sem comprometer a agilidade da inferência.

Dentro desse *framework*, foram implementadas estratégias engenhosas, como a convolução planejada reparametrizada, atribuições específicas de rótulos para *Heads* auxiliares e principais, e a normalização em lote na sequência *conv-bn-activation*. Esta sequência, comum em diversas arquiteturas de redes neurais, inclui uma camada convolucional, seguida por uma camada de *batch normalization* e, por fim, uma função de ativação. Essas modificações foram fundamentais para otimizar o desempenho global do modelo.

Além disso, o YOLOv7 incorporou a utilização de média móvel exponencial como modelo final de inferência. Essa escolha estratégica contribuiu para aprimorar a estabilidade e a robustez do modelo durante as operações de inferência, consolidando ainda mais sua posição como um detector de objetos de ponta, capaz de equilibrar efetivamente velocidade e precisão em tarefas desafiadoras.

Em comparação com versões anteriores, o YOLOv7 alcançou uma redução significativa em parâmetros e computação, mantendo ou aprimorando a AP. Essas mudanças consolidaram o YOLOv7 como uma evolução poderosa e eficiente na detecção de objetos. É importante observar que os valores padrão dos hiperparâmetros de limiar de IoU para o algoritmo de NMS e *threshold* de confiança são respectivamente 45% e 25%, e podem ser encontrados no *GitHub* oficial, disponível em Wang, Bochkovskiy e Liao (2024).

- **YOLOv8:** desenvolvida pela *Ultralytics*, baseia-se no sucesso das suas versões anteriores. Este modelo foi projetado para realizar a detecção e classificação de objetos em uma cena de maneira rápida e precisa, introduzindo novos recursos e melhorias para aprimorar o desempenho, flexibilidade e eficiência.

  Para otimizar a precisão das detecções, o YOLOv8 utiliza hiperparâmetros específicos, incluindo um limiar de IoU para o algoritmo de NMS fixado em 70% e um limiar de confiança estabelecido em 25%. Esses valores são fundamentais para controlar o comportamento do modelo durante a detecção de objetos, garantindo que apenas as detecções mais confiáveis e precisas sejam consideradas (JOCHER; CHAURASIA; QIU, 2023b).

Além disso, é importante destacar que o YOLOv8 oferece suporte a uma ampla gama de tarefas de visão computacional, como detecção, segmentação, estimativa de pose, rastreamento e classificação. Essa versatilidade permite que os usuários explorem e apliquem o YOLOv8 em diversos contextos e domínios, adaptando-o às necessidades específicas dessas aplicações.

Assim como acontece com outras versões mais recentes do YOLO, os criadores forneceram o modelo pré-treinado em cinco variantes: YOLOv8n (*nano*), YOLOv8s (*small*), YOLOv8m (*medium*), YOLOv8l (*large*) e YOLOv8x (*extra large*).

A versão *nano* destaca-se como a opção mais ágil e leve, proporcionando uma execução mais rápida e exigindo menor capacidade de processamento. Entretanto, é importante observar que essas vantagens vêm acompanhadas de uma redução na qualidade das detecções. Por outro lado, a versão *extra large* (YOLOv8x) se destaca por oferecer uma precisão substancialmente superior, mas requer recursos computacionais mais robustos, resultando em um desempenho ligeiramente inferior durante a execução.

Cada variante do modelo atende a requisitos específicos, permitindo aos usuários escolher a opção mais adequada com base nas necessidades individuais de velocidade, precisão e capacidade computacional disponível em suas aplicações. A Figura 10 apresenta uma visualização comparativa do desempenho entre as diferentes variantes do YOLOv8, auxiliando na tomada de decisões informadas sobre a escolha do modelo mais apropriado para cada cenário de uso.

Figura 10 – Comparação de desempenho das variantes YOLOv8 utilizando o *dataset* COCO.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Fonte:(JOCHER; CHAURASIA; QIU, 2023a).

2.5 Anotação de Imagens: explorando o Padrão YAML

O protocolo de anotação utilizado pelo YOLO para imagens é representado por um arquivo de texto simples contendo detalhes sobre as caixas delimitadoras dos objetos presentes na imagem. Cada linha neste arquivo corresponde a uma caixa delimitadora e contém informações

importantes, como classe do objeto ,coordenadas e tamanho das caixas delimitadoras. Este arquivo tem o seguinte formato:

<*object-class*> <*x*> <*y*> <*width*> <*height*>

- ***<object-class>***: representa o nome da classe do objeto.
- **<x>**: é a coordenada x do centro da caixa delimitadora.
- **<y>**: é a coordenada y do centro da caixa delimitadora.
- ***<width>***: é a largura da caixa delimitadora.
- ***<height>***: é a altura da caixa delimitadora.

Os arquivos *Yet Another Markup Language* (YALM) ou, em português, "Mais uma Linguagem de Marcação", por outro lado, desempenham um papel fundamental no YOLO, pois atuam como arquivos de configuração que definem parâmetros importantes do modelo de detecção de objetos. Este arquivo contém dados importantes sobre o modelo, incluindo: caminho para o conjunto de dados de treinamento (*train: path to train data*), caminho para o conjunto de dados de validação (*val: path to validation data*), número de classes (*nc: number of classes*), nomes das classes do modelo *(names)*. O tamanho padrão das imagens de entrada, podem variar de acordo com a versão do YOLO, por exemplo, o YOLOv8 utiliza como padrão 640x640 *pixels* de altura e largura respectivamente. A estrutura do arquivo YALM é explicada a seguir.

train: <*path-to-train-data*>
val: <*path-to-validation-data*>
nc: <*number-of-classes*>
names: <*path-to-class-names-file*>

2.6 Métricas de Avaliação

Ao longo da última década, através das várias versões do algoritmo YOLO, observamos a utilização de diversas técnicas para tornar eficiente o processo de detecção e classificação de objetos em imagens digitais. Essas abordagens variadas empregaram, ao longo dos anos, métodos distintos para avaliar a eficácia do algoritmo na identificação de objetos em imagens. Na presente subseção, destacamos algumas das métricas comumente utilizadas para essa avaliação.

2.6.1 Matriz de Confusão

A matriz de confusão é uma técnica utilizada para avaliar a eficiência de classificadores em algoritmos inteligentes. Ela permite analisar a qualidade de uma classificação, comparando os resultados com as expectativas, e identificar possíveis melhorias no algoritmo ou na análise dos dados(JÚNIOR et al., 2022).

Imagine a matriz de confusão como um guia que nos auxilia a compreender o desempenho de um algoritmo de classificação. Essa matriz divide as classificações em quatro grupos: verdadeiros positivos (quando algo é corretamente identificado como positivo), verdadeiros negativos (quando algo é corretamente identificado como negativo), falsos positivos (quando algo é erroneamente identificado como positivo) e falsos negativos (quando algo é erroneamente identificado como negativo), como está exemplificado na Figura 11.

Figura 11 – Matriz de confusão.

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

Fonte: (RODRIGUES, 2019).

Esses grupos são como peças-chave que utilizamos para calcular diversas métricas. Por exemplo, em estudos que envolvem a classificação de animais, a matriz de confusão pode ser empregada para avaliar quão efetivos são os algoritmos de IA na identificação de animais específicos, como gatos e cachorros. Essa abordagem auxilia a identificar quais atividades são mais desafiadoras de serem reconhecidas, orientando ajustes nos algoritmos para aprimorar a precisão e a eficácia. Entender a matriz de confusão de forma abrangente é fundamental para avaliar a eficácia do modelo em variados contextos. A diagonal principal da matriz concentra os verdadeiros positivos, ou seja, as previsões corretas realizadas pelo modelo. Quando as linhas representam os valores reais e as colunas representam as previsões feitas pelo modelo, qualquer desvio dessa diagonal indica classificações incorretas. Os elementos fora da diagonal principal indicam onde ocorrem as classificações equivocadas, apontando possíveis falhas do modelo em suas previsões e áreas para refinamento e aprimoramento.

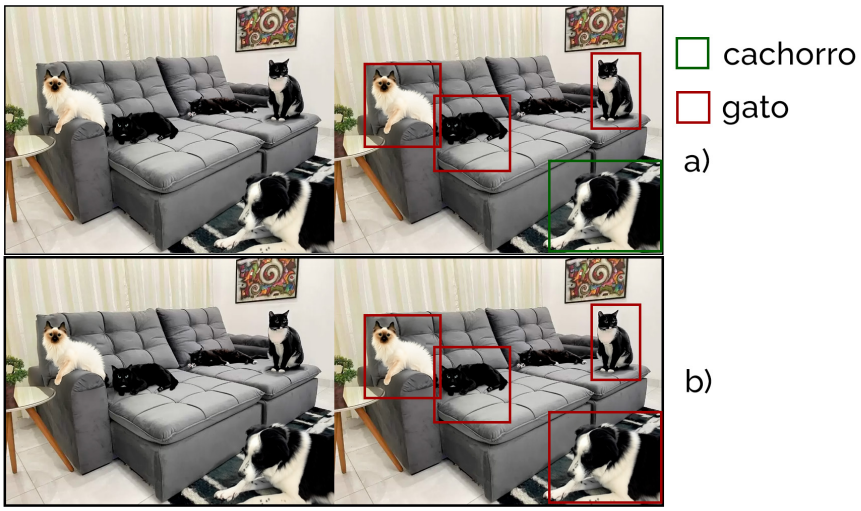
2.6.2 Precision (Precisão)

Em um contexto de classificação, a precisão é uma das notas de avaliação mais comuns (SANTOS; BEKO; LEITHARDT, 2022). A precisão em aprendizado de máquina para modelos de detecção e classificação de objetos em imagens refere-se à medida de acertos do modelo frente à quantidade de objetos identificados em uma imagem. Ela é calculada como a razão entre o número de objetos identificados corretamente e o total de objetos apontados pelo algoritmo na imagem. Em outras palavras, a precisão mede a proporção de acertos em relação ao total de tentativas do modelo. A fórmula de precisão é definida da seguinte forma:

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Positivos}}$$

A avaliação da performance de um modelo de classificação de imagens, como aquele que diferencia gatos de cachorros, é essencial para compreender sua eficácia. Um dos principais indicadores usados nessa avaliação é a precisão, que representa a proporção de previsões corretas em relação ao total de imagens analisadas. Por exemplo, se um modelo identifica corretamente três gatos e um cachorro em uma imagem que contém quatro gatos e um cachorro, sua precisão seria de 100%, indicando que acertou 100% das classificações, exemplificado na Figura 12a. No entanto, se esse mesmo modelo identificar três gatos e um cachorro como gato em uma imagem similar, sua precisão cairia para 80%, pois falhou em identificar o cachorro corretamente, apesar de ter acertado a maioria das classificações, como podemos ver na Figura 12b.

Figura 12 – Exemplo de Funcionamento da Precisão.



Fonte: Elaborado pelo Autor.

2.6.3 Recall (Sensibilidade)

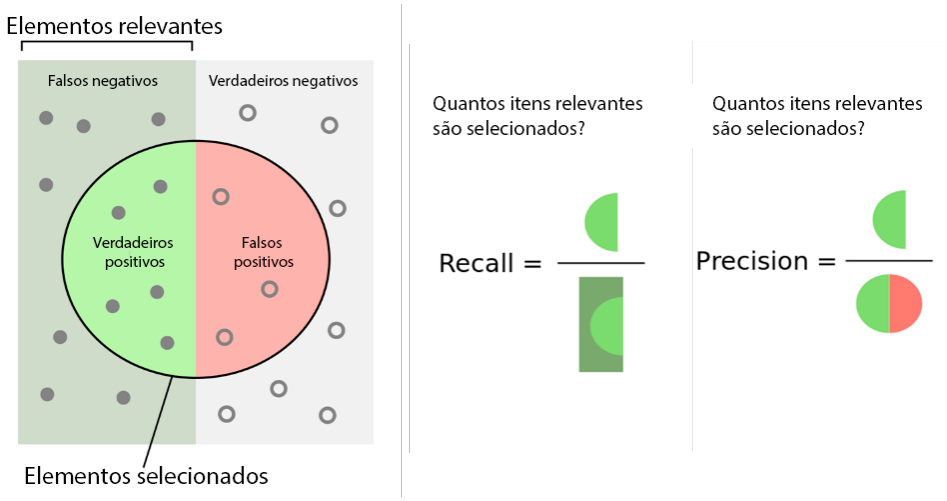
O *recall* é uma métrica utilizada no aprendizado de máquina para avaliar a capacidade de um modelo de identificar corretamente todos os exemplos positivos em um conjunto de dados. Ele é calculado como a razão entre o número de exemplos positivos corretamente identificados pelo modelo e o número total de exemplos positivos no conjunto de dados.(CUNHA; CAMARGO, 2019). O calculo é da seguinte forma:

$$\text{Recall} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Negativos}}$$

A distinção entre precisão e *recall* pode ser compreendida com uma analogia simples. Na precisão, você avalia quantas das suas identificações estão corretas, em relação ao total de identificações que você fez. Por exemplo, se você rotulou 20 fotos como "contendo gatos", mas apenas 15 delas realmente tinham gatos, e havia um total de 30 fotos, sua precisão seria de 75%. Isso indica que 75% das suas identificações foram precisas. Já o *recall* mede quantas das instâncias reais de uma classe você conseguiu identificar corretamente, em relação ao total de

instâncias dessa classe. Se havia 30 fotos com gatos, mas você só identificou corretamente 15 delas, seu *recall* seria de 50%, mostrando que você só "lembrou"metade das fotos com gatos. Resumindo, a precisão se concentra na qualidade das identificações, enquanto o *recall* foca na capacidade de identificar todas as instâncias relevantes de uma classe. Na Figura 13 podemos ver uma comparação ilustrada entre *recall* e precisão.

Figura 13 – Comparação entre *recall* e precisão.



Fonte: (Wikipédia, 2024).

2.6.4 F1-score

A pontuação F1 é uma medida da precisão de um modelo, equilibrando precisão e *recall*, e é comumente usada na avaliação de aprendizado de máquina. É a média harmônica de precisão e *recall*, com valor entre 0 e 1, onde 1 é a melhor pontuação F1 possível(NAYEEM; RANA; ISLAM, 2022). A pontuação F1 é calculada usando a seguinte fórmula:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.6.5 Average Precision (AP)

A AP (Precisão Média) é uma métrica fundamental em problemas de detecção de objetos. Ela é calculada a partir da curva de Precisão-*recall* para uma classe específica. A curva de Precisão-*recall* representa a *trade-off* entre a precisão do modelo (taxa de verdadeiros positivos entre todas as predições positivas) e o *recall* (taxa de verdadeiros positivos entre todas as instâncias reais positivas).

O cálculo da AP envolve a área sob a curva de Precisão-*recall*. Quanto maior a área sob a curva, maior a AP, indicando um desempenho mais consistente e preciso do modelo na detecção de objetos daquela classe específica.

A AP é especialmente relevante em problemas de detecção de objetos com múltiplas classes, pois o mAP é frequentemente utilizado para avaliar o desempenho global do modelo, considerando a média das APs de todas as classes.

Essa métrica é valiosa para compreender não apenas a precisão geral do modelo, mas também sua habilidade em recuperar instâncias específicas de objetos, equilibrando precisão e *recall*.

2.6.6 Intersection Over Union (IoU)

A IoU é frequentemente usada para avaliar a precisão do detector em tarefas de detecção de objetos. Ele mede a sobreposição entre as caixas delimitadoras previstas e reais e é calculada como a razão entre a área de sobreposição e a área combinada entre essas caixas delimitadoras previstas e reais.

As métricas de IoU desempenham um papel importante na determinação da precisão das detecções, classificando-as como verdadeiros positivos ou falsos positivos. A detecção de valores de IoU acima do limite definido é considerada correta e 0,5 é normalmente usado como valor padrão. Quanto maior o limite, maior será a sobreposição entre as caixas delimitadoras para verificar se a detecção está correta.

Esta métrica é essencial para problemas de detecção de objetos e permite avaliar a precisão espacial da detecção em relação à localização real do objeto. Essas métricas são frequentemente usadas para analisar e comparar o desempenho de modelos de detecção de objetos como o YOLO, permitindo avaliação quantitativa de sua precisão e capacidade de detecção. A fórmula para o cálculo da IoU é dada por:

$$IoU = \frac{\text{Área da Interseção}}{\text{Área da União}}$$

Aqui está como cada termo é calculado:

- **Área da Interseção:** A área comum entre a caixa delimitadora prevista pelo modelo e a caixa delimitadora verdadeira (ground truth).
- **Área da União:** A soma das áreas das caixas delimitadoras prevista e verdadeira, menos a área da interseção.

O valor resultante do cálculo IoU varia de 0 a 1. Quanto mais próximo estiver de 1, melhor será a sobreposição entre a detecção do modelo e a posição real do objeto. Se o IoU estiver próximo de 0, há pouca sobreposição e o reconhecimento do modelo não é considerado preciso (REZATOFIGHI et al., 2019). Por exemplo:

- $IoU \geq 0.5$, geralmente é considerado uma detecção válida.
- Se $IoU \geq 0.75$, é frequentemente considerado uma detecção de alta qualidade.

O IoU é uma métrica essencial na avaliação da precisão das detecções de objetos em visão computacional. Contudo, várias variações dessa métrica foram desenvolvidas para aprimorar sua

aplicabilidade e precisão. Entre essas variações, notáveis são o CIoU, o GIoU e o SIoU, os quais não apenas consideram a sobreposição entre as caixas delimitadoras, mas também outros fatores como a distância entre os centros das caixas e a proporção entre largura e altura. Essas variações têm sido incorporadas ao algoritmo YOLO, possibilitando a geração de detecções mais precisas e robustas. Isso tem contribuído significativamente para o sucesso do YOLO em uma ampla variedade de aplicações, desde a detecção de objetos em vídeos de vigilância até a condução autônoma. A seguir, iremos explorar mais detalhadamente essas variações

O CIoU é uma métrica de avaliação que busca aprimorar a precisão da tradicional IoU, considerando não apenas a sobreposição entre as caixas delimitadoras, mas também outros aspectos como a distância entre os centros das caixas e a diferença em termos de aspecto (proporção entre largura e altura). Em sua formulação técnica, o CIoU é calculado através da IoU, seguida pela inclusão de termos adicionais: o *Completeness Term*, que penaliza a diferença entre as áreas das caixas delimitadoras e normaliza a IoU; o *Centroid Distance Term*, que penaliza a distância entre os centros das caixas, incentivando a predição de caixas com centros mais próximos aos centros verdadeiros; e o *Aspect Ratio Term*, que penaliza a diferença entre as proporções das caixas delimitadoras, corrigindo distorções na sua forma. Esses elementos combinados fornecem uma métrica mais abrangente e precisa para a avaliação da qualidade das detecções de objetos em tarefas de visão computacional (WANG; SONG, 2021).

O cálculo do CIoU envolve várias etapas e termos. Aqui está a fórmula completa para calcular o CIoU:

Seja A a caixa delimitadora *(bounding box)* predita e B a caixa delimitadora verdadeira *(ground truth)*.

IoU:

$$IoU = \frac{Area(A \cap B)}{Area(A \cup B)}$$

Completeness Term:

$$v = \frac{4}{\pi^2} \cdot (\arctan(\frac{w_B}{h_B}) - \arctan(\frac{w_A}{h_A}))^2$$

Centroid Distance Term:

$$c = \frac{(x_A - x_B)^2 + (y_A - y_B)^2}{w_B^2 + h_B^2}$$

Aspect Ratio Term:

$$a = \frac{v}{(1 - IoU) + v}$$

CIoU:

$$CIoU = IoU - \frac{c}{a} - v$$

Onde: w_A, h_A são a largura e a altura da caixa delimitadora predita A. w_B, h_B são a largura e a altura da caixa delimitadora verdadeira B. x_A, y_A são as coordenadas do centro da caixa delimitadora predita A. x_B, y_B são as coordenadas do centro da caixa delimitadora verdadeira B.

O CIoU é então calculado subtraindo o termo de completude normalizado pela razão de aspecto e o termo de distância entre centros do IoU. Este cálculo visa fornecer uma métrica mais abrangente e precisa para avaliar a qualidade das detecções de objetos em tarefas de visão computacional.

A função de perda SIoU é uma medida utilizada em tarefas de detecção de objetos para aprimorar a precisão e a eficiência do treinamento. Segundo Gevorgyan (2022), ela considera o ângulo do vetor entre a regressão desejada e é composta por quatro funções de custo: Custo do ângulo, Custo da distância, Custo da forma e Custo da IoU. O SIoU destaca-se em situações onde a precisão da localização do objeto é crucial, pois não apenas avalia a sobreposição com precisão, mas também considera a localização espacial precisa do objeto. A integração do SIoU em algoritmos de detecção de objetos, como o YOLO, pode resultar em detecções mais precisas e confiáveis, contribuindo para melhorar o desempenho global do sistema em diversas aplicações, desde vigilância por vídeo até condução autônoma.

A fórmula para calcular o SIoU é composta por quatro termos principais: o termo do ângulo, o termo da distância, o termo da forma e o termo da IoU. Aqui está a fórmula completa:

$$SIoU = IoU - Termo\ do\ Ângulo - Termo\ da\ Distância - Termo\ da\ Forma$$

No cálculo do SIoU, diversas etapas são consideradas para avaliar a precisão das detecções de objetos em tarefas de visão computacional. Primeiramente, a IoU, uma medida fundamental, é calculada para determinar a sobreposição entre as caixas delimitadoras prevista e verdadeira. Além disso, o cálculo incorpora outros fatores cruciais, como o ângulo entre os vetores de regressão, que é utilizado para estimar a orientação da detecção. A distância entre os centros das caixas delimitadoras também é levada em conta, permitindo uma avaliação mais precisa da localização espacial dos objetos. Adicionalmente, a diferença na forma das caixas delimitadoras, considerando a relação de aspecto de cada caixa, é cuidadosamente analisada. Essas etapas combinadas resultam em uma métrica mais abrangente e precisa, proporcionando *insights* valiosos sobre a qualidade das detecções de objetos, e contribuindo para o desenvolvimento de sistemas de visão computacional mais eficientes e confiáveis.

GIoU é uma métrica e uma função de perda usada em tarefas de detecção de objetos. É uma extensão da métrica IoU, que é comumente usada em *benchmarks* de detecção de objetos.GIoU pode ser usado como uma perda de regressão para caixas delimitadoras 2D alinhadas ao eixo e também pode ser usado como uma nova métrica para comparar duas formas convexas arbitrárias. O GIoU aborda a fraqueza do IoU introduzindo uma versão generalizada que pode ser usada tanto como uma nova perda quanto como uma nova métrica. Foi demonstrado que o GIoU melhora o desempenho de estruturas de detecção de objetos de última geração em *benchmarks* populares de detecção de objetos, como PASCAL VOC e MS COCO (REZATOFIGHI et al., 2019). Esta métrica não apenas avalia a sobreposição entre as caixas delimitadoras prevista e verdadeira, mas também considera a diferença entre a área da união das caixas e a área da menor caixa que as envolve. Essa abordagem leva em conta não apenas a precisão da localização do objeto, mas também a precisão de sua escala, tornando o GIoU especialmente útil para objetos

de diferentes tamanhos e formas. Assim, o GIoU proporciona uma métrica mais abrangente e robusta para a avaliação da qualidade das detecções de objetos em diversas aplicações de visão computacional.

A fórmula do GIoU é calculada da seguinte forma:

$$GIoU = IoU - \left(\frac{C - U}{C} \right)$$

A fórmula do GIoU incorpora uma correção à medida tradicional de IoU, levando em consideração a diferença entre a área da união das caixas e a área da menor caixa que as envolve. Nessa fórmula, IoU representa a medida tradicional de sobreposição entre as caixas delimitadoras prevista e verdadeira, calculada como a proporção entre a área da interseção e a área da união das caixas. C é a área da menor caixa que envolve as caixas delimitadoras prevista e verdadeira, enquanto U representa a área da união das caixas delimitadoras. Essa abordagem resulta em uma métrica mais abrangente que considera não apenas a sobreposição, mas também a escala e a localização espacial dos objetos detectados.

2.6.7 Mean Average Precision (mAP)

A mAP é uma métrica comumente usada para avaliar o desempenho de sistemas de recuperação de informações e detecção de objetos. É calculado como a média das APs para cada consulta ou classe. No contexto da detecção de objetos, o mAP é usado para medir a precisão de um modelo na detecção e localização de objetos em uma imagem. Um valor mAP mais alto indica melhor desempenho (LEE; JEONG; PETER, 2022). O cálculo é expresso pela fórmula:

$$\text{mAP} = \frac{\text{Soma das APs de todas as classes}}{\text{Número total de classes}}$$

De acordo com Deng (2013) as mAP50 e mAP50-95 são métricas de avaliação usadas em tarefas de detecção de objetos. A mAP50 mede a mAP em uma IoU de 0,5, avaliando detecções consideradas mais fáceis. Já a mAP50-95 mede a precisão média média em limites de IoU que variam de 0,5 a 0,95, o que reflete o desempenho do modelo em uma gama mais ampla de dificuldades de detecção. A fórmula do calculo da mAP50 e mAP50-95 são expressas por:

$$\text{mAP50} = \frac{\text{Soma das APs dos pontos onde a IoU é de 0,5}}{\text{Número total de classes}}$$

$$\text{mAP50-95} = \frac{\text{Soma das APs dos pontos onde a IoU está entre 0,5 e 0,95}}{\text{Número total de classes}}$$

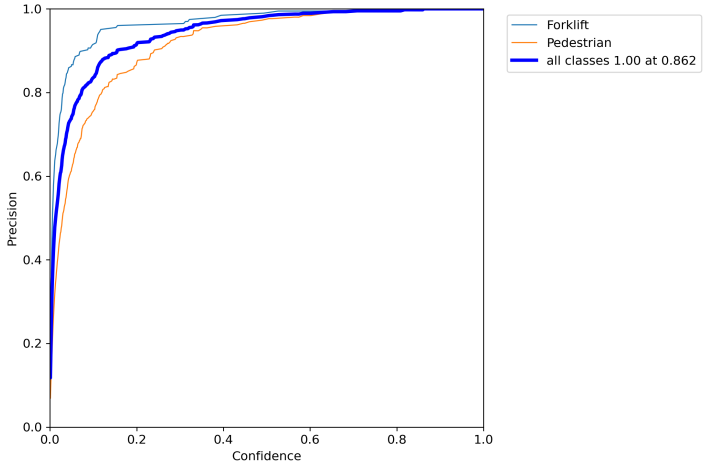
Essa métrica é particularmente útil ao lidar com conjuntos de dados que contenham objetos distribuídos de forma desigual entre diferentes categorias, permitindo uma avaliação igualitária do modelo em todo o espectro de classes.

2.6.8 Gráfico de Precisão-Confiança.

A precisão é a proporção de previsões corretas em relação ao total de previsões, enquanto a confiança é a probabilidade de que uma previsão esteja correta. O gráfico de precisão-confiança permite identificar se o modelo está superestimando ou subestimando a precisão, e se está subestimando ou superestimando a confiança. Dessa forma, é possível ajustar o modelo para melhorar seu desempenho.(STRAUSS; JÚNIOR; FERREIRA, 2023)

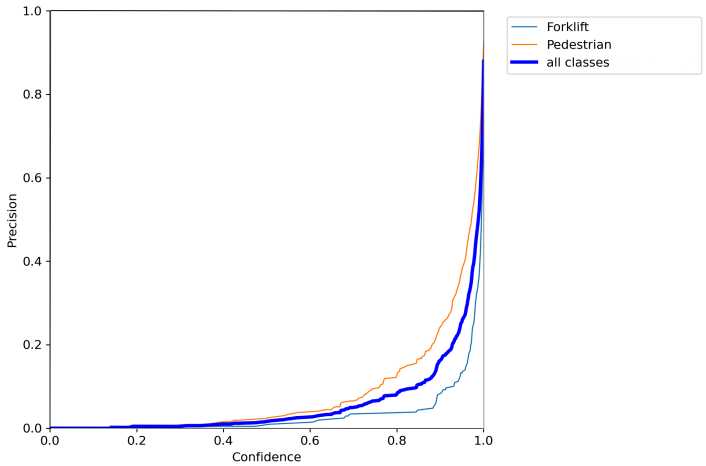
Na Figuras 14 e 15podemos ver um exemplo de um gráfico de *recall* bom e outro ruim.

Figura 14 – Exemplo de Grafico de Precisão-Confiança Bom.



Fonte: Elaborado pelo Autor.

Figura 15 – Exemplo de Grafico de Precisão-Confiança Ruim.



Fonte: Elaborado pelo Autor.

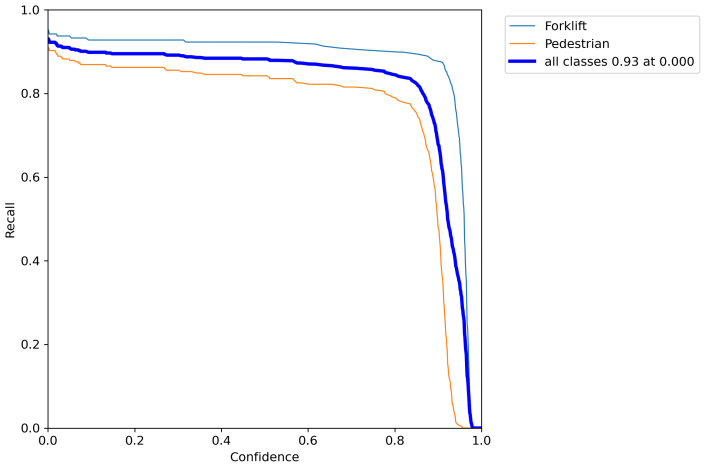
Ao analisar gráficos de precisão e confiabilidade de modelos de aprendizado de máquina para tarefas de classificação binária, é importante compreender os eixos. O eixo de confiança representa a probabilidade associada a uma previsão positiva, e o eixo de precisão representa a proporção de verdadeiros positivos entre todos os casos previstos como positivos. Para alcançar o desempenho ideal, é importante identificar pontos na curva que sejam confiáveis e precisos.

2.6.9 Gráfico de Recall-Confiança.

O gráfico de *Recall*-confiança é uma ferramenta utilizada na análise de desempenho de modelos de classificação. Ele representa a relação entre a taxa de *recall* (ou sensibilidade) e a confiança do modelo em diferentes pontos de corte de probabilidade. O eixo x representa a confiança do modelo, enquanto o eixo y representa a taxa de *recall*. A interpretação do gráfico é importante para avaliar o desempenho do modelo em diferentes cenários e escolher o ponto de corte que melhor atende às necessidades do problema em questão.(ANDRADE et al., 2021)

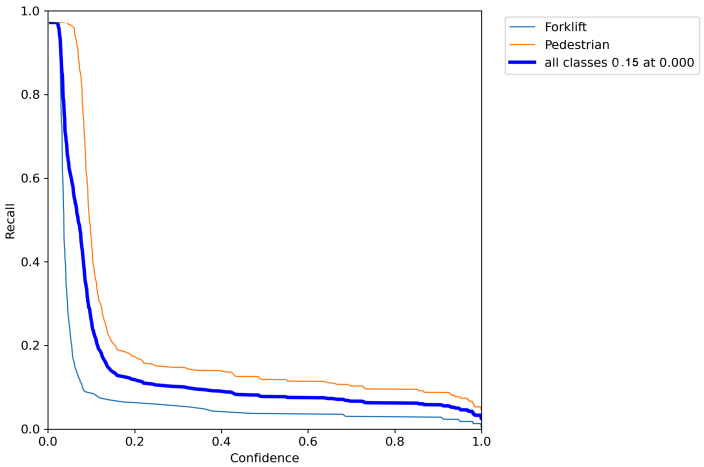
Na Figuras 16 e 17 podemos ver um exemplo de um gráfico de *recall* bom e outro ruim.

Figura 16 – Exemplo de Gráfico de Recall-Confiança Bom.



Fonte: Elaborado pelo Autor.

Figura 17 – Exemplo de Gráfico de Recall-Confiança Ruim.



Fonte: Elaborado pelo Autor.

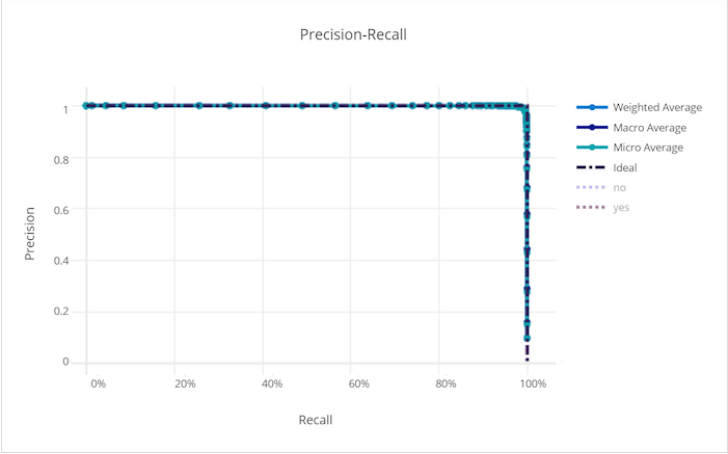
Os gráficos de *recall*-confiança desempenham um papel importante na avaliação de modelos de classificação. A dimensão horizontal representa a confiança na previsão positiva e a dimensão vertical representa a sensibilidade do modelo na identificação de casos positivos. A curva ascendente próxima ao canto superior direito indica bom desempenho, indicando que o

modelo mantém alta precisão mesmo ao recuperar a maioria das instâncias positivas. Encontrar o ponto ideal na curva requer um equilíbrio entre confiança e uma boa memória.

2.6.10 Gráfico de Precisão-Recall.

A precisão e o *recall* são métricas comuns para avaliar a qualidade de um modelo de classificação. A precisão mede a proporção de exemplos classificados como positivos que são realmente positivos, enquanto o *recall* mede a proporção de exemplos positivos que foram corretamente classificados. Um gráfico de precisão e *recall* mostra como a precisão e o *recall* variam para diferentes limites de decisão. Isso pode ajudar a escolher um limite que atenda aos requisitos específicos do problema, por exemplo, maximizando a precisão ou o *recall*, dependendo das necessidades do aplicativo. O eixo x do gráfico representa o *recall*, o eixo y representa a precisão e cada ponto no gráfico representa um limite de decisão diferente. Em geral, quanto mais próximo o gráfico estiver do canto superior direito, melhor será o desempenho do modelo. Este tipo de gráfico é especialmente útil quando as classes estão desbalanceadas, ou seja, quando há muito mais exemplos de uma classe do que da outra.(STRAUSS; JÚNIOR; FERREIRA, 2023). Nas Figuras 18 e 19 podemos ver exemplos de gráfico bom e ruim.

Figura 18 – Exemplo de Gráfico de Precisão-Recall Bom.



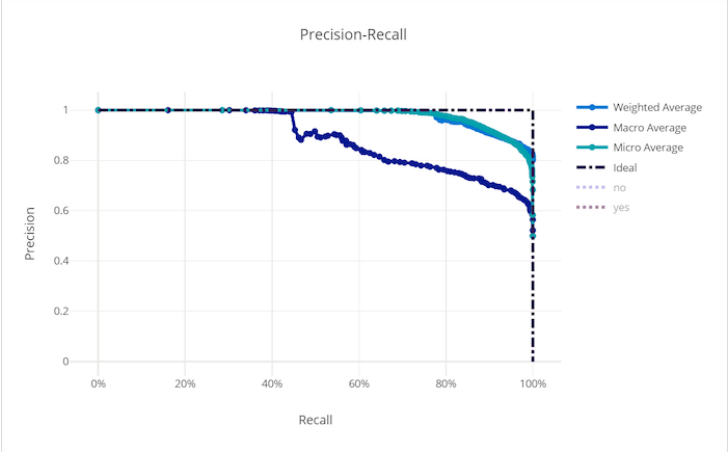
Fonte: (MICROSOFT, 2023).

2.6.11 Gráfico de F1-Confiança.

No gráfico de *F1-score* e confiança em machine learning para tarefas de classificação, é fundamental compreender os eixos do gráfico. O eixo horizontal, geralmente representando a confiança ou probabilidade associada às previsões positivas do modelo, indica o nível de certeza do modelo ao fazer uma previsão positiva. O eixo vertical, que exibe o *F1-score*, representa uma métrica que combina precisão e *recall*, fornecendo uma medida balanceada do desempenho do modelo. Na Figura 20 vemos um exemplo do gráfico.

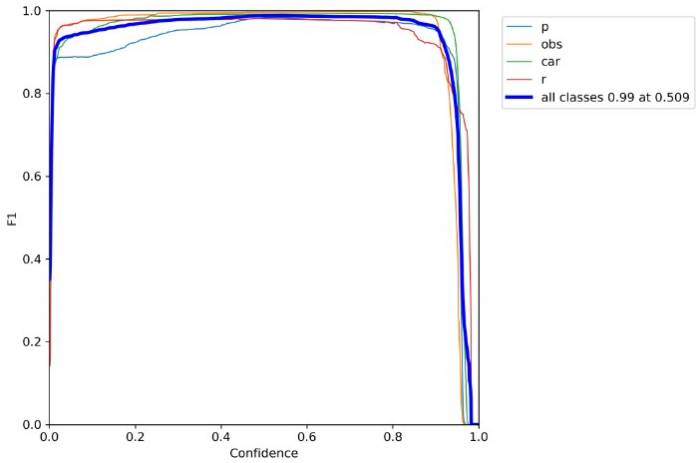
Ao interpretar gráficos, o objetivo é ter altas pontuações de confiança e F1, com uma relação efetiva entre a capacidade do modelo de fazer uma previsão correta (precisão) e sua

Figura 19 – Exemplo de Gráfico de Precisão-*recall* Ruim.



Fonte: (MICROSOFT, 2023).

Figura 20 – Exemplo de Gráfico F1-Confiança.



Fonte: (NAANAA, 2023).

capacidade de capturar todos os casos positivos (*recall*). ponto que mostra equilíbrio. . Identificar esse ponto ideal. A análise dos eixos auxilia a entender como o ajuste do limite de confiança afeta diretamente o equilíbrio entre precisão e recuperação no contexto da classificação binária.

2.7 *Roboflow.*

O *Roboflow* é uma plataforma baseada na web que visa simplificar o desenvolvimento e a implementação de modelos de visão computacional. Ela oferece um conjunto abrangente de ferramentas para preparar dados, treinar modelos e implantá-los em diversas plataformas. O foco principal da plataforma é em tarefas de detecção de objetos e classificação de imagens. Uma das principais funcionalidades do *Roboflow* é a preparação de dados. Ele fornece recursos para a anotação de imagens, criação de conjuntos de dados balanceados e manipulação de imagens, garantindo que os dados utilizados para o treinamento de modelos sejam de alta qualidade. Isso facilita o processo de treinamento e melhora a precisão dos modelos desenvolvidos. Além disso, a plataforma permite o treinamento de modelos utilizando *frameworks* e bibliotecas populares,

como *TensorFlow* e *PyTorch*. Após o treinamento, os modelos podem ser facilmente implantados em diversas plataformas, incluindo dispositivos de borda, servidores em nuvem e aplicativos móveis, o que amplia suas possibilidades de aplicação. O *Roboflow* também oferece integrações com outras ferramentas e serviços, o que permite a incorporação de visão computacional em projetos mais amplos e variados. Com essas características, a plataforma se torna uma solução acessível e eficiente para desenvolvedores e empresas que buscam aplicar a visão computacional em suas soluções(Roboflow, 2024).[1]

2.8 *Dataset COCO*

O conjunto de dados COCO, é utilizado para tarefas de visão computacional, como detecção de objetos, segmentação e legendagem. Com mais de 330.000 imagens anotadas com caixas delimitadoras, máscaras de segmentação, pontos-chave e legendas, o COCO cobre 80 categorias de objetos e é amplamente usado para treinar e avaliar modelos como YOLO e Faster R-CNN. Suas anotações detalhadas e a diversidade de cenários, incluindo variações e oclusões, fazem do COCO um desafio significativo e uma referência padrão. O conjunto de dados está disponível publicamente e é atualizado regularmente, com versões como COCO 2017 e conjuntos relacionados como COCO-*Stuff* (LIN et al., 2015).[2]

2.9 *Google Colab*

O *Google Colab* é uma plataforma gratuita baseada em nuvem fornecida pelo Google, permitindo a escrita e execução de código *Python* em notebooks *Jupyter*. Entre suas principais características estão o acesso gratuito, tornando-o ideal para estudantes e pesquisadores, e a possibilidade de usar GPUs e *Tensor Processing Unit* (TPU)s para aceleração, o que é útil para aprendizado profundo. Por ser baseado na nuvem, elimina a necessidade de configuração de ambiente local e gerenciamento de hardware. O *Google Colab* também se integra facilmente ao *Google Drive*, permitindo acesso e salvamento direto de arquivos. A colaboração em tempo real, similar ao *Google Docs*, é uma vantagem significativa, permitindo que vários usuários trabalhem no mesmo notebook simultaneamente. Entre os benefícios do *Google Colab* estão a facilidade de uso, com notebooks que são simples de configurar, e a disponibilidade de recursos gratuitos de GPU e TPU, que são atraentes para projetos exigentes. Muitas bibliotecas populares, como *TensorFlow* e *PyTorch*, vêm pré-instaladas, economizando tempo na configuração. O ambiente interativo permite a visualização imediata dos resultados, facilitando a análise de dados e a depuração.[3]

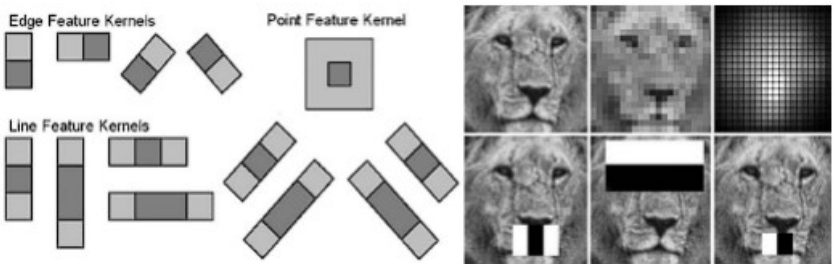
[1] <https://roboflow.com/>
[2] <https://cocodataset.org/>
[3] <https://colab.research.google.com/>

3 TRABALHOS RELACIONADOS

Na área de observação de animais, a visão computacional é fundamental para automatizar uma variedade de tarefas, tais como o rastreamento de movimentos, a identificação de padrões comportamentais e a análise de características morfológicas. As pesquisas de Burghardt e Calic (2006) e Kumar e Singh (2014) demostram um pouco sobre isso.

No estudo conduzido por Burghardt e Calic (2006), são apresentados os métodos usados na extração de informações em tempo real acerca das atividades motoras de animais selvagens por meio da detecção e monitoramento de suas faces. O procedimento adotado envolve a adaptação de um algoritmo derivado do método de detecção de faces humanas, fundamentado em características *Haar*, que funcionam como pequenos modelos que são aplicados a uma imagem para buscar padrões específicos. Esses modelos podem representar bordas, cantos ou mudanças abruptas na intensidade dos *pixels*. Elas são amplamente utilizadas em tarefas de detecção de objetos e reconhecimento facial (OREN et al., 1997), e do algoritmo de classificação *AdaBoost*, onde ele combina vários classificadores fracos para criar um classificador forte. Cada classificador fraco é treinado em uma versão modificada do conjunto de dados, onde o foco é corrigir os erros cometidos pelos classificadores anteriores (FREUND; SCHAPIRE et al., 1996). O algoritmo foi treinado utilizando imagens de leões selvagens como exemplo prático, conforme ilustrado na Figura 21. É relevante destacar que o foco do algoritmo não se concentra no reconhecimento individual dos leões, mas sim na detecção de sua presença nos vídeos analisados.

Figura 21 – Aplicação de características Haar em faces de leões.



Fonte: (BURGHARDT; CALIC, 2006).

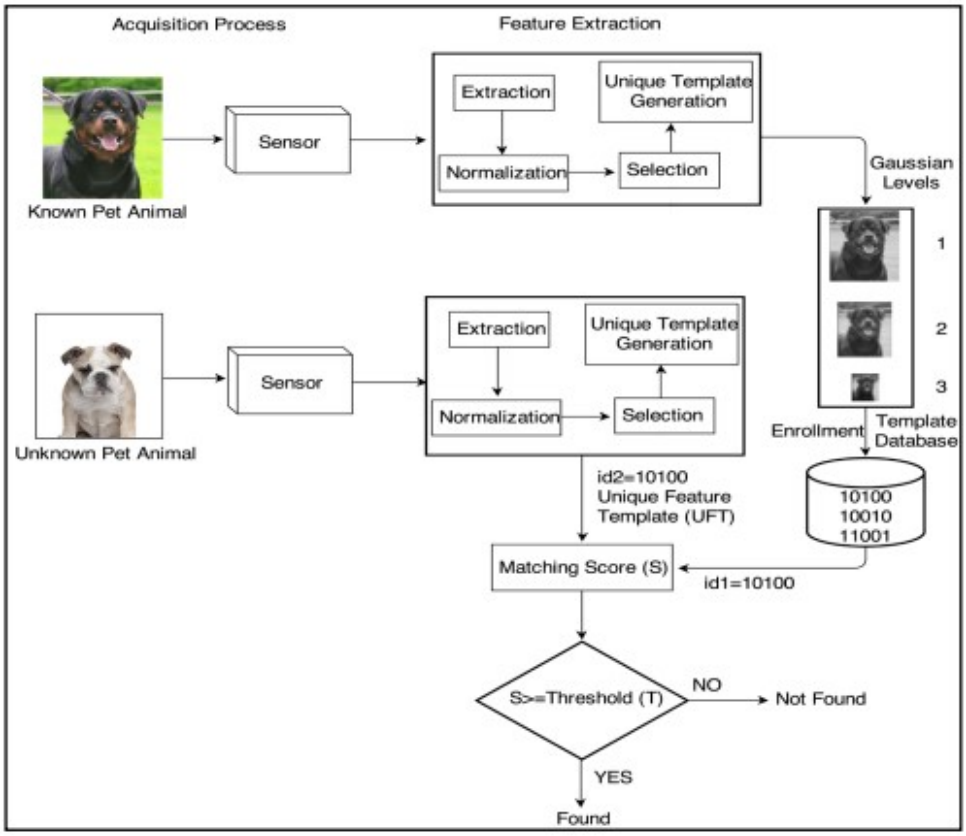
Na pesquisa conduzida por Kumar e Singh (2014), explora-se a implementação de algoritmos de reconhecimento biométrico facial em cães, com o objetivo de superar as dificuldades associadas à identificação de animais domésticos por meio de métodos convencionais, como características de pelagem ou outras marcações corporais. O estudo emprega algoritmos baseados em técnicas como Análise de Componentes Principais (PCA), Análise Discriminante Local (LDA) e Análise de Componentes Independentes (ICA), juntamente com suas variações, para extrair características relevantes dos dados. A PCA é utilizada para reduzir a dimensionalidade dos dados, transformando variáveis correlacionadas em componentes principais não correlacionados, capturando a maior parte da variação nos dados originais. Por sua vez, a LDA é empregada com foco na maximização da separação entre diferentes classes de dados, sendo especialmente útil em tarefas de classificação. Já a ICA busca desembaraçar sinais observados em suas fontes originais

independentes, útil em aplicações onde é crucial identificar padrões independentes entre os dados, como na separação de fontes em sinais de áudio ou imagens. Essas técnicas combinadas permitem uma extração eficiente de características relevantes para análise e modelagem dos dados.

A metodologia proposta na pesquisa alcança uma notável taxa de identificação correta de 94.86%. No entanto, são ressaltados desafios enfrentados pelo algoritmo, especialmente relacionados à variação na iluminação decorrente das condições climáticas. Além disso, destaca-se a complexidade de lidar com cães, que se mostram desafiadores para a aplicação de biometria. A pesquisa sugere que futuros trabalhos devem direcionar sua atenção para o desenvolvimento de algoritmos que não dependam da pose e expressão do animal.

A Figura 22 ilustra de maneira visual o funcionamento do algoritmo concebido pelos autores, oferecendo uma representação gráfica do processo de reconhecimento facial em cães. Essa abordagem inovadora na identificação de animais domésticos destaca a importância de considerar não apenas a eficácia do algoritmo, mas também os desafios práticos associados à natureza dos sujeitos biometrizados.

Figura 22 – Diagrama de blocos para o algoritmo de reconhecimento facial para cachorros.



Fonte: (KUMAR; SINGH, 2014).

4 METODOLOGIA

Este artigo propõe o desenvolvimento de um sistema de monitoramento simples para desktop, com foco na segurança e bem-estar de animais de estimação. Muitos cuidadores de animais enfrentam a preocupação constante de perder seus animais quando eles saem da residência ou se aventuram em áreas potencialmente perigosas. Esse sentimento representa uma das principais motivações para a realização deste sistema.

O sistema proposto oferece a possibilidade de monitorar as atividades dos animais de estimação, e identificar cada animal individualmente. Um modelo customizado baseado na arquitetura YOLOv8 é usado para obter essa funcionalidade. O objetivo desta abordagem inovadora é fornecer uma solução eficaz para reduzir os riscos associados à perda de animais de estimação, promovendo assim tranquilidade e segurança aos cuidadores.

Além disso, o sistema integrará funcionalidades avançadas, como o monitoramento de áreas restritas para os *pets*. Ao detectar a ultrapassagem dessas áreas delimitadas, o sistema acionará um alarme sonoro, oferecendo uma camada adicional de proteção e controle para os cuidadores dos animais.

Para promover a transparência e fornecer informações detalhadas sobre as interações dos *pets*, o sistema incorporou um recurso de *logs*. Isto permite o registo sistemático de atividades, identificação de animais presentes e incidentes de ultrapassagem de áreas restritas. Esses registros fornecem aos tratadores um registro completo e detalhado, facilitando um manejo mais eficaz e uma compreensão completa do comportamento de seus animais de estimação.

4.1 Base de Dados

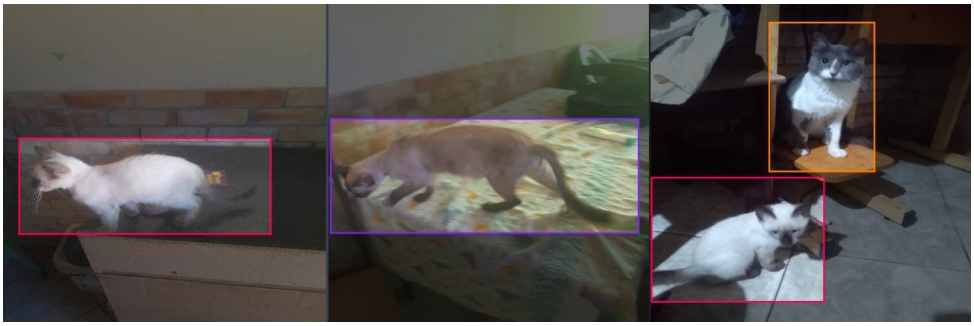
Nesse projeto, foi utilizado um *dataset* criado do zero *(from scratch)*, criado utilizando o *Roboflow*. Esta plataforma disponibiliza diversas ferramentas, como rotulagem de imagens, que são importantes para a conversão de imagens em modelos de visão computacional com treinamento *from scratch* que pode ser integrado em aplicações.

O conjunto de dados incluí três categorias de animais de estimação, especificamente os gatos do autor, que são chamados com nomes de Cleiton, Princesa e Tonzinha. Foram tiradas 3800 imagens, distribuídas proporcionalmente entre as três classes. Técnicas como *crop* de imagens usando a ferramenta de corte, geraram novas instâncias e rotation, usando a metodologia de rotação de fotos, foram aplicadas para enriquecer o conjunto de dados. Esse processo ampliou o acervo para um total de 9880 imagens rotuladas, das quais 9120 foram designadas para treinamento e 760 para validação. Podemos ver exemplos da rotulagem de imagens na Figura 23.

4.2 Treinamento

O conjunto de dados rotulado foi submetido a dois tipos de treinamento distintos. No primeiro método, o treinamento foi realizado completamente do zero, utilizando a arquitetura *YOLOv8n* e seguindo uma rotina de treinamento de 50 épocas. A escolha da variante YOLOv8n

Figura 23 – Exemplos de rotulação do *dataset* no Roboflow



Fonte: Elaborado pelo Autor.

foi motivada pela sua eficiência em termos de processamento, proporcionando uma velocidade de detecção de objetos aceitável. Devido às limitações da assinatura gratuita do *Colab*, o modelo não foi treinado até o *"early stopping"*, que é um mecanismo acionado quando o desempenho do modelo em um conjunto de validação deixa de melhorar, prevenindo o *overfitting* durante o treinamento. No entanto, os resultados foram satisfatórios para mostrar as diferenças entre os modelos.

Em paralelo, foi conduzido um segundo tipo de treinamento utilizando a técnica de *transfer learning*, com o *dataset* COCO. Nesse caso, também foi adotada uma rotina de treinamento de 50 épocas, e o *"early stopping"* não foi aplicado devido às mesmas limitações. Este método aproveita o conhecimento prévio de modelos treinados em domínios relacionados para melhorar o desempenho no domínio-alvo.

Após ambas as sessões de treinamento, foi criado um relatório detalhado mostrando o desempenho da CNN. Este relatório fornece informações sobre o sucesso alcançado em relação à detecção de objetos e outras métricas relacionadas.

4.3 Testes

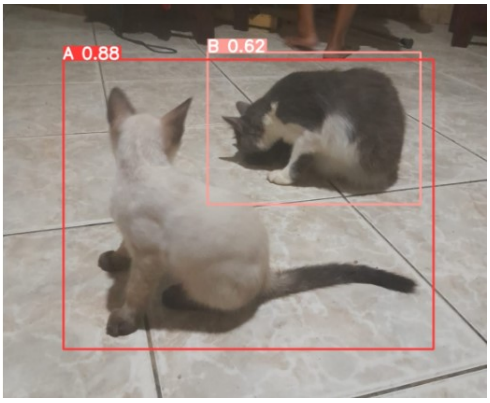
O modelo foi avaliado por meio de cerca de 760 imagens de validação presentes no próprio conjunto de dados, além de algumas imagens que o modelo nunca havia encontrado anteriormente. A Figura 24 mostra um exemplo de reconhecimento de imagem desconhecida usando YOLO. Neste reconhecimento, o gato “A” obteve 88% de confiança e o gato “B” obteve 62% de precisão, demonstrando capacidade de rotular corretamente.

5 MATERIAIS E MÉTODOS

Para criar o conjunto de dados, 3800 imagens foram tiradas com uma Mini Câmera de Vigilância Uso Interno YI IOT As fotografias foram tiradas de diferentes ângulos e posições dos animais, com o objetivo de proporcionar ao modelo uma variedade significativa de características individuais.

O processo de treinamento foi conduzido na plataforma em nuvem *Google Colab*. O hardware padrão disponível inclui opções de *Central Processing Unit* (CPU) ou GPU, com a

Figura 24 – Exemplos de imagens rotuladas pelo modelo



Fonte: Elaborado pelo Autor.

possibilidade de escolha pelo usuário. Para a realização do processo a opção escolhida foi a GPU, a plataforma oferece o modelo NVIDIA T4, uma unidade que acelera diversas tarefas na nuvem, incluindo treinamento e inferência para projetos de aprendizado profundo, aprendizado de máquina, análise de dados e processamento gráfico.

O código foi desenvolvido e executado em um notebook Dell G15, com *processador Intel® Core™ i5-13450HX* de 13ª geração, sistema operacional *Windows 11 Home*, placa de vídeo *NVIDIA® GeForce RTX™ 3050*, 8 GB de memória DDR5 e 256 GB SSD. Além disso, a mesma câmera utilizada para criar o conjunto de dados foi empregada na captura de imagens durante o processo de monitoramento.

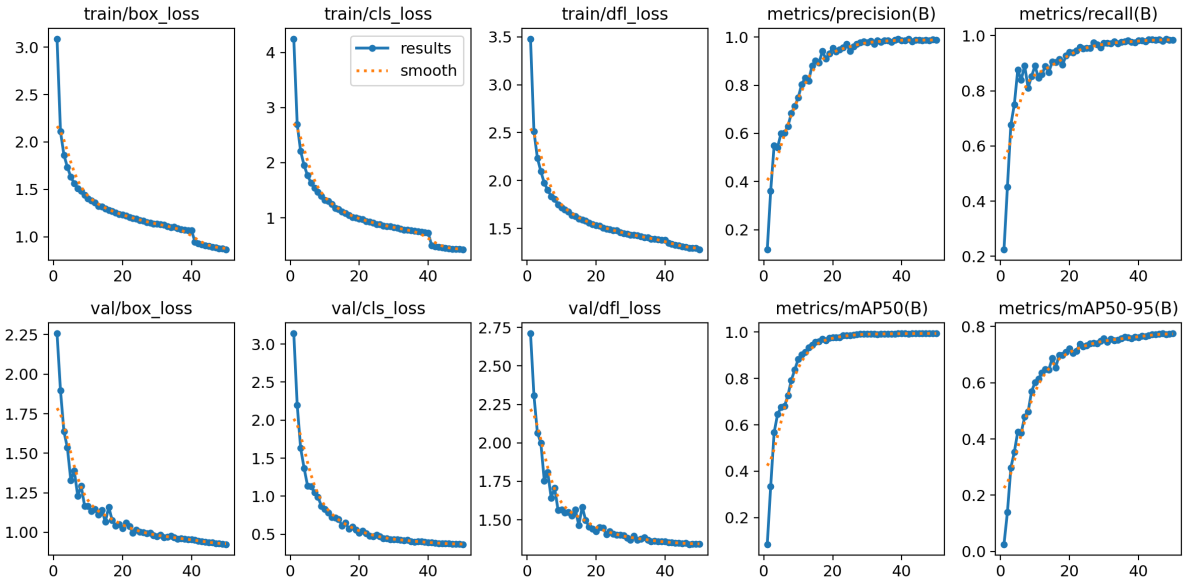
6 AVALIAÇÃO E RESULTADOS

Durante a discussão, foram comparados os resultados obtidos pelos dois modelos, destacando as vantagens entre um modelo *from scratch* e um que usa a abordagem de *transfer learning*. Essa análise visa determinar qual dessas abordagens YOLO demonstrou ser o modelo mais eficaz para a detecção de objetos. Ao destacar as características distintivas de cada modelo, buscamos fornecer uma compreensão abrangente para orientar a escolha do modelo mais adequado em termos de desempenho e aplicabilidade.

As Figuras 25 e 26 demostram gráficos que nos fornecem detalhes sobre o treinamento dos dois modelos. Nota-se que, à medida que os gráficos de loss diminuem, as taxas das métricas de precisão e *recall* aumentam. Esse comportamento pode ser atribuído ao refinamento contínuo dos modelos durante o treinamento, resultando em uma melhoria tanto na capacidade de acerto quanto na sensibilidade à identificação de objetos. Esta observação sugere que, ao longo das iterações do treinamento, os modelos demonstram uma eficácia crescente na tarefa de detecção de objetos.

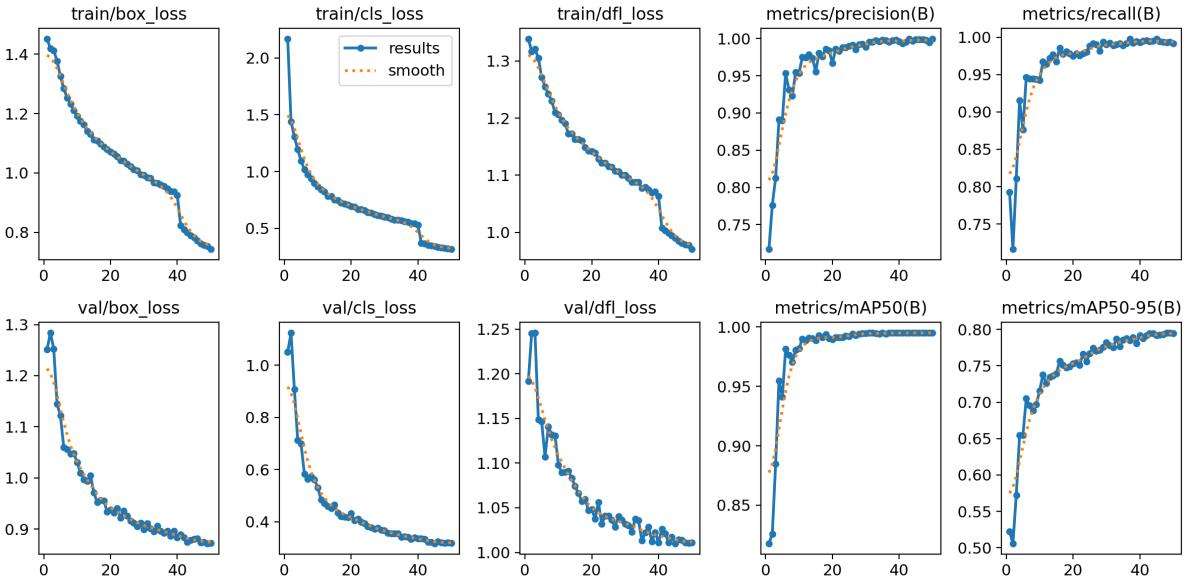
A Figura 28 revela o desempenho do modelo *from scratch* em uma matriz de confusão com quatro classes: Cleiton, Princesa, Tonzinha e "background". Analisando os resultados, observamos que o modelo apresenta 311 verdadeiros positivos (VP) para a classe Cleiton,

Figura 25 – Gráficos de Insights do Treinamento do Modelo *from scratch*.



Fonte: Elaborado pelo Autor.

Figura 26 – Gráficos de Insights do Ireinamento do Modelo com *transfer learning*.

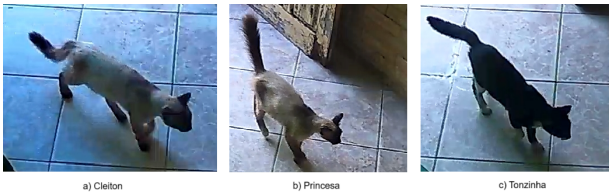


Fonte: Elaborado pelo Autor.

com apenas 6 falsos negativos (FN). Para a classe Princesa, são registrados 325 verdadeiros positivos (VP) e 4 falsos negativos (FN). Na classe Tozinha, o modelo atinge 329 verdadeiros positivos (VP) e 1 falso negativo (CN). A classe "background", que representa elementos que não pertencem a nenhuma das classes específicas. A 28 apresenta imagens reais das classes, permitindo a visualização das semelhanças e diferenças nas características dos gatos.

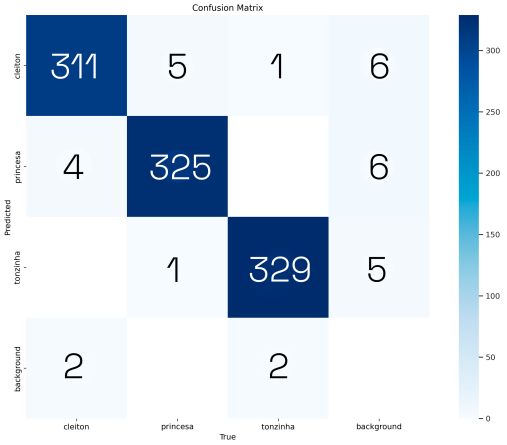
Na Figura 29, observamos que no modelo com *transfer learning*, a classe Cleiton apresenta uma quantidade considerável de verdadeiros positivos (313), indicando uma capacidade significativa de identificação, mas também registra quatro falsos negativos. A classe Princesa

Figura 27 – Imagens reais dos três gatos Cleiton, Princesa e Tonzinha.



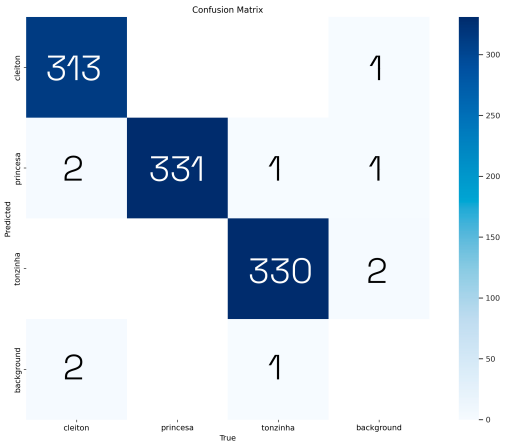
Fonte: Elaborado pelo Autor.

Figura 28 – Matriz de Confusão obtida através do modelo *from scratch*.



Fonte: Elaborado pelo Autor.

Figura 29 – Matriz de Confusão obtida através do modelo com *transfer learning*.



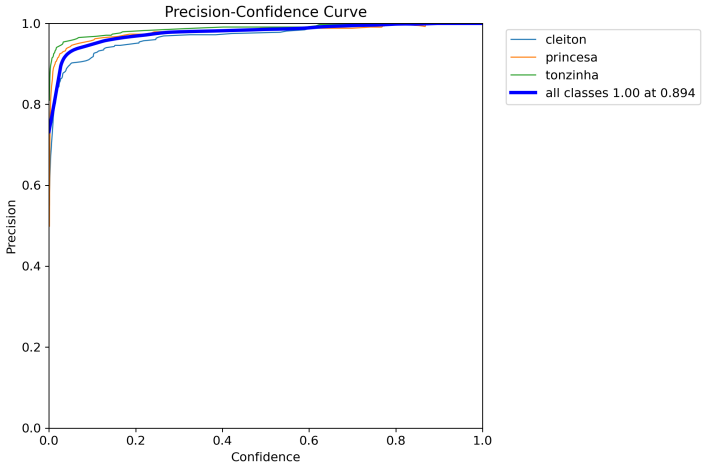
Fonte: Elaborado pelo Autor.

exibe um desempenho com 331 instâncias corretamente classificadas como verdadeiros positivos e três como falsos negativos. Na classe C, há uma identificação correta de 330 instâncias, sem falsos negativos.

Nas Figuras 30 e 31 mostramos as curvas de precisão e confiança do modelo de *from scratch* e do modelo de *transfer learning*, respectivamente. Ao analisar esses gráficos, algumas observações importantes se destacam. No caso do modelo *from scratch*, notamos que o modelo atinge boas taxas de acerto em todas as classificações dentro de um intervalo de confiança de $89,4\%$ a 100%. Por outro lado, quando o modelo foi examinado utilizando *transfer learning*,

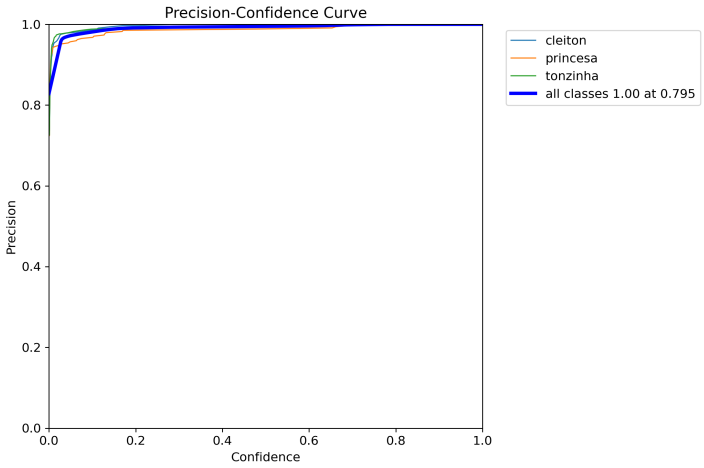
o mesmo intervalo de confiança estendeu-se para $79,5\%$. A análise mostra que o modelo de *transfer learning* é superior porque tem um escopo mais amplo no qual todas as inferências estão corretas em comparação com o modelo de personalização. Esses resultados indicam que o modelo de *transfer learning* possui maior confiabilidade e desempenho, o que é crucial em cenários de aplicação prática.

Figura 30 – Curva de Precisão-Confiança do Modelo *from scratch*.



Fonte: Elaborado pelo Autor.

Figura 31 – Curva de Precisão-Confiança do Modelo com *transfer learning*.

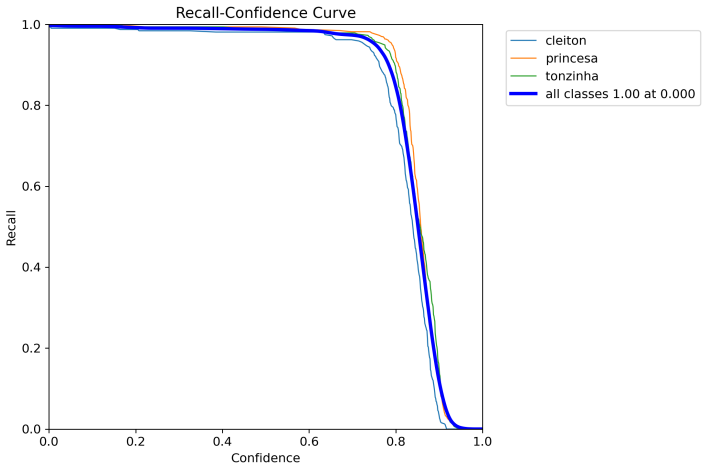


Fonte: Elaborado pelo Autor.

Podemos ver nas Figuras 32 e 33 que as curvas de *recall* e confiança de ambos os modelos apresentam desempenho satisfatório, mantendo alta sensibilidade em diferentes níveis de confiança. Destaca-se o modelo com *transfer learning*, apresentando *recall* mais significativo, indicando superior eficácia na recuperação de casos positivos, principalmente em níveis de confiança mais baixos.

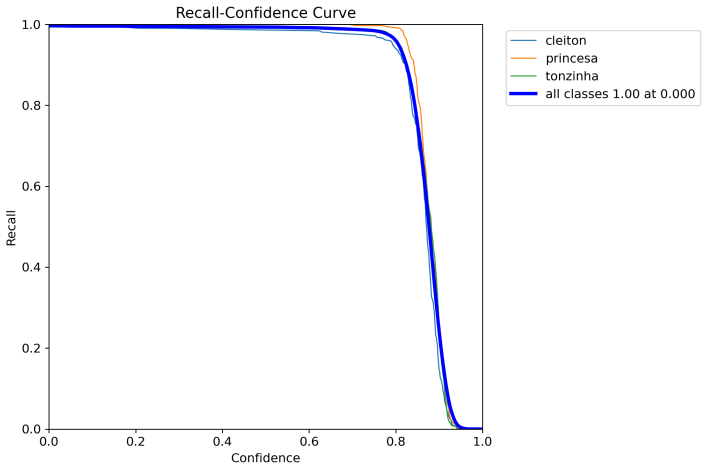
Veremos nas Figura 34 e 35 os gráficos de Precisão-*recall*, que no modelo *from scratch*, apresentou uma sólida capacidade de classificação, com precisões de $99,4\%$, $99,3\%$, e $99,5\%$

Figura 32 – Curva de Recall-Confiança do Modelo *from scratch*.



Fonte: Elaborado pelo Autor.

Figura 33 – Curva de Recall-Confiança do Modelo com *transfer learning*.

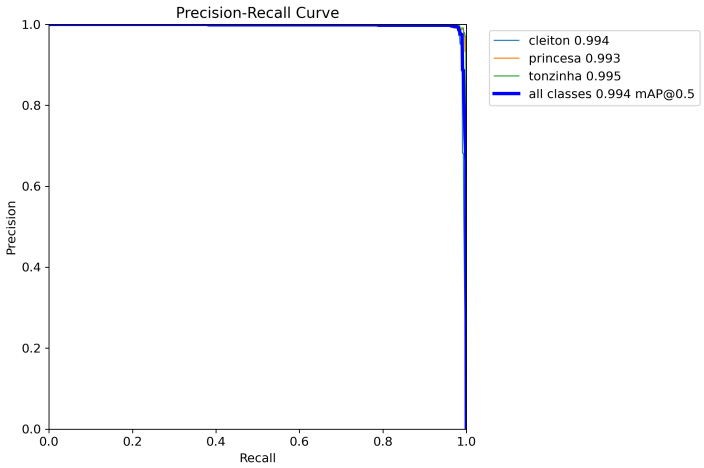


Fonte: Elaborado pelo Autor.

para as classes Cleiton, Princesa, e Tonzinha, respectivamente. O mAP50 para todas as classes foi de $99,4\%$.Por outro lado, o modelo com *transfer learning* destacou-se com precisões muito boas de $99,5\%$, $99,5\%$, e $99,5\%$ para as mesmas classes, acompanhadas de um mAP50 mais elevado, atingindo $99,5\%$. Essa comparação destaca a eficácia do *transfer learning*, demonstrando um treinamento de modelo de maior consistência, que são fatores importantes em cenários onde a precisão das previsões positivas é fundamental.

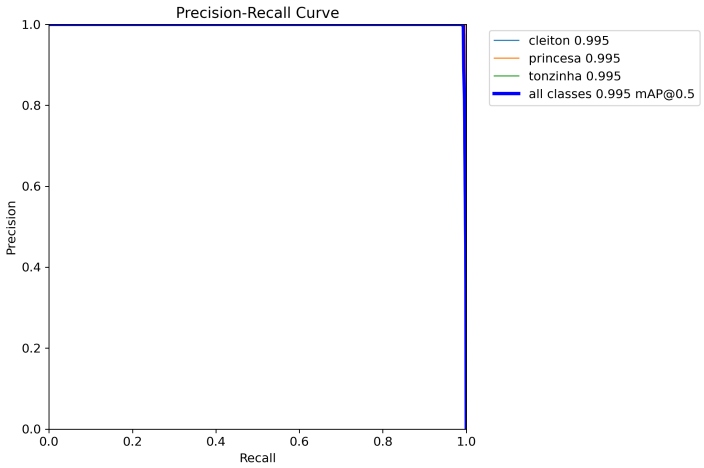
Examinando as Figuras 36 e 37 com gráficos dos resultados F1 relacionados à confiança podemos ver padrões diferentes entre o modelo de *transfer learning* e o modelo de *from scratch*. No modelo de *from scratch*, a pontuação F1 para todas as turmas combinadas foi de aproximadamente 62%, atingindo o nível de confiabilidade mais alto de 99%. Em contrapartida, o modelo com *transfer learning* mostra que todas as classes mantêm uma pontuação F1 constante em torno de $51,9\%$, com intervalo de confiança em torno de $99\ \%$. Estas diferenças mostram que o modelo de *transfer learning* têm uma relação mais clara entre confiança e desempenho,

Figura 34 – Curva de Precisão-*Recall* do Modelo *from scratch*.



Fonte: Elaborado pelo Autor.

Figura 35 – Curva de Precisão-*Recall* do Modelo com *transfer learning*.



Fonte: Elaborado pelo Autor.

mantendo um desempenho consistente em todos os níveis de confiança.

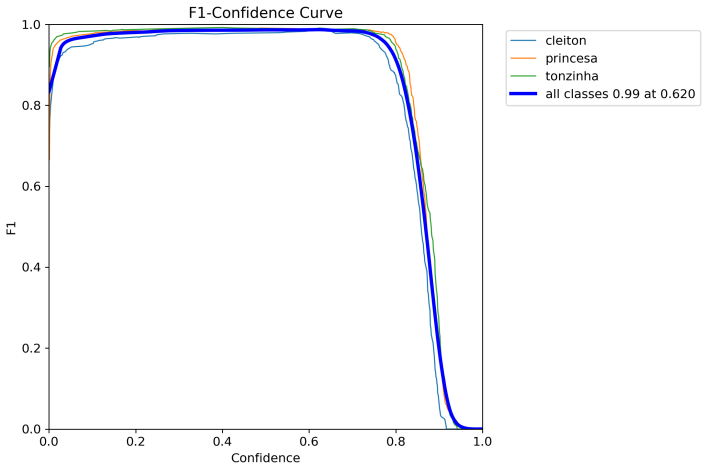
Na Tabela 1 e 2 podemos ver que desempenho do modelo *from scratch* e o do modelo com *transfer learning*, foram avaliados em diferentes classes, utilizando métricas padrão, incluindo Precisão (P), Recall (R), e mAP (mean Average Precision).

Classe	**Imagens**	**Instancias**	**P**	**R**	**mAP50**	**mAP50-95**
Geral	760	980	98,8%	98,5%	99,4%	77,5%
Cleiton	317	317	98,5%	98,1%	99,4%	74,6%
Princesa	331	331	98,8%	98,8%	99,3%	78,5%
Tonzinha	332	332	99,1%	98,5%	99,5%	79,4%

Tabela 1 – Resultados da Avaliação do Modelo *from scratch*.

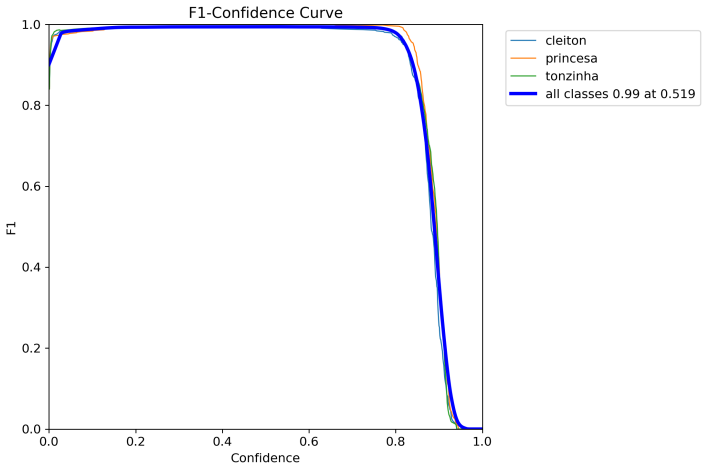
Em resumo, a análise comparativa entre o modelo com *transfer learning* e o modelo *from scratch* revela um desempenho notavelmente superior do modelo com *transfer learning*,

Figura 36 – Curva de F1-Confiança do Modelo *from scratch*.



Fonte: Elaborado pelo Autor.

Figura 37 – Curva de F1-Confiança do Modelo com *transfer learning*.



Fonte: Elaborado pelo Autor.

Classe	**Imagens**	**Instancias**	**P**	**R**	**mAP50**	**mAP50-95**
Geral	760	980	99,5%	99,3%	99,5%	79,5%
Cleiton	317	317	100%	98,6%	99,5%	77%
Princesa	331	331	98,9%	100%	99,5%	81,6%
Tonzinha	332	332	99,6%	99,4%	99,5%	80%

Tabela 2 – Resultados da Avaliação do Modelo com *transfer learning*.

conforme evidenciado pelas métricas de avaliação.

Para a métrica de precisão (**P**), o modelo com *transfer learning* apresenta uma melhoria geral em relação ao modelo *from scratch*. O modelo com *transfer learning* atinge uma precisão global de $99,5\%$, comparado a $98,8\%$ do modelo *from scratch*. Isso demonstra uma capacidade maior de identificar corretamente as instâncias em todas as classes.

A taxa de *recall* (**R**) também é superior no modelo com *transfer learning*, alcançando $99,3\%$ globalmente, em comparação com $98,5\%$ do modelo *from scratch*. Este aumento reflete

a eficácia aprimorada do modelo em identificar a maioria das instâncias presentes nas imagens.

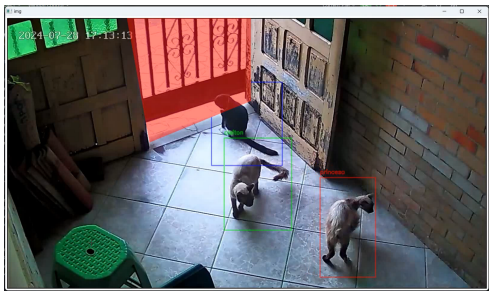
Quanto à métrica mAP50, o modelo com *transfer learning* exibe uma pontuação de $99,5\%$, superando os $99,4\%$ do modelo *from scratch*. Isso indica uma precisão levemente maior na determinação da posição de detecção. Já a métrica mAP50-95 do modelo com *transfer learning* é de $79,5\%$, enquanto o modelo *from scratch* apresenta $77,5\%$. Esta métrica mais alta no modelo com *transfer learning* sugere uma melhor consistência em diferentes níveis de confiança.

Ao analisar os resultados por classe, o modelo com *transfer learning* demonstra um desempenho aprimorado em todas as categorias. Para a classe Cleiton, o modelo com *transfer learning* atinge 100% de precisão, comparado a $98,5\%$ do modelo *from scratch*, e apresenta uma taxa de *recall* de $98,6\%$ em comparação com $98,1\%$. A classe Princesa também mostra melhorias, com $98,9\%$ de precisão e 100% de *recall* no modelo com *transfer learning*, frente a $98,8\%$ e $98,8\%$ do modelo *from scratch*.

Esses resultados destacam a eficácia do *transfer learning* como uma abordagem que impulsiona significativamente o desempenho do modelo de detecção de objetos, oferecendo maior precisão e consistência em comparação com a abordagem *from scratch*.

A Figura 38 mostra a janela de monitoramento, com a uma detecção feita durante sua execução, exibindo dois registros quem foram feitos,os *pets* B e C, e o local restrito em vermelho.

Figura 38 – Janela de monitoramento.



Fonte: Elaborado pelo Autor

A Figura 39 apresenta um exemplo de quando um dos animais passa pela área restita.

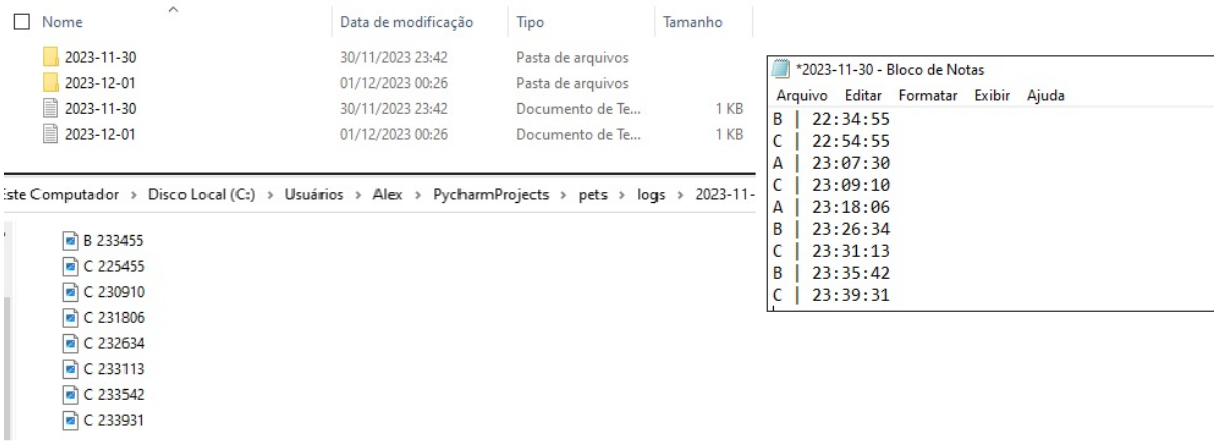
Figura 39 – Janela de monitoramento com uma detecção de restrição.

Fonte: Elaborado pelo Autor

A Figura 40 demostra os registros feitos, utilizando arquivos de texto separados pela data do monitoramento, contendo o nome e a hora da infração detectada e exemplos da forma que as

imagens são armazenadas durante o monitoramento realizado, separadas por pastas com a data do monitoramento, onde cada imagem é nomeada com o nome do animal e horário de detecção. Nos *logs* podemos verificar que o animal que mais ultrapassou a área restrita foi o animal C, com quatro interações.

Figura 40 – Demostração de como os *logs* são salvos



Fonte: Elaborado pelo Autor

7 CONCLUSÃO

Em resumo, este artigo apresentou uma abordagem abrangente para a construção e treinamento de modelos de detecção de objetos, explorando tanto um modelo *from scratch* quanto um modelo baseado em *transfer learning* com o *dataset* COCO. O modelo *from scratch*, treinado ao longo de 50 épocas, exibiu resultados promissores, com métricas satisfatórias que destacaram sua capacidade de discernir entre diferentes categorias de animais de estimação.

No entanto, uma análise comparativa mostra que o modelo de *transfer learning* supera significativamente o modelo *from scratch*. Com uma precisão geral de $99,5\%$, *recall* de $99,3\%$ e mAP50 de $99,5\%$, o modelo de *transfer learning* demonstrou excelente eficácia no reconhecimento preciso de objetos em imagens e foi testado com sucesso em diversas situações.

Esses resultados não apenas confirmam a utilidade da *transfer learning*, mas também destacam a importância de escolher uma abordagem apropriada para treinar modelos de reconhecimento de objetos. A implementação bem-sucedida desses modelos contribui para a gestão e segurança dos animais de estimação.

Além disso, em trabalhos subsequentes, a metodologia de criação do *dataset* inicial para detecção de animais específicos poderá ser aprimorada com a utilização de um sistema separado. Esse sistema, partindo de vídeos de cada animal, sozinho ou em grupo, poderá detectar, classificar e produzir o *dataset* automaticamente, sem a necessidade de capturar várias fotos e anotar as imagens manualmente. Isso não só economizaria tempo, mas também aumentaria a precisão na geração de *datasets* personalizados.

Diante disso, este estudo não só enriquece a compreensão das técnicas de detecção de objetos, mas também destaca a aplicabilidade prática dessas abordagens no monitoramento de animais domésticos. Com a contínua evolução dessas tecnologias, podemos antecipar melhorias contínuas e avanços que fortalecerão ainda mais a relação entre seres humanos e seus companheiros animais.

REFERÊNCIAS

ANDRADE, T. de et al. Confiança interpessoal e confiança organizacional como antecedentes dos comportamentos de cidadania organizacional. **REAd. Revista Eletrônica de Administração (Porto Alegre)**, 2021. Disponível em: <https://api.semanticscholar.org/CorpusID:245789897>.

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

BELLMAN, R. E. **Dynamic Programming**. [S.l.]: Princeton University Press, 1957.

BLUM, A.; MITCHELL, T. M. Combining labeled and unlabeled data with cotraining. In: **Proceedings of the Eleventh Annual Conference on Computational Learning Theory — COLT-98**. New York, USA: ACM Press, 1998. p. 92–100.

BOCHKOVSKIY, A. **YOLOv4 Source Code**. 2024. <https://github.com/AlexeyAB/darknet/blob/master/src/yolo.c>. Acessado em janeiro de 2024.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. **arXiv preprint arXiv:2004.10934**, 2020.

BURGHARDT, T.; CALIC, J. Real-time face detection and tracking of animals. In: IEEE. **2006 8th seminar on neural network applications in electrical engineering**. [S.l.], 2006. p. 27–32.

CARVALHO, R. Relação entre famílias, animais de estimação, afetividade e consumo: estudo realizado em bairros do rio de janeiro. **Revista Sociais e Humanas**, v. 26, p. 622–637, 09 2013. Citado na página 14.

CUNHA, W. S.; CAMARGO, V. V. de. Uma investigação da aplicação de aprendizado de máquina para detecção de smells arquiteturais. In: **Workshops on Software Visualization, Evolution and Maintenance**. [s.n.], 2019. Disponível em: <https://api.semanticscholar.org/CorpusID:203704838>.

DENG, C. Small-area population estimation: An integration of demographic and geographic techniques. 2013.

DING, X. et al. **Re-parameterizing Your Optimizers rather than Architectures**. 2023.

DING, X. et al. **RepVGG: Making VGG-style ConvNets Great Again**. 2021.

DOMINGUES, L. et al. Guarda responsável de animais de estimação na área urbana do município de pelotas, rs, brasil. **Ciência & Saúde Coletiva**, v. 20, p. 185–192, 2015. Citado na página 14.

DUAN, K. et al. Centernet: Keypoint triplets for object detection. In: **Proceedings of the IEEE/CVF international conference on computer vision**. [S.l.: s.n.], 2019. p. 6569–6578.

FENG, C. et al. **TOOD: Task-aligned One-stage Object Detection**. 2021.

FREUND, Y.; SCHAPIRE, R. E. et al. Experiments with a new boosting algorithm. In: CITESEER. **icml**. [S.l.], 1996. v. 96, p. 148–156.

GE, Z. et al. Yolox: Exceeding yolo series in 2021. **arXiv preprint arXiv:2107.08430**, 2021.

Ge, Zheng and Liu, Songtao and Wang, Feng and Li, Zeming and Sun, Jian. **YOLOX GitHub Repository**. 2024. <https://github.com/Megvii-BaseDetection/YOLOX>. Acessado em janeiro de 2024.

GEVORGYAN, Z. **SIoU Loss: More Powerful Learning for Bounding Box Regression**. 2022.

GHIASI, G.; LIN, T.-Y.; LE, Q. V. **DropBlock: A regularization method for convolutional networks**. 2018.

GIRSHICK, R. Fast r-cnn. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2015. p. 1440–1448.

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

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. [S.l.]: Springer, 2009.

HOSANG, J.; BENENSON, R.; SCHIELE, B. Learning non-maximum suppression. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 4507–4515.

HUSSAIN, M. Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection. **Machines**, v. 11, n. 7, 2023. ISSN 2075-1702. Disponível em: <https://www.mdpi.com/2075-1702/11/7/677>.

JIAO, R. et al. **Semi-supervised Semantics-guided Adversarial Training for Trajectory Prediction**. 2023.

JOCHER, G. **Ultralytics YOLOv5**. 2020. Disponível em: <https://github.com/ultralytics/yolov5>.

JOCHER, G.; CHAURASIA, A.; QIU, J. **Ultralytics YOLOv8**. 2023. Disponível em: <https://github.com/ultralytics/ultralytics>.

JOCHER, G.; CHAURASIA, A.; QIU, J. **Ultralytics YOLOv8 Configuração**. 2023. Acessado em janeiro de 2024. Disponível em: <https://docs.ultralytics.com/pt/usage/cfg/>.

JÚNIOR, G. de B. V. et al. DeterminaÇão das métricas usuais a partir da matriz de confusão de classificadores multiclasses em algoritmos inteligentes nas ciências do movimento humano. **Centro de Pesquisas Avançadas em Qualidade de Vida**, 2022. Disponível em: <https://api.semanticscholar.org/CorpusID:248066150>.

KIMATA, J.; NITTA, T.; TAMAKI, T. Objectmix: Data augmentation by copy-pasting objects in videos for action recognition. In: **Proceedings of the 4th ACM International Conference on Multimedia in Asia**. ACM, 2022. (MMAsia '22). Disponível em: <http://dx.doi.org/10.1145/3551626.3564941>.

KO, M.; KO, D.; LEE, J. **Deep Learning Bible - 4. Object Detection - Eng.** 2024. Online. Acessado em fevereiro de 2024. Disponível em: <https://wikidocs.net/188051>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in Neural Information Processing Systems**, v. 25, p. 1097–1105, 2012.

KUMAR, S.; SINGH, S. Biometric recognition for pet animal. **Journal of Software Engineering and Applications**, v. 7, p. 470–482, 2014.

LAW, H.; DENG, J. **CornerNet: Detecting Objects as Paired Keypoints**. 2019.

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

LEE, T. Y.; JEONG, M.-H.; PETER, A. Object detection of road facilities using yolov3 for high-definition map updates. **Sensors and Materials**, 2022. Disponível em: <https://api.semanticscholar.org/CorpusID:246461078>.

LI, C. et al. **YOLOv6 v3.0: A Full-Scale Reloading**. 2023.

LIN, T.-Y. et al. **Microsoft COCO: Common Objects in Context**. 2015. Disponível em: <https://arxiv.org/abs/1405.0312>.

LIU, H. et al. Sf-yolov5: A lightweight small object detection algorithm based on improved feature fusion mode. **Sensors**, v. 22, p. 5817, 08 2022.

MEITUAN. **Implementação do YOLOv6 no GitHub**. 2023. Acessado em janeiro de 2024. Disponível em: <https://github.com/meituan/YOLOv6/blob/main/hubconf.py>.

MICROSOFT. **How to understand automated machine learning**. 2023. Disponível em: <https://learn.microsoft.com/pt-br/azure/machine-learning/how-to-understand-automated-ml?view=azureml-api-2>.

NAANAA, H. Yolov5 fine-tuning on duckietown object detection dataset. 2023. Disponível em: <https://wandb.ai/hamnaanaa/Duckietown-Object-Detection/reports/YOLOv5-fine-tuning-on-Duckietown-Object-Detection-dataset--VmlldzozODc1ODI1>.

NAYEEM, M. J.; RANA, S.; ISLAM, M. R. Prediction of heart disease using machine learning algorithms. **European Journal of Artificial Intelligence and Machine Learning**, 2022. Disponível em: <https://api.semanticscholar.org/CorpusID:254344012>.

NEWMAN, J. et al. Automated pre-play analysis of american football formations using deep learning. **Electronics**, v. 12, p. 726, 02 2023.

OREN, M. et al. Pedestrian detection using wavelet templates. In: IEEE. **Proceedings of IEEE computer society Conference on computer vision and pattern recognition**. [S.l.], 1997. p. 193–199.

PATTANSHETTI, S. S.; NIVADE, S. I. Real-time object detection with pre-eminent speed and precision using yolov4. **International Journal of Research in Engineering, Science and Management**, v. 4, n. 7, p. 26–31, 2021.

RAFFEL, C. et al. **Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer**. 2023.

REDMON, J. **Exemplo de Código YOLO em C**. 2024. <https://github.com/pjreddie/darknet/blob/master/examples/yolo.c>. Acessado em janeiro de 2024.

REDMON, J. et al. You only look once: Unified, real-time object detection. **CoRR**, abs/1506.02640, 2015. Disponível em: <http://arxiv.org/abs/1506.02640>.

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

REZATOFIGHI, H. et al. **Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression**. 2019.

RIBEIRO, C. H. C. A tutorial on reinforcement learning techniques. In: **Supervised Learning Track Tutorials of the 1999 International Joint Conference on Neuronal Networks**. Washington: INNS Press, 1999.

Roboflow. **Supervision**. 2024. <https://roboflow.github.io/supervision/>. Código-fonte disponível em: <https://github.com/roboflow/supervision>. Licença MIT. Disponível em: <https://roboflow.github.io/supervision/>.

RODRIGUES, V. B. **Métricas de Avaliação: acurácia, precisão, recall... quais as diferenças?** 2019. <https://vitorborbarodrigues.medium.com/mÃl'tricas-de-avaliaÃğÃčo-acurÃącia-precisÃčo-recall-quais-as-diferenÃğas-c8f05e0a513c/>. Accessed: 2024-01-03.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. [S.l.]: Prentice Hall, 2009.

SANTOS, P. D. D.; JUNIOR, D. A. D. S. A importância do uso de drones no patrulhamento ambiental. **Brazilian Journal of Development**, 2023. Disponível em: <https://api.semanticscholar.org/CorpusID:259522243>.

SANTOS, R. P. dos; BEKO, M.; LEITHARDT, V. R. Q. Modelo de machine learning em tempo real para agricultura de precisão. **Anais da XXII Escola Regional de Alto Desempenho da Região Sul (ERAD-RS 2022)**, 2022. Disponível em: <https://api.semanticscholar.org/CorpusID:247833811>.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural Networks**, v. 61, p. 85–117, 2015.

SHIN, H.-C. et al. **Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning**. 2016.

SINGH, A. Selecting the right bounding box using non-max suppression (with implementation). **Analytics Vidhya**, 2020. Disponível em: <https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/>.

STRAUSS, E.; JÚNIOR, M. V. B.; FERREIRA, W. L. L. A importância de utilizar métricas adequadas de avaliaÇão de performance em modelos preditivos de machine learning. **Projectus**, 2023. Disponível em: <https://api.semanticscholar.org/CorpusID:260187436>.

TERVEN, J.; CORDOVA-ESPARZA, D. A comprehensive review of yolo: From yolov1 to yolov8 and beyond. **arXiv preprint arXiv:2304.00501**, 2023.

TIAN, Z. et al. **FCOS: Fully Convolutional One-Stage Object Detection**. 2019.

ULTRALYTICS. **Código-fonte YOLOv3 em Python**. 2024. <https://github.com/ultralytics/yolov3/blob/master/detect.py>. Acessado em janeiro de 2024.

ULTRALYTICS. **Implementação do YOLOv5 no GitHub**. 2024. Acessado em janeiro de 2024. Disponível em: <https://github.com/ultralytics/yolov5/blob/master/detect.py>.

WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. **YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors**. 2022.

WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. **Código-fonte YOLOv7 em Python**. 2024. <https://github.com/WongKinYiu/yolov7/blob/main/detect.py>. Acessado em janeiro de 2024.

WANG, X.; SONG, J. Iciou: Improved loss based on complete intersection over union for bounding box regression. **IEEE Access**, v. 9, p. 105686–105695, 2021. Disponível em: <https://api.semanticscholar.org/CorpusID:236920264>.

WEISS, E.; SLATER, M.; LORD, L. Frequency of lost dogs and cats in the united states and the methods used to locate them. **Animals**, v. 2, n. 2, p. 301–315, 2012. ISSN 2076-2615. Disponível em: <https://www.mdpi.com/2076-2615/2/2/301>. Citado na página 14.

WENG, K. et al. **EfficientRep:An Efficient Repvgg-style ConvNets with Hardware-aware Neural Network Design**. 2023.

Wikipédia. **Precisão e revocação**. 2024. Acessado em 6 de março de 2024. Disponível em: <https://pt.wikipedia.org/wiki/Precis%C3%A3o_e_revoca%C3%A7%C3%A3o>.

WITTEN, I. H. Data mining: Practical machine learning tools and techniques. **The Morgan Kaufmann Series in Data Management Systems**, v. 47, p. 130–131, 2005.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. **European Conference on Computer Vision**, 2014.

ZHANG, H. et al. **mixup: Beyond Empirical Risk Minimization**. 2018.

ZHANG, H. et al. **VarifocalNet: An IoU-aware Dense Object Detector**. 2021.

ZHANG, H.; ZHANG, S. Shape-iou: More accurate metric considering bounding box shape and scale. **arXiv preprint arXiv:2312.17663**, 2023.

ZHANG, Z. et al. **Bag of Freebies for Training Object Detection Neural Networks**. 2019.