UM ESTUDO ANALÍTICO ENTRE VERSÕES DO ALGORITMO DE VISÃO COMPUTACIONAL *YOLO*

Arthur Germano de Oliveira*
Alexandro Lima Damasceno**
Ruan Dos Santos Gondim***

RESUMO

Este trabalho apresenta uma análise comparativa das versões YOLOv4, YOLOv5 e YOLOv8, algoritmos de detecção de objetos amplamente utilizados em Visão Computacional. A YOLO é conhecida por sua eficiência e capacidade de realizar detecção em tempo real, o que a torna ideal para aplicações como vigilância, veículos autônomos e assistência médica. O estudo explora as Redes Neurais Convolucionas (CNNs), que são a base das melhorias contínuas da YOLO, permitindo maior precisão na identificação e localização de objetos em imagens e vídeos. A metodologia envolve a análise de desempenho dessas versões em cenários de detecção de veículos automotivos, utilizando métricas como Precisão, mAP e IoU. O treinamento das redes foi realizado com a metodologia split hold-out (80-20) para avaliar o impacto das atualizações em termos de precisão e eficiência. Os resultados indicam que a YOLOv8 é mais eficiente, enquanto a YOLOv4 demonstra uma capacidade superior de generalização em ambientes variados. Conclui-se que a escolha da versão YOLO depende das exigências específicas do projeto, destacando a relevância contínua da YOLO na evolução da Visão Computacional.

Palavras-chave: Inteligência Artificial. Detecção de Objetos. Redes Neurais Convolucionas. You Only Look Once.

* Autor, Graduando em Ciência da Computação, Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Aracati, CE, Brasil. E-mail: arthur.germano.oliveira07@aluno.ifce.edu.br
** Orientador, Mestre em Ciência da Computação, docente do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Aracati, CE, Brasil. E-mail: alexandro.lima@ifce.edu.br
*** Coorientador, Mestre em Ciência da Computação, docente do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Aracati, CE, Brasil. E-mail: ruan.gondim@ifce.edu.br

ABSTRACT

This study presents a comparative analysis of YOLOv4, YOLOv5, and YOLOv8, widely used object detection algorithms in computer vision. YOLO is known for its efficiency and real-time detection capabilities, making it suitable for applications such as surveillance, autonomous vehicles, and medical assistance. The study explores Convolutional Neural Networks (CNNs), which underpin the continuous improvements in YOLO, allowing for increased precision in identifying and locating objects in images and videos. The methodology includes performance analysis of these versions in vehicle detection scenarios, using metrics like Precision, mAP, and IoU. Training was conducted using an 80-20 split hold-out method to assess the impact of updates on accuracy and efficiency. Results indicate that YOLOv8 is more efficient, while YOLOv4 demonstrates superior generalization capability in varied environments. In conclusion, the choice of YOLO version depends on the specific project requirements, highlighting YOLO's ongoing relevance in advancing computer vision.

Keywords: Artificial Intelligence. Object Detection. Convolutional Neural Networks. You Only Look Once.

1 INTRODUÇÃO

A Inteligência Artificial vem desempenhando um papel fundamental em uma ampla área do cotidiano humano. Dentre as infinidades de possibilidades que a Inteligência Artificial proporciona à sociedade, está o uso de aplicações de Visão Computacional tais como: vigilância, veículos autônomos, assistência médica etc (CIMIRRO, 2022).

De acordo com Redmon et al. (2016), os humanos têm a capacidade de olhar para uma imagem e saber exatamente quais objetos estão na imagem, onde estão e como eles interagem. O sistema visual humano é rápido e preciso, permitindo realizar tarefas complexas como dirigir com pouco pensamento consciente. Utilizar-se de algoritmos robustos que sejam de rápido processamento e precisão significativa para detecção de objetos permitiria que computadores dirigissem carros, transmitindo informações da cena em tempo real para usuários humanos possibilitando o máximo potencial para sistemas autônomos responsivos de uso geral.

A capacidade de identificar e localizar objetos em imagens ou vídeos é uma tarefa desafiadora e altamente requisitada na área de processamento de imagens e Aprendizado de Máquina. Nesse contexto, surgiram diversas técnicas e algoritmos de Visão Computacional, cada um com seus pontos fortes e fracos.

Segundo Cimirro (2022), Redes Neurais Convolucionas *(CNN)* são arquiteturas que permitem serem treinadas. *CNN* é um método de aprendizado profundo *(Deep Learning)* com inspiração biológica que pode aprender com recursos constantes. De maneira semelhante aos processos tradicionais de Visão Computacional, uma *CNN* é capaz de aplicar filtros em dados

visuais de forma que a relação de vizinhança entre os *pixels* da imagem ao longo do processamento da rede seja mantida. Este tipo de rede vem sendo amplamente utilizada principalmente nas aplicações de classificação, detecção e reconhecimento em imagens e vídeos.

Visão Computacional é uma tarefa desafiadora. Recentemente, o *Deep Learning* tem sido adotado na detecção de objetos devido ao suporte de dispositivos de computação poderosos, as *GPUs* (NGUYEN et al., 2019).

Uma das abordagens mais notáveis e amplamente adotadas na detecção de objetos é o método de detecção *YOLO (You Only Look Once)*. A *YOLO* é conhecida por sua eficiência e capacidade de realizar detecção em tempo real, tornando-o ideal para muitas aplicações em tempo real. Oferecendo uma das melhores compensações entre precisão e velocidade na detecção de objetos. É a única rede neural que prevê o objeto em caixas delimitadoras e a probabilidade de classe em uma única avaliação. Desde sua primeira versão, a *YOLO* evoluiu consideravelmente, com várias novas funcionalidades e melhorias, cada uma trazendo a promessa de desempenho aprimorado e maior precisão na detecção de objetos.

Tendo como objetivo geral deste trabalho comparar a precisão entre versões da *YOLO* baseando-se na utilização das principais métricas para detecção de objetos. Com objetivos específicos buscou-se realizar um estudo aprofundado sobre as camadas das Redes Neurais Convolucionas, camadas arquiteturais da *YOLO*, pesquisa para busca de artigos para estudos e testes através da metodologia proposta e análise dos resultados obtidos.

Este trabalho estudou algumas versões específicas do *YOLO*, destacando suas diferenças fundamentais, como capacidade de detecção de objetos, desempenho em termos de precisão, além de discutir casos de uso típicos para cada versão. Com base nessa análise comparativa, pretende-se fornecer referências valiosas para a escolha da melhor versão do *YOLO*, dependendo das exigências do projeto e das limitações computacionais.

Este estudo não apenas incita sobre o panorama em constante evolução da detecção de objetos, mas também destaca o impacto significativo que o *YOLO* teve e continua a ter no campo da Visão Computacional. A compreensão das diferenças entre as versões do *YOLO* é fundamental para que os profissionais e pesquisadores possam tomar decisões informadas na implementação de soluções de detecção de objetos em suas aplicações específicas.

Ao final deste estudo, espera-se contribuir para o avanço contínuo dessa grande área de pesquisa e desenvolvimento de novas tecnologias com base na Visão Computacional.

Em relação a agenda deste trabalho, na Seção 2 está definida a fundamentação teórica contextualizando sobre Inteligência Artificial com ênfase em Redes Neurais Convolucionas e a relevância destes tópicos com o trabalho. Na Seção 3 está presente os trabalhos relacionados a este projeto, nesta seção foi estudado os principais conceitos usados pelos autores desta área. Na Seção 4 está exposto a metodologia adotada e suas etapas de desenvolvimento. Na Seção 5 são apresentados os resultados obtidos através da analise dos dados de teste e uma discussão a respeito dos mesmos. Ao final, na Seção 6 são expostas as conclusões obtidas.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão abordados os tópicos fundamentais para melhor compreensão sobre os principais aspectos técnicos utilizados no desenvolvimento deste trabalho, além de um aprofundamento específico em Redes Neurais Convolucionas e métricas utilizadas na análise dos dados.

2.1 Inteligência Artificial

A Inteligência Artificial (IA) como é definida por Bellman (1978) é a automação de atividades que associamos ao pensamento humano, atividades como: a tomada de decisões; a resolução de problemas; o aprendizado[...].

A IA é o pilar do constante progresso no ramo de Visão Computacional e na detecção de objetos (TAULLI, 2020). Neste ramo, a IA tem como característica capacitar os sistemas à aprender e compreender de forma autônoma o conteúdo visual das imagens, permitindo que as máquinas tomem decisões com base nos dados previamente processados na rede neural durante a fase de treino e validação dos modelos criados para processamento de imagens (COELHO et al., 2017).

2.2 Visão Computacional e Detecção de Objetos

A Visão Computacional é uma área de conhecimento interdisciplinar que combina elementos de processamento de imagens, Inteligência Artificial e *Deep Learning* para permitir que computadores entendam e interpretem informações visuais a partir de imagens ou vídeos (LOPES, 2012). A detecção de objetos é uma das tarefas mais desafiadoras na área da Visão Computacional e tem uma ampla gama de aplicações práticas, desde a vigilância de segurança até a assistência médica e a condução de veículos autônomos (ANDRADE, 2022).

Os métodos tradicionais de detecção de objetos muitas vezes empregavam técnicas baseadas em características específicas, como bordas: e texturas (para identificar regiões de interesse) estes métodos frequentemente necessitavam de múltiplos passos, como: extração de características, segmentação e classificação, o que aumentava a complexidade computacional e introduzia fontes potenciais de erro. Além disso, a generalização desses métodos, as diversas condições ambientais de iluminação, perspectivas e escalas de objeto era frequentemente uma problemática, resultando em desempenho variável e limitações na generalização para cenários do mundo real.

A eficiência em tempo real era uma das principais preocupações dos métodos tradicionais. A necessidade de realizar várias etapas sequenciais tornavam esses métodos inadequados para aplicações que demandavam respostas em tempo real, como em sistemas de monitoramento de tráfego e interações em tempo real.

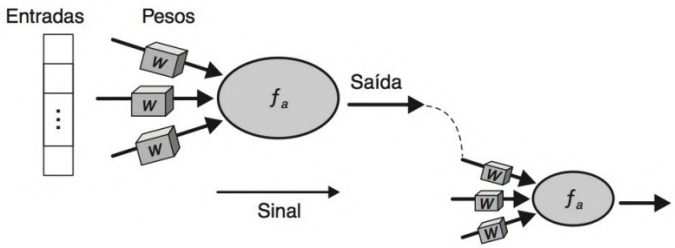
2.3 *Deep Learning*

Deep Learning (Aprendizado Profundo) é o subcampo da IA que tem ênfase em desenvolver modelos de Redes Neurais que conseguem tomar decisões através da identificação de padrões extraídos dos dados. O aprendizado profundo é mais usado em grandes conjuntos de dados com alto nível de complexidade. É uma forma flexível de aprendizado, pois permite que computadores aprendam através da experiência e entendam o mundo através de uma hierarquia de conceitos. Desta forma o computador tem total autonomia pois se baseia na experiência obtida, não havendo necessidade de supervisão operacional humana em especificar formalmente o conhecimento que o computador precisa. A hierarquia de conceitos permite que o computador aprenda conceitos complexos construindo-os a partir de conceitos mais simples (LAGE, 2023).

2.4 Redes Neurais

Para discorrer a respeito de Redes Neurais é necessário antes definir o conceito de neurônios em IA. O neurônio é a unidade de processamento fundamental de uma Rede Neural Artificial (RNA). Cada unidade de processamento desempenha um papel simples, com uma entrada que recebe um valor específico. O princípio por trás da criação dos neurônios de uma rede neural surgiu através do conceito de *Natural Computing* que usa métodos de computação de dados baseados em como a natureza realiza essa tarefa (TEIXEIRA et al., 2022). Na Figura 1 é possível observar a estrutura que representa um neurônio artificial:

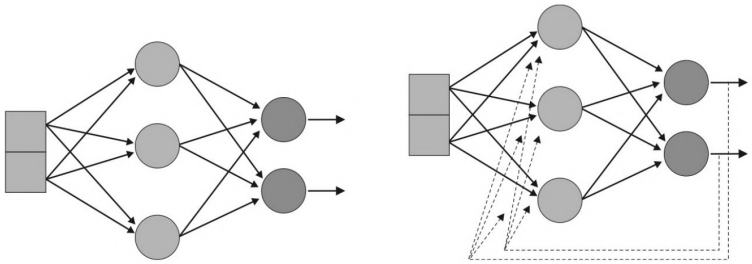
Figura 1 – Figura com representação de um neurônio artificial



(FACELI et al., 2021).

Nas Redes Neurais cada neurônio possui uma conexão que conta com um peso que serve para armazenar o conhecimento. Cada um dos neurônios da rede possui uma função de ativação que define se o neurônio vai propagar ou inibir o sinal. Durante o treinamento da rede são distribuídos pesos para cada conexão, servindo como parte do processo de decisão de quais neurônios serão ativados na próxima camada. A função de ativação irá definir a intensidade da ativação, que neste caso possui diversos tipos *(Sigmoid, ReLU, ELU etc.)* onde a escolha da função determina como devem ser as saídas de cada neurônio (CIMIRRO, 2022). Como evidencia a Figura 2:

Figura 2 – Figura com representação de dois tipos de Redes Neurais *feedforward* e recorrente



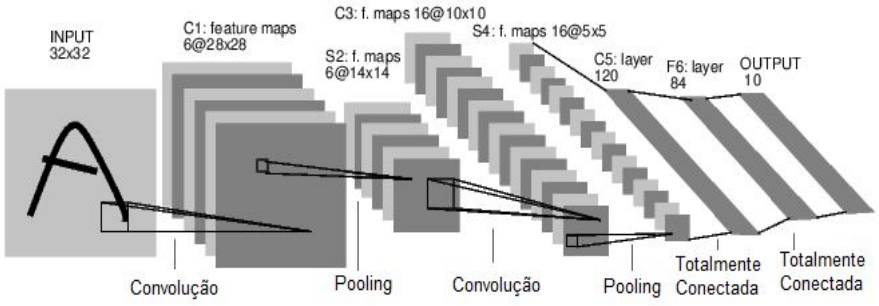
(a) Rede *feedforward* (b) Rede recorrente

(FACELI et al., 2021).

2.5 Redes Neurais Convolucionas

Nesta tópico é abordado características de cada uma das camadas que compõem as Redes Neurais Convolucionas, como: Camada convolução 2.5.1, Camada de *pooling* 2.5.2 e Camada totalmente conectada 2.5.3. Segundo Imamura et al. (2021) uma rede neural convolucional (*Convolutional Neural Network – CNN*) é muito utilizada em Visão Computacional devido a sua facilidade de extrair padrões dos *pixels* sem a necessidade de se realizar algum tipo de pré-processamento. A aplicação do filtro na camada de convolução realiza o mapeamento das características mais relevantes na imagem, esses filtros geram mapas de características *(Features Maps)* (GALVÃO, 2023). Como é observado na Figura 3:

Figura 3 – Figura de demonstração da arquitetura *LeNet* com camadas convolucionais, *pooling* e totalmente conectadas para classificação de imagens na entrada

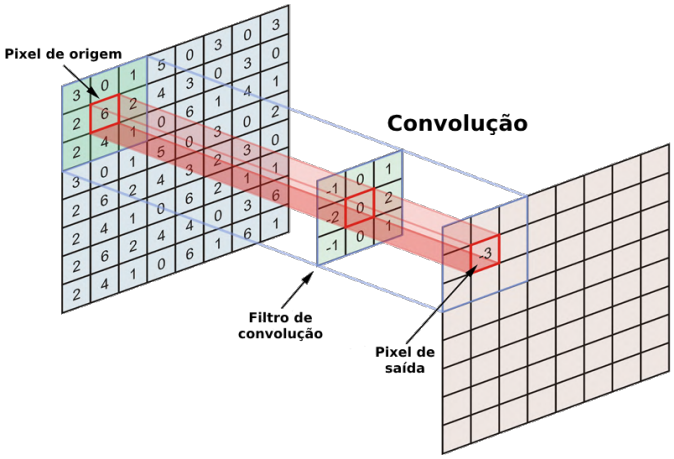


(LECUN et al., 1989).

2.5.1 Camada convolução

Na camada de convolução é realizado um processo que consiste em um tipo especializado de operação linear, nesta operação é onde são extraídos as características mais relvantes da imagem. Como diz o autor Cimirro (2022), são usados filtros para realizar as convoluções, gerando mapas de características. Cada filtro tem sua dimensão reduzida, mas se entendendo por toda a profundidade do volume de entrada. No caso o volume se resume ao número de canais que a imagem possui *(R, G, B)*. A Figura 4 demonstra uma melhor visualização do funcionamento da operação de convolução.

Figura 4 – Operação de convolução com filtro de dimensão 3x3

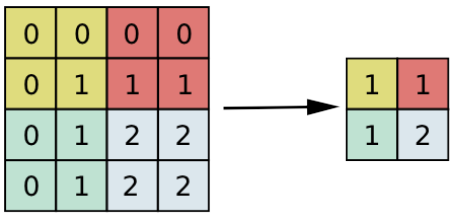


(ARAÚJO et al., 2017).

2.5.2 Camada de pooling

Na camada de *pooling*, (realizada após a camada convolucional) é executado um procedimento de redução progressiva na dimensão espacial do volume de entrada, consequentemente reduzindo o custo de processamento da rede. De acordo com Imamura et al. (2021), a forma mais comum de *pooling* consiste em substituir os valores de uma região pelo valor máximo, essa operação é conhecida como *max-pooling*. Como pode ser notado na Figura 5.

Figura 5 – Aplicação *max-pooling* em imagem 4x4 usando filtro 2x2

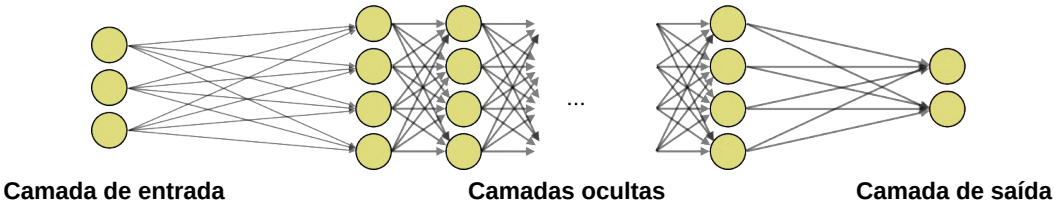


(ARAÚJO et al., 2017).

2.5.3 Camada totalmente conectada

Após percorrer todo o processo de extração das características da imagem, (gerados nas camadas preliminares) é necessário realizar mais um passo dentro da camada totalmente conectada, que é a responsável por traçar um caminho de decisão para cada classe de resposta. Esta técnica é similar ao uso da *Non-maximum Suppression (NMS)* que consiste em selecionar uma única entidade de classe entre diversas outras entidades de classes sobrepostas. Assim como abordado na Figura 6:

Figura 6 – Camada totalmente conectada

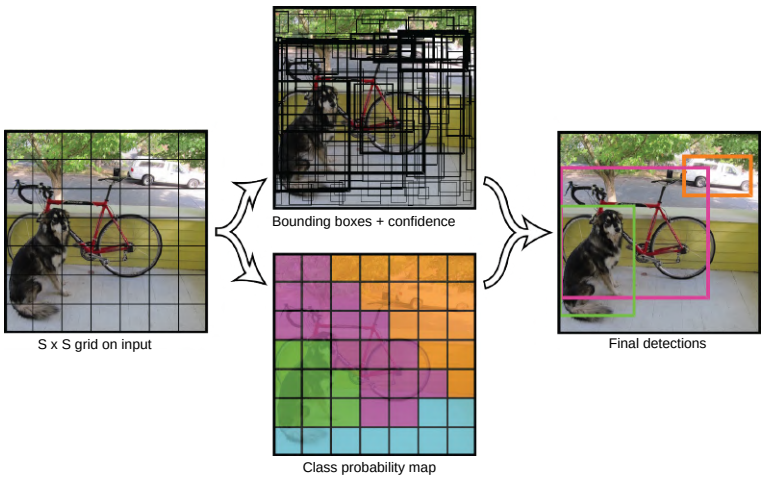


Autor (2024).

2.6 *You Only Look Once*

A *YOLO* ou *"You Only Look Once"*, é um dos métodos mais notáveis e amplamente adotados na detecção de objetos. Sua principal característica é a capacidade de realizar detecção em tempo real, tornando-o uma escolha ideal para aplicações que requerem baixa latência (menor tempo de processamento de inferência) (GOMES, 2022). A *YOLO* aborda a detecção de objetos como um problema de regressão, prevendo diretamente as coordenadas das caixas delimitadoras e as probabilidades das classes em uma única avaliação da rede neural (ANDRADE, 2022). Um exemplo do funcionamento da *YOLO* é ilustrado na Figura 7:

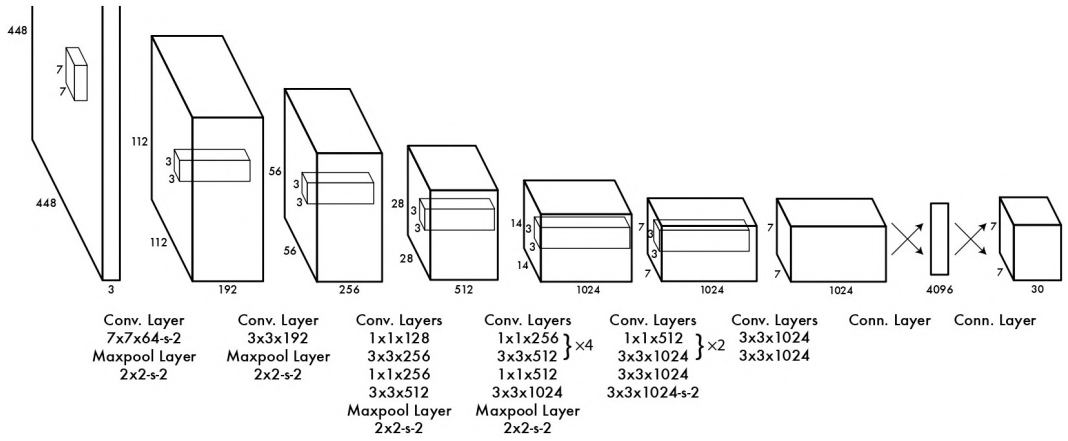
Figura 7 – Exemplo do funcionamento da YOLO



(REDMON et al., 2016).

Como descreve no seu trabalho Redmon et al. (2016) afirma que a *YOLO* tem similaridades com os demais detectores de objetos. Como por exemplo, a *R-CNN*. A grande diferença é a forma como a *YOLO* aborda o problema da detecção de objetos e a forma como foi projetado o design da rede, que consiste em 24 camadas de convolução e 2 totalmente conectadas como demonstrado no exemplo abaixo. A Figura 8 demonstra de forma visual o esquema arquitetural da *YOLO*.

Figura 8 – Camadas da rede convolucional *YOLO*



(REDMON et al., 2016).

De acordo com Redmon et al. (2016) a arquitetura da rede foi inspirada no modelo de classificação de imagens do *GoogLeNet*, porém não é citado mais detalhes a respeito de parâmetros que foram usados em cada uma das camadas convolucionais desenvolvidas. Contudo, foi referido que estas mesmas 24 camadas da rede foram pré-treinadas dentro da *ImageNet classification task*.

2.7 Métricas

Para verificar a qualidade da detecção de objetos e a validação das imagens do respectivo modelo treinado, é preciso observar os dados da classificação das imagens através da saída de resultados da rede. Essa saída permite com que seja possível a geração de métricas capazes de avaliar a qualidade e o desempenho do modelo treinado.

Existem métricas de avaliação de classificação binária amplamente utilizadas na literatura. Precisão e Interseção pela União são métricas comumente utilizadas para avaliar modelos de detecção de objetos, bem como os modelos da *YOLO*. Tendo em vista que uma das etapas do trabalho é realizar a detecção de objetos, temos duas possibilidades de resposta da rede: sendo positiva (para quando o modelo detecta a caixa delimitadora do objeto presente na imagem) e negativa (para quando o modelo não detecta nenhum objeto) (ARAÚJO et al., 2022).

2.7.1 Precisão e Recall

Para compreender a razão entre precisão e *recall*, é necessário entender as frequências de classificação para cada modelo avaliado, sendo elas:

- Verdadeiro Positivo *(True Positive - TP)*: Na ocorrência de exemplos classificados de maneira positiva correta.
- Verdadeiro Negativo *(True Negative - TN)*: Na ocorrência de exemplos classificados de maneira negativa correta.

- Falso Positivo *(False Positive - FP)*: Na ocorrência de exemplos classificados de maneira positiva incorreta.
- Falso Negativo *(False Negative - FN)*: Na ocorrência de exemplos classificados de maneira negativa incorreta.

Precisão é a razão entre verdadeiros positivos e a soma dos verdadeiros positivos com os falsos positivos, sendo considerado como a relação ao total de vezes que o modelo tenta acertar. A Figura 9 demonstra a matriz de confusão com essas classificações.

$$Precisão = \frac{VerdadeiroPositivo}{VerdadeiroPositivo + FalsoPositivo}$$

Figura 9 – Tabela demonstrando a precisão na matriz de confusão indicada pela célula na cor azul

Autor (2024).

Recall é a razão entre verdadeiros positivos e a soma dos verdadeiros positivos com os falsos negativos, sendo considerado taxa de detecção, em outras palavras, de todas as amostras que ele poderia classificar como positivas, quantas ele acertou. A Figura 10 ilustra como funciona a matriz de confusão para o *recall*.

$$Recall = \frac{VerdadeiroPositivo}{VerdadeiroPositivo + FalsoNegativo}$$

Figura 10 – Tabela demonstrando o recall na matriz de confusão indicada pela célula na cor azul

Autor (2024).

2.7.2 Precisão Média, mAP e Interseção pela União

A Precisão Média (*AP*) é uma métrica frequentemente usada para suavizar a curva de precisão-*recall*. Ela resume essa curva em um único valor que representa a média das precisões. O *Mean Average Precision (mAP)*, por sua vez, calcula a média das precisões entre diferentes classes, sendo amplamente utilizado em tarefas de reconhecimento, como a segmentação de imagens. Essa métrica realiza uma comparação entre a caixa delimitadora real e a caixa predita pelo modelo. Quanto maior a área de interseção entre as duas, melhor é o desempenho do modelo em termos de acurácia (CIMIRRO, 2022).

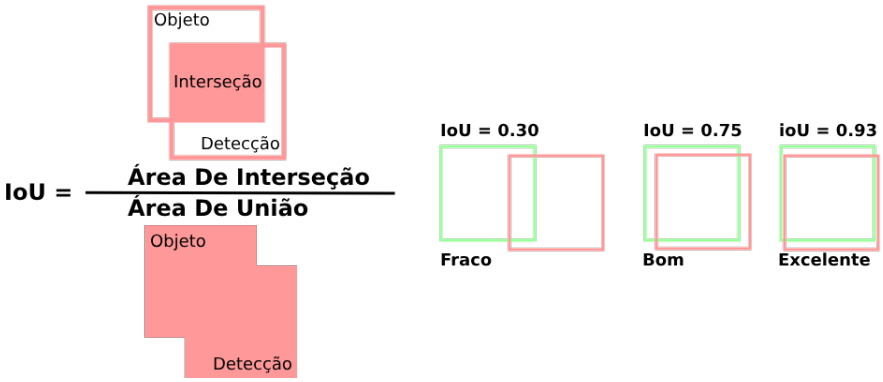
A *AP* é obtida ao calcular a precisão média para cada classe nos dados testados, considerando diferentes pontos de *recall*. A precisão é calculada dividindo o número de verdadeiros positivos pela soma de verdadeiros positivos e falsos positivos. A média dessas precisões resulta na Área Sob a Curva (*AUC*) da curva de precisão-*recall*. A média dessa precisão média para as classes individuais é o que gera o *mAP*. Dentro das métricas de porcentagem do *mAP* encontramos suas variantes como: *mAP*50 e *mAP*50-95 (ARAÚJO et al., 2022).

A Interseção pela União (*IoU*) é uma outra métrica de avaliação comum, frequentemente utilizada para detecção de objetos e avaliação de caixas delimitadoras. Essa métrica mede a qualidade da sobreposição entre a área delimitada pela caixa real e a caixa predita pelo modelo. O *IoU* é calculado utilizando a seguinte fórmula:

$$IoU = \frac{ÁreaDetectada \cap ÁreaReal}{ÁreaDetectada \cup ÁreaReal} \tag{1}$$

Essa métrica assume valores entre 0 e 1, onde valores mais próximos de 1 indicam um melhor desempenho, ou seja, uma maior sobreposição entre as áreas. Além disso, o *IoU* considera tanto verdadeiros positivos quanto falsos positivos e negativos, e sua avaliação é frequentemente usada para medir o desempenho de modelos em tarefas de detecção de objetos, como na métrica utilizada pelo desafio *PASCAL VOC* (ARAÚJO et al., 2022). A Figura 11 exemplifica visualmente como a Interseção pela União avalia a área de interseção entre a caixa predita e a caixa real.

Figura 11 – Representação visual da Interseção pela União



Autor (2024).

3 TRABALHOS RELACIONADOS

Na procura de trabalhos, foram encontrados inúmeros, que citam processamento de imagens e detecção de objetos através da *YOLO*. Porém, foram filtrados trabalhos onde caracterizavam especificamente o uso da *YOLO* para processamento de imagens de carros, veículos e placas de trânsito através da *string* de busca na Tabela 1:

Tabela 1 – *String* de busca

("YOLO")
AND ("VEICULOS"OR "VEICULAR")
AND ("PLACAS"OR "PLACA")
AND ("CARRO"OR "CARROS")

Autor (2024).

Os trabalhos utilizados foram encontrados em duas plataformas, *Google* Acadêmico e Portal de Periódicos da CAPES através do acesso CAFe.

3.1 Reconhecimento de imagens: uso do método *YOLO* no reconhecimento de placas de trânsito

O autor Jean Lucas da Silva Cimirro (2022) aborda o crescente uso da detecção de objetos na Visão Computacional, com foco no reconhecimento de placas de trânsito. O autor destaca a falta de uma base de dados adequada para placas nacionais, levando à criação de um dataset próprio. A metodologia envolve a divisão das placas em três classes (regulamentação, advertência e indicação) com um total de 3283 imagens para treinamento e validação, utilizando o modelo YOLOv5.

A ferramenta desenvolvida inclui uma interface gráfica e atende a requisitos como inserção de vídeo, conversão para 18*fps*, detecção de objetos e armazenamento em um banco de dados georreferenciado. Durante o treinamento da rede neural, o autor criou três modelos distintos *(Small, Large* e *Extra Large)* para comparação de resultados e otimização do projeto. O

autor destaca a importância da conversão para 18 *fps* para manter a sincronização dos dados de localização.

Nas considerações finais, ele discute os resultados obtidos com a *YOLO* para detecção de placas de trânsito brasileiras, destacando a contribuição do seu trabalho na criação de um *dataset* nacional disponível através do *GitHub* e na disponibilização da ferramenta com interface gráfica. O trabalho é bem implementado, preenche lacunas em trabalhos correlatos e fornece um *dataset* robusto para usos futuros.

3.2 Detecção e reconhecimento de placas de licenciamento veicular em tempo real usando *CNN*

Os autores Marcelo Eidi Imamura, Francisco Assis da Silva, Leandro Luiz de Almeida, Danillo Roberto Pereira, Almir Olivette Artero e Marco Antonio Piteri destacam a relevância de seu trabalho, que aborda a ampla frota de veículos licenciados no Brasil. O foco principal é a automação da detecção de placas veiculares, dada a alta demanda no país. A metodologia proposta é dividida em quatro etapas: Detecção da placa de licenciamento veicular, delimitação da área da placa, segmentação dos caracteres, reconhecimento dos caracteres.

A detecção das placas é realizada com o uso da *YOLO*, utilizando um *dataset* próprio capturado por uma *GoPro Hero 5 Black* em diferentes condições. O treinamento do modelo resultou em um erro de 0,2631 (quanto menor o erro, melhor a precisão do modelo) A delimitação da área das placas envolve o uso de diversas técnicas, como: filtros em tons de cinza, filtro gaussiano, filtro laplaciano e *threshold* de *Ostu*, (para lidar com problemas de confusão na detecção devido ao desgaste dos caracteres das placas).

Na etapa de experimentos, o modelo treinado da *YOLO* demonstrou 100% de acertos na detecção e 86,66% na segmentação, considerando uma média entre letras e números. Os autores concluem que a metodologia funciona em ambientes não controlados, podendo ser aplicada na fiscalização de trânsito em diversos cenários.

O trabalho destaca a preocupação dos autores com a precisão da *YOLO* em ambientes diversos, utilizando técnicas mistas com filtros variados e equalização de histograma para suavização de amostras discrepantes. No entanto, o trabalho não descreve qual versão da *YOLO* especificamente foi usada e não demonstra a fundo as métricas normalmente utilizadas na *YOLO* como: *mean Average Precision (mAP)*, suas variáveis e o *recall*.

3.3 *You Only Look Once*: Detecção unificada de objetos em tempo real

Os autores Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi apresentam a criação de um novo modelo de arquitetura para detecção de objetos utilizando Redes Neurais Convolucionas: a *YOLO (You Only Look Once)*, a ideia para a criação do modelo foi imaginar a detecção de objetos de forma reestruturada como um único problema de regressão na imagem, para detectar os objetos com base em classificadores e probabilidade de classes.

A arquitetura propõe uma rede de detecção com 24 camadas convolucionais e 2 camadas totalmente conectadas, as camadas convolucionais foram pré-treinadas na tarefa de classificação *ImageNet* com metade da resolução e depois dobrada para a resolução de detecção.

É interessante salientar que os autores discorrem bastante a respeito das similaridades entre a *YOLO* e a *R-CNN*, o estudo tem bastante ênfase na comparação entre os modelos *Fast R-CNN* e *R-CNN*. Na seção de comparação entre os sistemas de detecção em tempo real, a *YOLO* e *Fast YOLO* se destacam e apresentam resultados significativos, com um bom *mAP* e *fps* altos, bem maiores que os demais.

Concluem ressaltando como a *YOLO* é rápida e precisa, tornando uma aplicação ideal para Visão Computacional, apontam que a *Fast YOLO* é o detector mais rápido da literatura e a *YOLO* oferece o que se tem de mais moderno na detecção de objetos em tempo real. Afirmando a capacidade de generalização, bem como sendo a aplicação ideal para sistemas que dependem de detecção rápida e robusta.

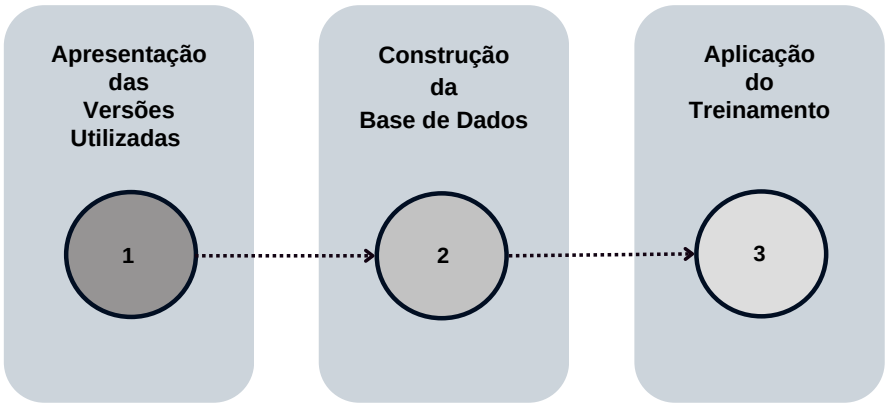
Neste artigo os autores apresentam uma arquitetura nova, rápida e robusta capaz de lidar com detecção em tempo real. O trabalho é muito bem fundamentado e apresenta solidez com pontos fortes e fracos da arquitetura se comparado com as demais tecnologias já existentes na Visão Computacional. Os autores conseguem posicionar bem a *YOLO* diante dos demais detectores como sendo a melhor forma para detectar objetos tendo bom equilíbrio entre os resultados apresentados.

Em aspectos gerais, após observados todos os artigos que fundamentam este estudo é possível notar parâmetros e métricas mais relevantes a serem consideradas para o uso em Redes Neurais Convolucionas, e principalmente na *YOLO*. Estas métricas foram adicionadas a este estudo devido a grande relevância dada pelos autores em seus respectivos estudos, onde os mesmos comprovam de forma matemática o desempenho das redes e a real eficiência do algoritmo. Por fim vale destacar a amplitude deste estudo onde ele, não somente verifica a capacidade de cada versão da *YOLO*, como também a compara entre outras versões, onde nenhum dos trabalhos citados fez tal comparação.

4 METODOLOGIA

Nesta seção serão apresentadas as atividades de elaboração dos modelos de detecção de objetos que serão analisados neste estudo, destacando os principais componentes do processo de desenvolvimento, sendo eles: apresentação das versões do algoritmo de detecção de objetos utilizados, a construção da base de dados para o treinamento do modelo e a aplicação do treinamento. Definindo essas estratégias, será possível demonstrar os vários cenários onde cada versão da *YOLO* se destaca, seus pontos positivos e negativos, conforme será observado nessa seção. Para o melhor entendimento da construção e do fluxo de desenvolvimento deste trabalho segue a Figura 12, apresentando o fluxograma que ilustra as etapas do desenvolvimento da metodologia utilizada neste trabalho.

Figura 12 – Fluxograma representando a metodologia proposta para esse trabalho



Autor (2024).

4.1 Versões da *YOLO* utilizadas no estudo

Para a detecção de objetos em imagens, foi escolhido o sistema *YOLO* devido à sua alta eficiência e velocidade em comparação com outros detectores. Para análise e visualização dos resultados, foram selecionadas as versões *YOLO*v8, *YOLO*v5 e *YOLO*v4, com o objetivo de avaliar as melhorias entre as gerações deste algoritmo. A escolha tem fundamento na cronologia e na estabilidade de cada versão. A *YOLO*v4 foi selecionada por ser uma das versões iniciais e mais estáveis ao rodas no ambiente do *Google Colab*, utilizando técnicas como *Bag of Freebies*, *Bag of Specials* e uma nova arquitetura com *Backbone*, *Neck* e *Head*. A *YOLO*v5, lançada pela *Ultralytics*, trouxe melhorias na arquitetura e novos recursos, como otimização de hiperparâmetros, monitoramento em tempo real e suporte à formatos populares como *ONNX* (*Open Neural Network Exchange*), permitindo sua execução em *CPU*s x86. Já a *YOLO*v8 introduz as tecnologias mais recentes, como melhorias de performance, técnicas de *Data Augmentation* e novas tarefas de Visão Computacional, incluindo classificação, detecção, segmentação, rastreamento e pose.

4.2 Construção da base de dados *(Dataset)*

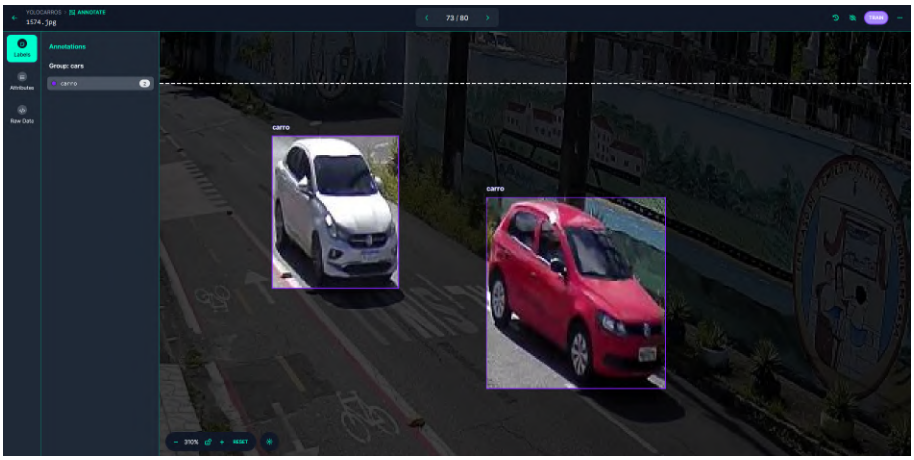
Na construção da base de dados para o modelo de detecção de objetos, foi necessário primeiramente adquirir um grande volume de imagens que contivessem os objetos dos quais serão detectados. Para este estudo conveio o uso de imagens de veículos automotivos, sendo aproximadamente 1160 imagens de trânsito, (de autoria própria, obtidas através de câmeras de seguranças) contendo cenários de iluminações variadas e diversos tipos de automóveis.

Seguindo as etapas de construção do *dataset*, antes de realizar qualquer rotulação (ou anotação) das imagens foi necessário fazer um pré-processamento dessas imagens para evitar ruído no *dataset*, as imagens que apresentaram avarias (falhas na imagem, borrão de movimento),

foram removidas. Estas avarias são caracterizadas como artefatos nas imagens devido ao processo de compressão das mesmas, e são facilmente detectadas no processo de eliminação manual.

No processo de inicialização do projeto, é preciso selecionar qual a modalidade de treino será realizada, tal qual o nome do *dataset* e as classes pertencentes a este *dataset*, após isto é dado início as anotações das imagens, que demanda bastante tempo por ser um processo manual, dependendo da quantidade de imagens e objetos em cada uma. A ferramenta escolhida foi a *Roboflow*, que permite anotações manuais ou automáticas. Optou-se por realizar o processo manualmente para evitar erros de identificação de objetos nas imagens como pode ser observado na Figura 13:

Figura 13 – Interface de anotação de imagens da *Roboflow*



Autor (2024).

Após realizada esta etapa é preciso exportar o *dataset*, para isso, foi necessário realizar algumas etapas até chegar à versão final do *dataset*. Primeiramente foi conveniente definir como seria dividido *(split)* o *dataset*. Neste estudo foi utilizado o *split hold-out* 80-20, que divide o número total de amostras em 80% para treino e 20% para teste e validação. Após o processo de separação, foi necessário redimensionar as imagens para 640x640 *pixels* com intuito de serem totalmente detectadas pelo filtro de processamento da *YOLO*. Logo após, foi feito o uso de uma técnica chamada *data augmentation*, esta técnica visa aumentar o número de amostras no *dataset*, com intuito de melhorar a assertividade do modelo treinado, neste estudo o único parâmetro utilizado foi de rotação horizontal, para diversificar a forma como o modelo "enxerga"os objetos no plano bidimensional. Como resultado final o *dataset* partiu de 1160 imagens para 1646 imagens.

Depois de todas essas etapas, finalmente é chegado o momento de exportar o *dataset*, para isso, é preciso selecionar o formato de exportação para ser usado no algoritmo, no caso da *YOLO* é necessário usar o formato *TXT* e para qual versão o *dataset* será utilizado. O dataset desenvolvido neste estudo está disponível através da *Roboflow*.

4.3 Conjunto de testes de generalização

Ao prosseguir com a preparação dos dados de pré-treinamento, foi necessário criar um segundo conjunto de imagens de testes. Este "conjunto de generalização"visa ampliar os testes do algoritmo e selecionar manualmente imagens de cenários que afetam a performance da *YOLO* com intuito te testar o algoritmo em cenários do mundo real, como baixa luminosidade, ruído e sobreposição, os quais reduzem a precisão e aumentam o tempo de processamento. Exemplo das imagens utilizadas neste conjunto de generalização na Figura 14:

Figura 14 – Algumas das imagens usadas nos diferentes conjunto do teste de generalização

Autor (2024).

Neste conjunto, assim como no dataset, as imagens foram reescaladas para 640x640 *pixels*, com intuito de padronizar os testes e manter a escala de acordo à *YOLO*, evitando falhas no resultado final da detecção dos objetos. O conjunto foi dividido em quatro partes: Imagens com Alta Iluminação, Baixa Iluminação, Ruído e Sobreposição, totalizando 46 imagens. Cada conjunto foi testado individualmente e no final de cada teste foi realizada uma média de precisão, gerando um valor total para cada um dos quatro conjuntos testados.

Como teste extra foi adicionado um conjunto contendo três vídeos com diferentes Quadros Por Segundo *(fps ou Frames Per Second)*, sendo: 30 *fps*, 60 *fps* e 120 *fps*. Neste caso o intuito é observar a integridade da *YOLO* em processar vídeos com alta taxa de quadros, criando um cenário mais realista, onde o modelo seria capaz de realizar a inferência em tempo real através de entradas externas, como: *webcams*, câmeras de celular ou câmeras de segurança.

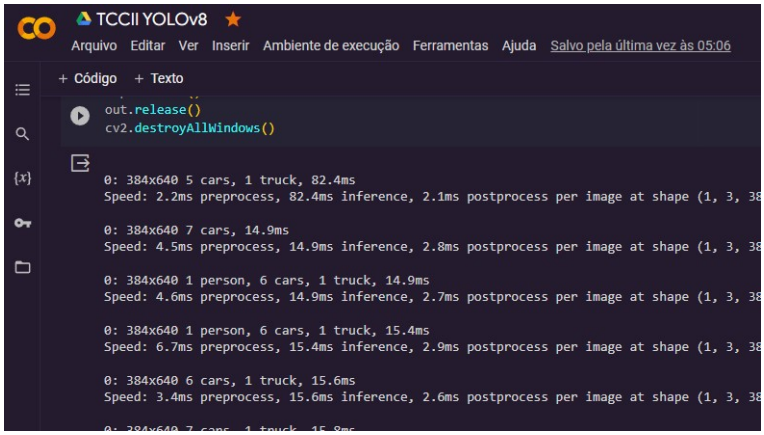
O objetivo final desde testes de generalização visa comparar numericamente se há diferenças significativas no processamento dessas imagens e vídeos nas diferentes versões da *YOLO* testadas neste estudo e concluir a respeito.

4.4 Aplicação do Treinamento

Após a construção do *dataset*, o próximo passo foi o treinamento do modelo proposto. Os modelos foram treinados e testados dentro do ambiente de execução no *Google Colab*, escolhido pela acessibilidade da plataforma, que facilita a execução dos códigos, a instalação das dependências necessárias (como *PyTorch*, *TensorFlow*, *OpenCV*, *CUDA* etc.) e oferece hardware de aceleração para IA (*GPU* e *RAM*) que reduz significativamente o tempo de treinamento

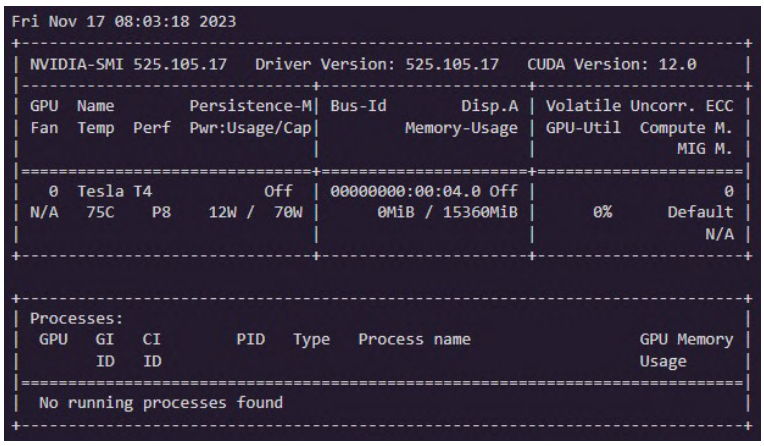
e inferência. No entanto, o principal ponto negativo do *Colab* é o limite de processamento, exigindo a assinatura do *Colab PRO* para treinamentos mais longos com mais épocas.

Figura 15 – Interface do *Google Colab*



Autor (2024).

Figura 16 – *GPU* usada no ambiente de execução



Autor (2024).

Prosseguindo com o treinamento do modelo, foi preciso indicar os argumentos, através da Interface de Linha de Comando *(CLI)*. Na tabela 2 é possível observar os parâmetros relevantes utilizados.

Tabela 2 – Argumentos da linha de comando

(a) Configuração de treino

Image size	*Patience*	*Cache*	*Batch*	*Epochs*
640 *pixels*	25 *Epochs*	*True*	-1	300 *Epochs*

Autor (2024).

A cerca dos argumentos utilizados, é preciso especificar cada um:

- *Image Size (img* ou *imgsz)* é a escala alvo da qual as imagens serão redimensionadas para o treinamento, a escala usada nesse treinamento foi de 640x640 (escala padrão usada na *YOLO* para não afetar a precisão e a complexidade computacional do modelo).

- *Patience* ou critério de parada, é usado para evitar *overfitting* (característica da qual o modelo acaba se enviesando ao conjunto de treino, sendo menos eficaz no teste e menos generalizado), o *patience* encerra o treinamento de forma antecipada, quando não ocorre melhoria significativa no erro *(loss)* da rede, este valor é definido em número de épocas, para todos os teste o valor foi fixado em 25 épocas por ter demonstrado melhores resultados.

- *Cache* habilita o armazenamento de algumas imagens durante o treinamento na memória *RAM*, reduzindo significativamente o tempo de processamento de cada época de treino, na configuração de treino o *cache* está ativado.

- *Batch* é uma técnica usada para dividir o treino em lotes e evitar a complexidade do treino com grandes quantidades de épocas, o *batch* está definido como -1, de acordo com a documentação obtida na *ultralytics*, este *batch size* define de forma automática a alocação de 60% do total de memória disponível da *GPU*, em todos os testes o valor não foi alterado.

- *Epochs* ou número de épocas, a quantidade de épocas pode variar muito de acordo com o que será treinado, nos testes realizados 300 épocas demonstrou ser um valor satisfatório para o treino da rede, portanto este foi o valor definido como padrão para todas as versões testadas.

Vale salientar que o modelo arquitetural da rede neural usado é baseado na *YOLO*s (*small*), quando foi possível selecionar o modelo. É possível acessar o código fonte usado para o treinamento das respectivas versões usadas neste estudo através do *Github*.

5 RESULTADOS

Esta seção destaca os principais resultados obtidos através de relatórios do desempenho da rede neural, com estes dados, é possível avaliar e compreender as capacidades dos modelos treinados e discorrer sobre os principais aspectos técnicos que diferenciam as versões do algoritmo testado. Também é importante citar alguns resultados significativos que este estudo proporciona, como: a construção de uma base de dados, estudo aprofundado sobre arquitetura e características da *YOLO* e avaliação de múltiplos cenários do mundo real na detecção de objetos.

Na tabela 3 é possível observar os resultados da melhor época de treinamento da rede, estes resultados foram obtidos através do *log* de saída do console ao final do treino. A melhor época de treinamento é armazenada em um arquivo denominado *best.pt* ou *best weights*.

Tabela 3 – Resultados do treinamento das redes

Treinamento YOLOv4		
Precision	Recall	mAP50-95
88.00%	77.00%	78.41%
Treinamento YOLOv5		
Precision	Recall	mAP50-95
97.87%	92.88%	86.35%
Treinamento YOLOv8		
Precision	Recall	mAP50-95
96.70%	94.68%	88.04%

Autor (2024).

Após observado os resultados do treinamento da rede, é preciso verificar o que foi obtido ao realizar os testes de generalização com cada modelo da *YOLO*. As tabelas 4 e 5 exibem os resultados do teste de generalização de todos os modelos treinados em suas respectivas versões:

Tabela 4 – Resultados de generalização com grupo de imagens

Imagens			
Alta iluminação	Precisão de Classificação	Detectados	Observados
YOLOv4	71.13%	29	47
YOLOv5	59.05%	20	47
YOLOv8	78.04%	22	47
Baixa iluminação	Precisão de Classificação	Detectados	Observados
YOLOv4	49.20%	20	32
YOLOv5	34.50%	2	32
YOLOv8	62.00%	4	32
Ruído	Precisão de Classificação	Detectados	Observados
YOLOv4	57.60%	5	11
YOLOv5	00.00%	0	11
YOLOv8	54.00%	1	11
Sobreposição	Precisão de Classificação	Detectados	Observados
YOLOv4	60.00%	6	18
YOLOv5	58.16%	6	18
YOLOv8	75.00%	3	18

Autor (2024).

Tabela 5 – Resultados de generalização dos testes com vídeos

Vídeos			
30fps	Precisão de Classificação	Detectados	Observados
YOLOv4	53.25%	4	6
YOLOv5	84.00%	1	6
YOLOv8	63.00%	2	6
60fps	Precisão de Classificação	Detectados	Observados
YOLOv4	53.75%	4	6
YOLOv5	83.00%	1	6
YOLOv8	61.00%	2	6
120fps	Precisão de Classificação	Detectados	Observados
YOLOv4	56.75%	4	6
YOLOv5	82.00%	1	6
YOLOv8	63.00%	2	6

Autor (2024).

Dando continuidade e contextualizando todos os resultados obtidos, durante o treinamento dos modelos, a *YOLO*v8 demonstrou ser a versão mais precisa considerando seu *mAP*50-95 (que determina a razão de precisão das caixa delimitadores) com 88.04%. Observando apenas o número de precisão dos modelos, a *YOLO*v5 é a melhor durante o treinamento. Porém, como discutido na seção 2, a precisão não reflete exatamente na qualidade de um modelo de detecção de objetos, sendo então o mAP50-95 a melhor métrica para definir o melhor modelo. Em detrimento disso, a *YOLO*v8 levou muito mais tempo para finalizar seu treinamento, levando aproximadamente 2h53min, enquanto a *YOLO*v5 levou aproximadamente 1h47min e a *YOLO*v4 aproximadamente 1h40min. Partindo para os cenários de generalização, analisando apenas os testes com o grupo de imagens, é possível notar que a *YOLO*v4 é o melhor entre os modelos testados pela capacidade de conseguir detectar mais objetos nos diferentes cenários. Este cenário se repete nos testes com vídeos, onde a *YOLO*v4 foi capaz de detectar mais objetos durante a inferência do que as demais versões testadas. Então, vale salientar que a grande diferença entre as versões mais recentes da *YOLO* é a quantidade de novas ferramentas (*tasks*) que ela disponibiliza, como: Classificação, Detecção, Segmentação, Rastreio e Pose. E principalmente a atualização constante das dependências necessárias para executar o algoritmo e a adição de novas ferramentas de detecção, o que possibilita que futuramente a *YOLO* seja executada em uma ampla gama de dispositivos e com mais funções.

Como considerações finais, apesar da *YOLO*v4 ter apresentado precisão inferior aos outros modelos nos testes, foi o único que se aproximou de detectar todos os objetos em cenários extremos, como baixa iluminação, presença de ruído ou sobreposição de objetos, fatores que reduzem a precisão. Contudo, tecnicamente, não houve diferença significativa no treinamento entre as versões testadas, com a *YOLO*v4 mostrando desempenho inferior, atingindo até 16,78% a menos no *Recall*. Conclui-se que a *YOLO*v5 e *YOLO*v8 são mais sensíveis ao dataset, necessitando maior variedade de ângulos, o que influencia na generalização do modelo.

6 CONCLUSÃO

O algoritmo de detecção de objetos *You Only Look Once (YOLO)* demonstrou ser bastante eficiente e rápido, como observado pelos resultados obtidos ao longo de suas interações. Com atualizações constantes, o *YOLO* se mantém entre as principais tecnologias de detecção de objetos e Visão Computacional, destacando-se mesmo em um cenário de rápido desenvolvimento de novos modelos de Inteligência Artificial. Sua acessibilidade é um diferencial, especialmente quando comparado a outros algoritmos, apesar da necessidade de uma *GPU* dedicada. No entanto, essa exigência é compensada pela eficiência e pela redução de custos em *hardware*, treinamento e tempo de inferência. A evolução contínua do *YOLO* reflete sua adaptabilidade às demandas do mercado, garantindo sua relevância e competitividade. A *YOLO*v8 se destaca como a versão mais eficiente durante o treinamento treinamento, adicionando novas funcionalidades, mas apresentou limitações nos testes de generalização, assim como a *YOLO*v5. A *YOLO*v4 oferece os melhores resultados em generalização e resultados medianos no treinamento geral, sendo então a melhor versão apresentada durante este estudo.

6.1 Trabalhos futuros

Dentre as linhas de pesquisa e desenvolvimento deste artigo, é possível desenvolver novos estudos, dentre eles, a expansão do conjunto de dados, com intuito de ampliar a capacidade dos algoritmos testados e realizar novos teste de generalização dos novos modelos treinados, proporcionando modelos mais robustos e que melhor se adaptem às condições de uso no mundo real. Para uma maior amplitude de dispositivos onde a *YOLO* possa ser usada é preciso testar uma maior variedade de plataformas gráficas e observar como o algoritmo se comporta, essas plataformas podem ser desde *GPU*, *APU* (Unidade de Processamento Acelerado), Gráficos Integrados ou até mesmo *NPU* (Unidades de Processamento Neural). A *NPU* é um grande destaque para a nova era da Inteligência Artificial, onde muito provavelmente através de uma delas seja possível o uso da *YOLO* de forma externa, sem a necessidade do uso de ambientes de execução robustos e com alta capacidade de hardware, possibilitando a alocação dentro de circuitos integrados de baixo consumo energético.

REFERÊNCIAS

ANDRADE, G. G. d. Uma proposta para detecção de objetos e estimação de distância em um simulador de veículos autônomos. 2022.

ARAÚJO, A. M. et al. Detecção e destaque em vídeo de objetos utilizando yolo. Universidade Federal da Paraíba, 2022.

ARAÚJO, F. H. et al. Redes neurais convolucionais com tensorflow: Teoria e prática. **SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. III Escola Regional de Informática do Piauí. Livro Anais-Artigos e Minicursos**, Sociedade Brasileira de Computação, v. 1, p. 382–406, 2017.

BELLMAN, R. An introduction to artificial intelligence: can computer think? San Francisco, Cal.: Boyd & Fraser,, 1978.

CIMIRRO, J. L. d. S. Reconhecimento de imagens: Uso do método yolo no reconhecimento de placas de trânsito. Universidade Federal do Pampa, 2022.

COELHO, J. V. d. A. B. et al. Aplicações e implicações da inteligência artificial no direito. 2017.

FACELI, K. et al. Inteligência artificial: uma abordagem de aprendizado de máquina. 2021.

GALVÃO, W. N. Segmentação de vasos sanguíneos utilizando redes neurais convolucionais: visualização e análise de correlação dos mapas de ativação. Universidade Federal de São Carlos, 2023.

GOMES, J. V. E. Detecção de objetos com a arquitetura yolo. 2022.

IMAMURA, M. E. et al. Detecção e reconhecimento de placas de licenciamento veicular em tempo real usando cnn. In: **Colloquium Exactarum. ISSN: 2178-8332**. [S.l.: s.n.], 2021. v. 13, n. 1, p. 89–99.

LAGE, F. d. C. A inteligência artificial na repercussão geral: análise e proposições da vanguarda de inovação tecnológica no poder judiciário brasileiro. 2023.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. **Neural computation**, MIT Press, v. 1, n. 4, p. 541–551, 1989.

LOPES, A. M. d. A. Estratégias de mediação para o ensino de matemática com objetos de aprendizagem acessíveis: um estudo de caso com alunos com deficiência visual. 2012.

NGUYEN, D. T. et al. A high-throughput and power-efficient fpga implementation of yolo cnn for object detection. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, IEEE, v. 27, n. 8, p. 1861–1873, 2019.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 779–788.

TAULLI, T. **Introdução à Inteligência Artificial: Uma abordagem não técnica**. [S.l.]: Novatec Editora, 2020.

TEIXEIRA, A. L. R. et al. Desenvolvimento de módulo de recursos lexicogramaticais baseado em regras para realização superficial em tarefas de geração de língua natural em português brasileiro. Universidade Federal de Minas Gerais, 2022.