Análise Comparativa de Algoritmos de Detecção de Objetos para Reconhecimento de Buracos em Vias e Estradas

Luiz Fernando Mello
luizf.mello@hotmail.com
Centro Universitário Avantis - UNIAVAN
Balneário Camboriú, Santa Catarina
Brasil

Evandro L. Viapiana
evandro.luis.viapiana@gmail.com
Centro Universitário Avantis - UNIAVAN
Balneário Camboriú, Santa Catarina
Brasil

Luiz Fernando M. Arruda
eng.luizarruda@gmail.com
Centro Universitário Avantis - UNIAVAN
Balneário Camboriú, Santa Catarina
Brasil

ABSTRACT

This work presents a comparative analysis using the mAP metric of the YOLOv7 and SSD algorithms applied to the field of computer vision. The comparison is performed through the detection of potholes on highways and roads. In works defended by other authors, both techniques are considered for applications in computer vision. In this study, both models achieved promising results in pothole detection; however, it is observed that the YOLOv7 model reached an mAP metric of 80%, while the SSD model obtained an mAP of 73%. Thus, the results indicate that YOLOv7 is more efficient and accurate in pothole detection in this specific context.

KEYWORDS

Single Shot Multibox Detector, Pothole Detection, SSD, Computer Vision, You Only Look Once, YOLOv7

1 INTRODUÇÃO

Acidentes em estradas e rodovias ocorrem cotidianamente. Alguns deles são ocasionados por buracos ou irregularidades que surgem do desgaste das rodovias ou de eventos meteorológicos naturais, como as chuvas. Esses buracos, quando atingidos por um veículo, podem causar diversos transtornos aos motoristas, danificando o veículo e/ou resultando em acidentes como colisões e atropelamentos, inclusive acidentes fatais [1].

Segundo a base de dados aberta da Polícia Rodoviária Federal (PRF) [2], no ano de 2021, ocorreram 3.974 acidentes decorrentes de buracos nas rodovias (Fig. 1). Destes, 15,10% resultaram em vítimas fatais e 72,45% em vítimas feridas. No ano de 2022, observou-se um aumento de 40% em relação a 2021 nos acidentes relacionados a buracos, dos quais 17,11% resultaram em vítimas fatais e 71,73% em vítimas feridas.

Figura 1: Acidente por buraco em vias públicas [3]

Efetuando uma análise dos acidentes registrados até o final de fevereiro de 2023, observa-se que 1.312 foram ocasionados por buracos nas rodovias, dos quais 21,42% resultaram em vítimas fatais e 70,50% em vítimas feridas. Desta forma, os dados apontam para a necessidade de adoção de medidas urgentes para mitigar esse problema e garantir a segurança dos motoristas e pedestres [2].

Uma estratégia para reduzir o número de acidentes é a implementação de sistemas autônomos inteligentes para detecção de objetos. Esses sistemas têm a capacidade de abordar problemas complexos, como o reconhecimento de buracos, sem depender da interação humana, utilizando periféricos de entrada de informação, como sistemas de imagem/vídeo. Essa abordagem é comumente referida na literatura como visão computacional ou sistemas de visão assistida por computador [4].

Figura 2: Visão Computacional [5]

A visão computacional é uma subárea da inteligência artificial que tem como objetivo modelar e replicar a visão humana nos computadores por meio de *software* e *hardware*, conforme demostrado na Fig. 2. Estes sistemas extraem informações significativas de imagens, vídeos e outras entradas visuais, com o propósito de fornecer dados para facilitar a tomada de decisões ou automatizar tarefas específicas [4].

Na literatura existem diversos algoritmos que são empregados para o reconhecimento de objetos, incluindo o *Single Shot Multibox Detector* (SSD) [6] e o *You Only Look Once* (YOLO) [7], que pertencem à categoria de algoritmos de estágio único. Essas abordagens são discutidas em detalhes nos estudos de [8–13].

Com o intuito de avaliar adequadamente o desempenho dos algoritmos, torna-se essencial selecionar uma métrica de avaliação. A métrica *Mean Average Precision* (mAP) combina de forma ponderada outras três métricas relevantes: *Precision*, *Recall* e *Intersection Over Union* (IoU). Estas são amplamente reconhecidas no campo da aprendizagem de máquina, o que faz da mAP uma boa escolha para avaliação dos algoritmos [14].

2 ALGORITMOS DE DETECÇÃO DE OBJETOS

A detecção de objetos é considerada uma tarefa clássica na área de visão computacional, que tem como objetivo classificar e localizar um ou mais objetos em uma imagem estática (foto) ou dinâmica (vídeo) [15]. Ao contrário da tarefa de classificação, que atribui uma classe à imagem como um todo, na detecção de objetos, destaca-se o objeto específico por meio de uma caixa delimitadora (Fig. 3).

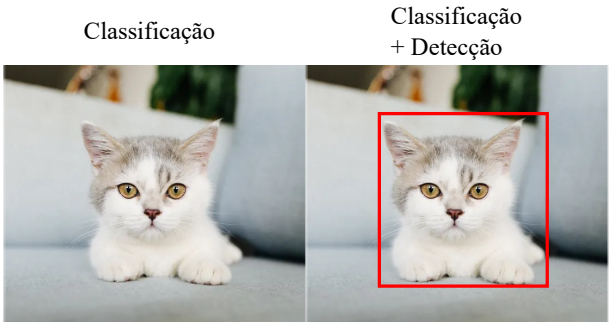


Figura 3: Diferença entre Classificação e Detecção de Objetos

Embora existam técnicas propostas há mais de 50 anos para resolver problemas relacionados à visão computacional, como, por exemplo, a detecção de objetos por meio do detecção de bordas (Fig. 4), conforme proposto por [16], essas técnicas não são completamente adequadas para lidar com os desafios presentes em problemas do mundo real. Consequentemente, ao longo dos anos, novos algoritmos vem sendo propostos com o intuito de abordar de forma mais eficiente essa tarefa.

Com a evolução dos *hardwares* ao longo do tempo e a explosão na quantidade de dados gerados em grande velocidade, volume e variedade, tornou-se possível a utilização de algoritmos de redes neurais artificiais que desempenham um papel crucial nas técnicas atuais de baseadas em inteligência artificial para detecção de objetos [18] nas quais pode-se destacar o SSD e o YOLO.

2.1 *Single Shot MultiBox Detector (SSD)*

O algoritmo SSD, introduzido por [6], é reconhecido por sua capacidade de realizar simultaneamente as tarefas de localização e classificação de objetos, onde o termo *"Single Shot"* refere-se à habilidade do modelo de realizar ambas as tarefas em uma única passagem pela rede. Além disso, o nome *"Multibox"* faz alusão à técnica de regressão de caixa delimitadora incorporada ao SSD, que propõe rapidamente as coordenadas das caixas delimitadoras, independentemente da classe do objeto [19]. Esse modelo alcança uma pontuação de 74,3% de mAP no conjunto de dados Pascal VOC2007 [6].

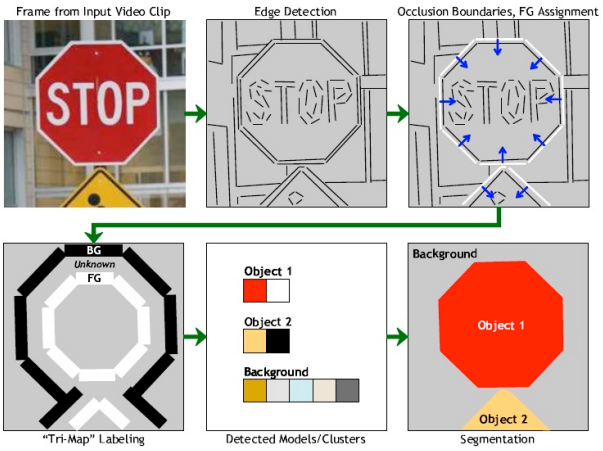
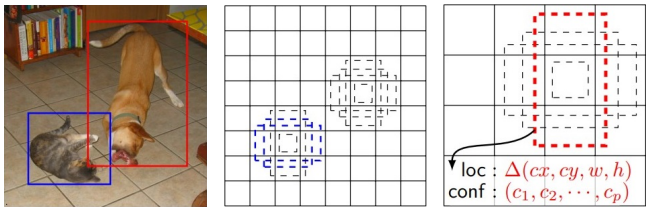


Figura 4: Usando limites para segmentação e reconhecimento [17]

A detecção de objetos no modelo SSD é dividida em duas etapas: extração de mapas de recursos e aplicação de filtros convolucionais para detecção de objetos [19].

Na primeira etapa do algoritmo SSD, são geradas caixas delimitadoras de tamanho fixo, também conhecidas como *Default Boxes*. As dimensões dessas caixas delimitadoras são pré-calculadas levando em consideração as dimensões e localizações das caixas delimitadoras de referência correspondentes a cada classe no conjunto de dados [20].



(a) Fig. com caixa padrão **(b) 8 x 8 Mapa de recursos** **(c) 4 x 4 Mapa de recursos**

Figura 5: Caixas padrão da Imagem de Entrada [21]

A Fig. 5 ilustra o funcionamento das caixas delimitadoras padrão em uma imagem de entrada. A Fig. 5a exibe as caixas delimitadoras de referência. É necessário gerar mapas de características que descrevem a imagem original em células, conforme as imagens Fig. 5b e Fig. 5c.

A camada inicial da rede neural no algoritmo SSD consiste da utilização de uma rede pré-treinada, tais como: VGG (Fig. 6), ResNet ou MobileNet. Essas redes são comumente empregadas em tarefas de classificação e já foram treinadas para extrair mapas de características [20].

A arquitetura VGG apresenta uma camada totalmente conectada no final da sua estrutura. No entanto, ao lidar com problemas de detecção de objetos, as camadas totalmente conectadas são removidas da arquitetura, utilizando a VGG16 exclusivamente para a

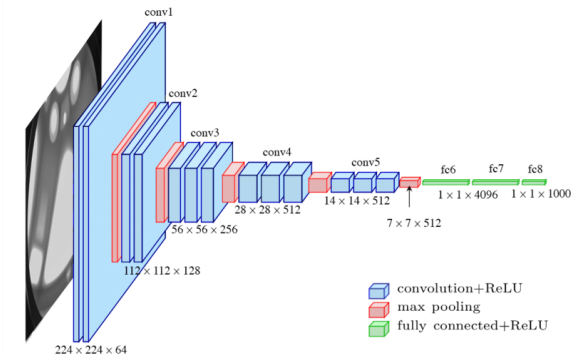


Figura 6: Arquitetura padrão da rede VGG 16 [22]

extração de características. Essa rede de extração de características é conhecida como rede base [23].

A classe *MultiBoxLoss* (Fig. 7) é responsável pela implementação detalhada da função de perda no algoritmo SSD. Para garantir a fidelidade e precisão do algoritmo, utilizou-se a função de perda descrita no artigo original do SSD como referência.

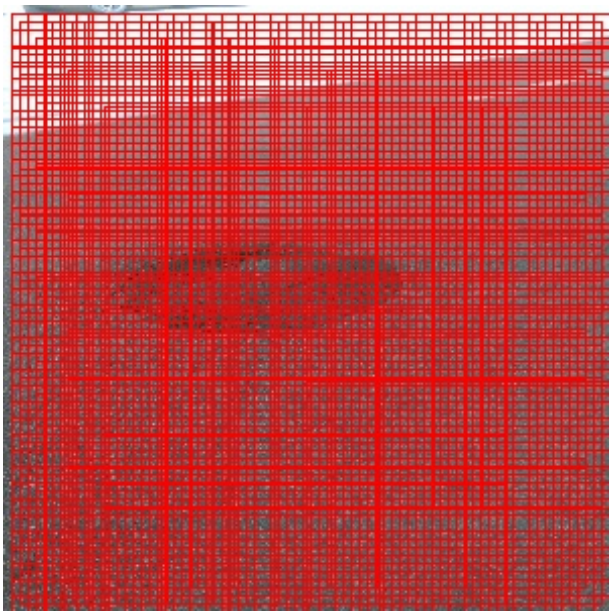


Figura 7: Aplicação do Multibox

O resultado da rede base é fornecido como entrada para seis camadas de convolução, onde os mapas de características têm tamanhos progressivamente menores, resultando em uma resolução decrescente. Esse processo é realizado com o objetivo de detectar objetos em diferentes escalas [23], conforme ilustrado na Fig. 8.

Em cada mapa de características, são definidas caixas de detecção padrão, nas quais a rede calcula a probabilidade de um objeto estar presente, bem como sua variação de posição, largura e altura [6]. Para realizar a detecção das classes e suas respectivas caixas

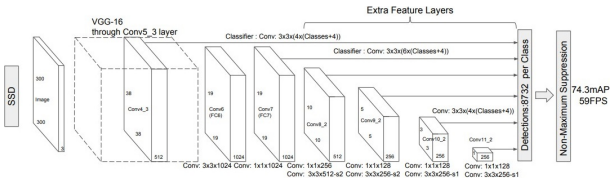


Figura 8: Arquitetura do algoritmo SSD

delimitadoras, o algoritmo utiliza convoluções no mapa de características com filtros de tamanho 3x3xMxN, onde M é a profundidade da camada e N é o número de filtros aplicados. A camada de convolução possui um valor de *stride* e *pad* iguais, garantindo que o resultado da convolução tenha a mesma largura e altura da camada de entrada [23].

O número de filtros aplicados para detectar K classes em um mapa de características que possui C caixas de detecção padrão é dado por $(K + 4) \cdot C$. Isso significa que são utilizados $K + 4$ filtros, sendo um para representar a presença de cada classe e mais quatro para representar as variações na caixa delimitadora, para cada uma das C caixas presentes no mapa [23].

Todas as detecções obtidas em todas as células das camadas de convolução são submetidas a uma camada de *Non Maximum Suppression* (NMS), que tem como objetivo remover detecções sobrepostas e manter apenas uma caixa delimitadora por objeto presente na imagem. O método NMS utiliza o critério *Intersection over Union* (IoU), calculado como a porcentagem de interseção entre duas áreas dividida pela união das áreas das duas figuras geométricas. Esse critério é empregado para determinar se uma detecção deve ser removida ou mantida durante o processo de supressão [23].

A função de perda do SSD é composta por duas partes: perda de classificação e perda de localização. A perda de classificação mensura a diferença entre as previsões de classe feitas pelo modelo e as classes reais dos objetos nas imagens. Essa diferença é calculada usando a função de perda de entropia cruzada. Por outro lado, a perda de localização avalia a diferença entre as previsões de caixas delimitadoras feitas pelo modelo e as caixas delimitadoras de referência (*ground truth*). Para calcular essa perda, é empregada a *Smooth L1 Loss* [6].

2.2 *You Only Look Once* - YOLO

O YOLO é uma família de algoritmos de detecção de objetos em tempo real de estágio único que utiliza de redes neurais convolucionais (CNN) para detecção de objetos. O objetivo da rede é realizar a detecção de objetos, abordando essa tarefa como um problema de regressão. Dessa forma, uma CNN pode prever caixas delimitadoras e a confiança da classe [12].

A arquitetura escolhida para ser utilizada no presente trabalho foi a YOLOv7. No momento da elaboração deste trabalho, sua arquitetura representa o estado da arte para detecção de objetos em imagens estáticas e dinâmicas [7].

Segundo [7], a YOLOv7 é a detectora de objetos em tempo real mais rápida e precisa até hoje. O algoritmo estabeleceu uma melhora significativa para os algoritmos da família YOLO ao aumentar seu desempenho.

O aprimoramento na velocidade e precisão são alcançados através da implementação de reformas arquitetônicas significativas. Entre essas reformas, destacam-se a adoção da técnica *Extended Efficient Layer Aggregation* (E-ELAN), o ajuste dimensional para modelos que se baseiam na concatenação e a reparametrização. Essas mudanças são estratégicas para alcançar um equilíbrio ótimo entre eficiência e precisão na detecção dos objetos[24]

O algoritmo YOLOv7 consiste em quatro principais módulos: o módulo de *input*, a rede *backbone*, a rede *Head* e a rede *detection*, conforme mostrado na Fig. 9 [24].

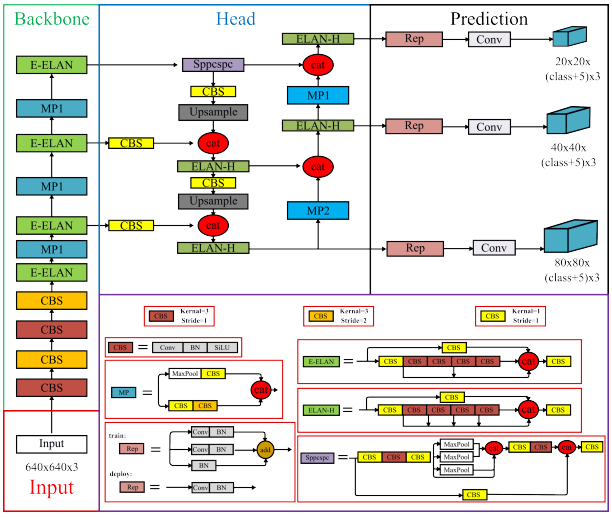


Figura 9: Adaptado de [24] Arquitetura da rede YOLOv7

O **Módulo de entrada** tem como finalidade garantir que as imagens de entrada sejam redimensionadas uniformemente para o tamanho de 640x640 e atendam aos requisitos de entrada da rede principal. A etapa de pré-processamento do YOLOv7 utiliza técnicas de aprimoramento de dados, como mosaico e híbrido, e se baseia no método de cálculo de quadro de âncora adaptativo estabelecido pelo YOLOv5 [24].

A **Rede *Backbone*** é composta por três componentes principais: CBS, E-ELAN e MP1. O CBS é formado por funções de convolução, normalização em lote e ativação SiLU. O E-ELAN preserva a arquitetura de design original ELAN e melhora a capacidade de aprendizado da rede ao orientar diferentes blocos computacionais de grupos de características a aprender características mais diversas, preservando o caminho original do gradiente [24].

Já o MP1 é formado por CBS e MaxPool e é dividido em dois caminhos: o caminho superior e o caminho inferior. No caminho superior, o MaxPool é usado para reduzir pela metade o comprimento e a largura da imagem, enquanto o CBS, com 128 canais de saída, reduz pela metade os canais da imagem [24].

No caminho inferior, um CBS com um *kernel* e *stride* de 1x1 é usado para reduzir pela metade os canais da imagem, e outro CBS com *kernel* 3x3 e *stride* 2x2 é usado para reduzir pela metade o comprimento e a largura da imagem. As características extraídas de ambos os caminhos são então combinadas por meio de uma operação de concatenação (Cat) [24].

O *MaxPool* tem como objetivo extrair informações do valor máximo dos *pixels* em pequenas áreas locais, enquanto o CBS extrai todas as informações de valor em pequenas áreas locais, melhorando assim a capacidade de extração de características da rede [24].

A ***Rede Head*** é estruturada utilizando a arquitetura *Feature Pyramid Network* (FPN), que emprega o *design* PANet. A rede é composta por vários blocos de convolução, normalização em lote e ativação SiLU (no bloco CBS), juntamente com a introdução de uma estrutura de *Spatial Pyramid Pooling and Convolutional Spatial Pyramid* (SPPESPE), a rede E-ELAN e o MaxPool-2 (MP2) [24].

A estrutura SPPESPE melhora o campo perceptual da rede ao incorporar uma estrutura de *Convolutional Spatial Pyramid* (CSP) dentro da estrutura de *Spatial Pyramid Pooling* (SPP), junto com uma borda residual ampla para auxiliar na otimização e extração de características[24].

A camada ELAN-H é uma fusão de várias camadas de características baseadas em E-ELAN, aprimorando ainda mais a extração de características. O bloco MP2 possui uma estrutura semelhante ao bloco MP1, com uma pequena modificação no número de canais de saída [24].

A **Rede *Prediction*** utiliza uma estrutura denominada REP para ajustar o número de canais de características da imagem gerada pela rede *Head* e é aplicada uma convolução 1x1 para realizar as previsões de confiança, classe e caixas delimitadoras. A estrutura REP é inspirada no RepVGG [25] e apresenta um design residual para auxiliar no processo de treinamento.

Essa estrutura residual pode ser simplificada para uma convolução simples na etapa de inferência, reduzindo assim a complexidade da rede sem comprometer seu desempenho [24]. Após as previsões das caixas delimitadores, o algoritmo YOLO utiliza o mesmo pós-processamento do algoritmo SSD, o *non-maximum suppression* (NMS).

Na etapa de pós-processamento, o algoritmo YOLO emprega a técnica NMS (*non-maximum suppression*), a mesma utilizada pelo algoritmo SSD, eliminando a sobreposição e duplicação de detecções durante o processo de detecção de objetos em uma imagem.

2.3 Métricas de Avaliação

Para avaliar os modelos de detecção de objetos são utilizadas métricas de avaliação, considerando o valor verdadeiro da localização do objeto em uma imagem (*ground-truth*) e a predição feita pelo modelo, sendo possível calcular a precisão do modelo. Uma das métricas mais utilizadas para esse fim é o *Mean Average Precision* (mAP). O mAP é a combinação de três métricas: *precision*, *recall* e *Intersection Over Union* (IoU) [26].

2.3.1 Precision. A métrica de precisão é a relação entre os verdadeiros positivos (V_P) e o total de previsões classificadas como positivas para o modelo, sendo essas previsões verdadeiros positivos e falsos positivos (F_P). Portanto, a precisão indica a porcentagem dos resultados que são relevantes [12].

$$Precision = \frac{V_P}{V_P + F_P} \tag{1}$$

2.3.2 Recall. A métrica de *recall* é a relação entre o V_P e a soma dos V_P com o falso negativo (F_N). Assim, o *recall* é a capacidade de

um modelo de encontrar todos os pontos de dados de interese em um conjunto de dados, ou seja, a porcentagem do total de resultados relevantes classificados corretamente por seu algoritmo [12].

$$Recall = \frac{V_P}{V_P + F_N} \quad (2)$$

2.3.3 Intersection Over Union. A métrica de avaliação IoU é frequentemente utilizada para medir a qualidade de um modelo de detecção de objeto. As imagens que compõe o conjunto de dados são previamente rotuladas especificando onde estão os objetos na imagem. Assim, é possível avaliar o desempenho do modelo utilizando a métrica IoU. A métrica é calculada verificando a interseção entre a caixa delimitadora da detecção e do objeto previamente rotulado, pela área total de ambas as caixas delimitadoras, conforme a Fig. 10 [12].

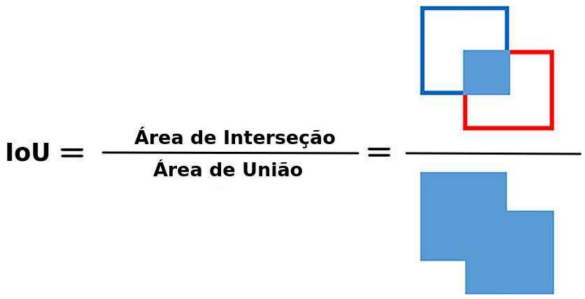


Figura 10: Equação da métrica IoU [12]

A métrica de IoU produz um resultado que varia de 0 a 1, sendo que um valor maior indica um melhor desempenho do modelo [12].

2.3.4 Mean Average Precision. A métrica mAP é a média das precisões médias (*Average Precision*) calculada para cada classe de objeto em um problema de detecção de objetos. É uma métrica amplamente utilizada para avaliar o desempenho de um sistema de detecção de objetos. Essa métrica varia de 0 a 1 (100%), sendo que valores mais próximos de 1 indicam um resultado melhor obtido pelo sistema [27].

A *Average Precision* (AP) é calculada como a área sob a curva que representa a relação entre *Precision* e *Recall*. No caso da AP, é necessário estabelecer uma condição para que a *Precision* seja considerada correta. Essa condição é obtida se a métrica IoU for igual ou superior a um valor definido, também conhecido como limite (*threshold*) [27].

Por exemplo, suponha que um *threshold* de 0,5 seja pré-definido para o IoU ao classificar uma previsão. Se o valor do IoU for maior ou igual a 0,5, a detecção do objeto é classificada como VP. Um VP ocorre quando o detector prevê corretamente as classes. Por outro lado, se o valor do IoU for menor que 0,5, indica uma detecção incorreta, sendo então classificada como FP. Um FP é considerado um erro do detector e ocorre quando o classificador comete um equívoco na previsão.

Um Verdadeiro Negativo (VN) ocorre quando o detector prevê corretamente que não há objeto de interesse na região em questão e de fato não há. Por outro lado, um FN, também classificado como um erro do detector, ocorre quando o classificador prevê que não há objeto de interesse na região, mas na realidade há um objeto presente [28].

3 APLICAÇÃO

Este capítulo apresenta a metodologia adotada para o desenvolvimento do trabalho, sendo dividido em três partes principais: Aquisição da Base de Dados, Treinamento do Algoritmo SSD e Treinamento do Algoritmo YOLO. A implementação completa da análise comparativa está disponível no GitHub [1].

3.1 Aquisição da Base de Dados

Para realizar a análise comparativa dos modelos SSD e YOLO, foi coletada uma base de dados real a partir do portal de competições de ciência de dados Kaggle [2].

A escolha de utilizar uma base de dados real é fundamental para avaliar o desempenho e a eficácia dos modelos SSD e YOLO em condições reais de aplicação. Essa base de dados específica disponibilizada no Kaggle apresenta características relevantes para a análise comparativa dos modelos, fornecendo um conjunto diversificado de exemplos para treinamento e teste.

O conjunto de dados utilizado é composto por uma coleção de 665 imagens de buracos em rodovias e estradas, sendo cada imagem acompanhada de suas respectivas caixas delimitadoras anotadas. As anotações seguem o formato amplamente conhecido na comunidade de visão computacional, o PascalVoc, garantindo uma estrutura padronizada e de fácil compreensão. Esses dados já estão divididos em conjunto de treinamento e teste.

No total, 80% das imagens foram destinadas ao conjunto de treinamento, permitindo treinar os modelos utilizados neste trabalho. Os 20% restantes foram reservados para o conjunto de teste, usado para avaliar o desempenho dos modelos e medir sua capacidade de generalização em relação a dados não vistos anteriormente.

Essa separação de conjuntos de dados é uma prática comum em aprendizado de máquina e visão computacional, garantindo uma avaliação objetiva do desempenho dos modelos. A partir dessas avaliações é possível obter métricas precisas, como mAP e IoU, que são essenciais para avaliar a eficácia e a confiabilidade dos modelos na detecção de buracos em rodovias e estradas.

3.2 Treinamento do Algoritmo SSD

Para realizar a codificação do algoritmo SSD, foi feito uso do artigo original do algoritmo como fonte de referência, além de utilizar o repositório do Wei Liu hospedado no GitHub[3].

A implementação do algoritmo seguiu o paradigma de programação orientada a objetos, usando classes como estrutura fundamental. Todos os arquivos implementados estão disponíveis no repositório [1], possibilitando acesso e revisão do código-fonte.

Após a implementação do algoritmo SSD, dá-se início ao treinamento do modelo utilizando o conjunto de dados de buracos em rodovias representado na Tabela 1. Durante o treinamento do algoritmo não foi utilizado a técnica de *augmentation* para criar dados sintéticos. No entanto, foi adotada a técnica de *transfer learning* para aproveitar os pesos pré-treinados da rede VGG-16, treinada

[1] https://github.com/luizfmello01/tcc_ssd_yolo
[2] https://www.kaggle.com/datasets/chitholian/annotated-potholes-dataset
[3] https://github.com/weiliu89/caffe/tree/ssd

Tabela 1: Treinamento do algoritmo SSD

Iniciando os treinamentos				
Alterando a taxa de aprendizado para 0.0001				
Epoch [0]	batch_idx [0]	loc_loss [2.65]	cls_loss [7.44]	total_loss [10.09]
Epoch [0]	batch_idx [1]	loc_loss [2.86]	cls_loss [7.09]	total_loss [9.96]
Epoch [0]	batch_idx [2]	loc_loss [2.92]	cls_loss [6.83]	total_loss [9.75]
Epoch [0]	batch_idx [3]	loc_loss [2.93]	cls_loss [6.53]	total_loss [9.47]
Epoch [0]	batch_idx [4]	loc_loss [2.92]	cls_loss [6.29]	total_loss [9.21]
Epoch [0]	batch_idx [5]	loc_loss [2.88]	cls_loss [6.04]	total_loss [8.92]
Epoch [0]	batch_idx [6]	loc_loss [2.90]	cls_loss [5.81]	total_loss [8.71]
Epoch [0]	batch_idx [7]	loc_loss [2.87]	cls_loss [5.61]	total_loss [8.48]
Epoch [0]	batch_idx [8]	loc_loss [2.87]	cls_loss [5.41]	total_loss [8.28]
Epoch [0]	batch_idx [9]	loc_loss [2.87]	cls_loss [5.23]	total_loss [8.10]
Epoch [0]	batch_idx [10]	loc_loss [2.87]	cls_loss [5.07]	total_loss [7.94]

sobre o conjunto de dados ImageNet. O modelo treinado será posteriormente avaliado através da inferência em novas imagens que não foram vistas pelo mesmo.

3.3 Treinamento do Algoritmo YOLO

A primeira etapa para iniciar o processo de treinamento do algoritmo YOLOv7 é realizar a clonagem do repositório oficial do YOLOv7 disponibilizado no GitHub[4]. Essa ação consiste em criar uma cópia exata de todos os arquivos e diretórios presentes no repositório, possibilitando o acesso aos recursos necessários para o treinamento.

O repositório oficial do YOLOv7 foi treinado utilizando o dataset MS Coco, cujo formato dos arquivos difere do formato dos dados presentes em base de dados. Para superar essa diferença, desenvolveu-se dois *notebooks* específicos para converter o conjunto de dados do formato PascalVoc para o formato suportado pelo YOLOv7.

No processo de adaptação da estrutura para o formato YOLO, é necessário realizar a cópia dos diretórios *ImageSets*, *Annotations* e *JPEGImages* presentes na estrutura padrão do PascalVoc para uma nova pasta. Essa nova pasta possui uma estrutura simplificada contendo apenas os diretórios "train/images", "train/annotations", "test/images"e "test/annotations". Essa simplificação visa facilitar a transformação dos dados para o formato YOLO.

Após a execução do *notebook* mencionado anteriormente, as imagens resultantes foram organizadas nos diretórios "train/images"e "test/images", enquanto as anotações dos objetos foram agrupadas nos diretórios "train/annotations"e "test/annotations". Essa estrutura de diretórios separa adequadamente as imagens e as anotações de acordo com as divisões de treinamento e teste.

No próximo passo do processo é necessário converter as anotações dos objetos do formato padrão do PascalVoc para o formato YOLO. Para realizar essa transformação em cada arquivo XML individualmente, desenvolveu-se um *notebook* separado. Esse *notebook* é extenso e abrange todos os passos necessários para realizar a conversão de forma adequada, incluindo a leitura dos arquivos XML, a extração das informações relevantes, como as coordenadas dos objetos anotados, e a geração dos arquivos de texto no formato YOLO correspondentes.

No processo de treinamento do algoritmo YOLO, foi utilizada a técnica de augmentation para criar dados sintéticos e foi adotado o transfer learning para aproveitar os pesos pré-treinados no conjunto de dados MS Coco. A Tabela 2 apresenta o início ao treinamento com o algoritmo YOLOv7 utilizando o conjunto de dados de buracos em rodovias.

[4]https://github.com/WongKinYiu/yolov7

Tabela 2: Treinamento do algoritmo YOLOv7

Epoch	Gpu_Mem	Box	Obj	Cls	Total	Labels	Img Size	mAP@.5
0/299	9.93G	0.089	0.008264	0	0.097	24	320	0.002
1/299	9.97G	0.082	0.008746	0	0.091	34	320	0.008
2/299	9.97G	0.079	0.008431	0	0.087	9	320	0.009
3/299	9.97G	0.074	0.009733	0	0.084	32	320	0.007
4/299	9.93G	0.071	0.009912	0	0.081	45	320	0.001

4 RESULTADOS

Para avaliar o modelo, foi utilizado um treinamento que se consistiu em um total de 300 épocas. Além disso, consideraram-se 133 imagens em ambos os modelos durante a avaliação. Vale ressaltar que o conjunto de dados utilizado abrange exclusivamente a classe de buracos.

Tabela 3: Resultado da avaliação dos modelos

Algoritmo	Épocas	Total de Imagens	mAP
SSD	300	133	0,736
YOLOv7	300	133	0,806

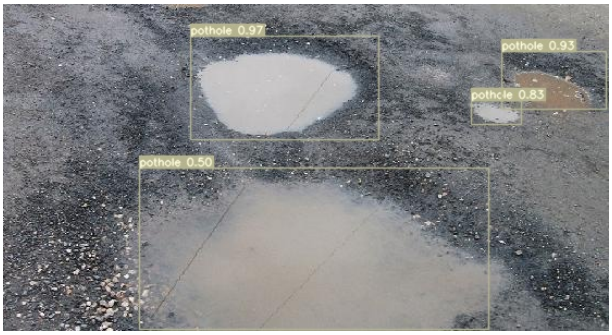
A Tabela 3 apresenta os valores do modelo avaliado, incluindo a quantidade de épocas de treinamento, o total de imagens usadas na fase de avaliação e a métrica de avaliação conhecida como mAP, que foi obtida durante a análise. Após realizar a avaliação, os resultados indicaram que o desempenho do algoritmo YOLOv7 superou o do SSD em relação à métrica mAP.

4.1 Detecção de Buracos nas imagens do conjunto de testes

Nas figuras 11a e 11b, o algoritmo YOLO identificou um total de 6 buracos na rodovia. No entanto na Fig.a 11b, o modelo classificou erroneamente um objeto como buraco, quando na verdade se trata de um pequeno desnível na via. Apesar desse equívoco, é importante ressaltar que, de maneira geral, o desempenho do algoritmo na detecção de buracos foi satisfatório.

Por outro lado, nas figuras 12a e 12b, é possível afirmar que o algoritmo SSD apresentou um desempenho inferior na detecção de buracos, quando comparado ao Yolov7, identificando um total de 3 buracos nas imagens analisadas, deixando de detectar 2 buracos presentes nas rodovias.

Portanto, fica evidente que ambos os modelos obtiveram um sucesso na detecção de buracos, em figuras não vistas previamente durante o processo de treinamento, o que ressalta a habilidade dos modelos em aprenderem efetivamente o padrão de detecção de buracos.

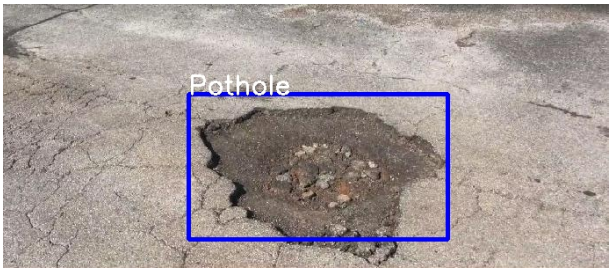


(a) Detecção de buracos img-449 utilizando YoloV7

(b) Detecção de buracos img-464 utilizando YoloV7

Figura 11: Análise de imagens do conjunto de testes usando YoloV7

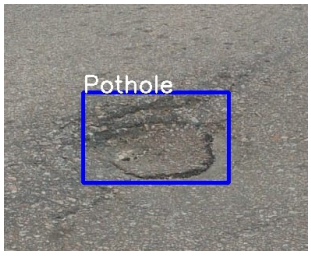
(a) Detecção de buracos img-449 utilizando *Single Shot Detection*



(b) Detecção de buracos img-464 utilizando *Single Shot Detection*

Figura 12: Análise de imagens do conjunto de testes usando *Single Shot Detection*

4.2 Detecção de Buracos em novas imagens capturadas pelo o autor através de aparelho telefônico



(a) Análise de Fig. utilizando o *Single Shot Detection*

(b) Análise de Fig. utilizando o *Single Shot Detection*

Figura 13: Detecção de Buracos em novas imagens capturadas pelo o autor através de aparelho telefônico

Em busca de observar a generalização do modelo, adquiram-se 4 novas figuras (13a, 13b, 14a e 14b) na região de Balneário Camboriú, e observa-se que ambos os algoritmos detectaram os buracos nas vias, conforme as figuras 13 e 14.

(a) Análise de Fig. utilizando o Yolo

(b) Análise de Fig. utilizando o Yolo

Figura 14: Detecção de Buracos em novas imagens capturadas pelo o autor através de aparelho telefônico

5 CONCLUSÕES

Com base nos resultados obtidos, pode-se concluir que ambos os modelos desempenharam um bom trabalho na detecção de buracos nas rodovias. O modelo YOLOv7, em particular, demonstrou um desempenho superior ao modelo SSD, conforme evidenciado pela métrica mAP, com o YOLOv7 alcançando um mAP de 80% em comparação aos 73% do SSD. Esta diferença pode ser atribuída à eficácia do YOLOv7 em lidar com variações, como escala, iluminação, ocultação e outros. Estes são desafios comuns na detecção de objetos em ambientes dinâmicos, complexos e não controlados, como é o caso de rodovias.

Entretanto, é importante observar que ambos os modelos possuem potencial para melhorar sua precisão. A utilização de um conjunto de treinamento maior e mais variado pode proporcionar

aos modelos a capacidade de aprender com uma maior variedade de exemplos de buracos, incluindo buracos com características menos comuns. Além disso, a implementação de técnicas avançadas, como transformações geométricas e variações de iluminação, que não foi abordado neste artigo, deve enriquecer o conjunto de treinamento sem a necessidade de coletar mais imagens manualmente.

Outro aspecto que merece atenção é a otimização dos parâmetros dos modelos. Ajustes finos nos parâmetros, como a taxa de aprendizado, o tamanho do lote e o número de épocas, podem ter um impacto significativo no desempenho dos modelos. Testes com diferentes arquiteturas de rede e mecanismos de atenção também podem revelar melhorias na precisão da detecção dos buracos.

Em suma, os resultados atuais são promissores, mas há espaço para aprimoramento. Pesquisas futuras focadas na expansão do conjunto de dados, nos testes com técnicas de aumento de dados e na otimização de parâmetros têm o potencial de melhorar o desempenho desses modelos de detecção de buracos nas rodovias.

REFERÊNCIAS

[1] Ricardo Silveira Rodrigues. Detecção de buracos em estradas: uma abordagem automatizada baseada na transformada wavelet de haar. Master's thesis, UNIVERSIDADE FEDERAL DE SANTA MARIA, Santa Maria, RS, June 2019.

[2] PRF. Documento csv de acidentes 2023 (agrupados por pessoa - todas as causas e tipos de acidentes). https://www.gov.br/prf/pt-br/acesso-a-informacao/dados-abertos/dados-abertos-da-prf, 2023.

[3] Graziella Almeida. Após acidente por causa de buraco, moradores reclamam de abandono no caiobá, 2017. URL https://www.campograndenews.com.br/direto-das-ruas/apos-acidente-por-causa-de-buraco-moradores-reclamam-de-abandono-no-caioba.

[4] Danilo de Milano and Luciano Barrozo Honorato. Visao computacional. 2010.

[5] Rafhael Zrenner. Explorando o potencial prático da visão computacional: o que é, aplicação e possibilidades, 2023. URL https://blog.pti.org.br/visao-computacional/.

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0.

[7] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

[8] Paulo Fernando Neres Ferreira, Daniel Lima Gomes Junior, and Alex Martins Santos. Detecçao de uso de máscaras em ambientes fechados com mobilenetv2 e single shot multibox detector. In *Anais da X Escola Regional de Computação do Ceará, Maranhão e Piauí*, pages 61–68. SBC, 2022.

[9] Mariana Alves de Sousa, KXS de SOUZA, J CAMARGO NETO, S TERNES, and IH YANO. Usando a rede neural ssd para identificar frutos verdes em pomares de laranja. In *"CONGRESSO INTERINSTITUCIONAL DE INICIAÇÃO CIENTÍFICA"*, pages 1–10, "Campinas - SC", 2021. Embrapa Agricultura Digital.

[10] Pedro Filipe Costa Araujo. Protótipo de braço robótico capaz de capturar objetos fazendo uso de processamento de imagens e redes neurais. *RE3C-Revista Eletrônica Científica de Ciência da Computação*, 13(1), 2018.

[11] Isabel Costa, Elias Silva Jr, Antônio Rodrigues, Leandro Angeloni, and Edmilson Dias. Avaliação do processo para embarcar uma rede neural baseada em yolo utilizando um acelerador de hardware dedicado. In *Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, pages 174–179. SBC, 2020.

[12] Aline Moura Araújo. Detecção e destaque em vídeo de objetos utilizando yolo. Master's thesis, Universidade Federal da Paraíba - UFPB, João Pessoa , PB, October 2022.

[13] Larissa Barbado, Lucas Medeiros Reinaldet dos Santos, Miguel Diogenes Matrakas, and Jasmine Moreira. Aplicação da rede convolucional yolo para análise de fluxo de veículos. In *Anais do XIX Congresso Latino-Americano de Software Livre e Tecnologias Abertas*, pages 43–49. SBC, 2022.

[14] Ana Cláudia Banderchuk. Detecção de defeitos de fabricação em placas de circuito impresso utilizando técnicas de aprendizado profundo. Technical report, Instituto Federal de Santa Catarina - IFSC, Florianópolis, SC, May 2021.

[15] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on neural networks and learning systems*, 2021.

[16] Yoram Yakimovsky. Boundary and object detection in real-world images. *Journal of the ACM (JACM)*, 23(4):599–618, 1976.

[17] Mohammed Zakariah and Khaled A. AlShalfan. Image boundary, corner, and edge detection: Past, present, and future. 2020. URL https://api.semanticscholar.org/CorpusID:231771734.

[18] João Vitor Esteves Gomes. Detecção de objetos com a arquitetura yolo. Technical report, Universidade Federal de Ouro Preto - UFOP, João Monlevade, MG, Outubro 2022.

[19] Adriana Carrillo Rios. Comparação dos algoritmos yolov3 e ssd para identificação de objetos em imagens, em um dataset para navegação de robôs em ambientes interiores. Technical report, Universidade Federal de Santa Maria - UFSM, Santa Maria, RS, February 2022.

[20] JOÃO OTAVIO GONÇALVES CALIS. AplicaÇÃo de redes neurais convolucionais para reconhecimento automÁtico de placas de veÍculos. Technical report, Universidade Estadual Paulista "Júlio de Mesquita Filho, São José do Rio Preto - SP, December 2018.

[21] Angel Ivanov and Nayden Chivarov. Methods for object recognition and classification for tele-controlled service robots. *IOP Conference Series: Materials Science and Engineering*, 878:012005, 07 2020. doi: 10.1088/1757-899X/878/1/012005.

[22] Max Ferguson, Ronay ak, Yung-Tsun Lee, and Kincho Law. Automatic localization of casting defects with convolutional neural networks. pages 1726–1735, 12 2017. doi: 10.1109/BigData.2017.8258115.

[23] Raimundo Vidal de Sousa Junior. Estudo de métodos de inteligência computacional para detecção de humanos em imagens de webcam. Technical report, Universidade Federal do Ceará - UFC, Fortaleza, CE, 2018.

[24] Kaiyue Liu, Qi Sun, Daming Sun, Lin Peng, Mengduo Yang, and Nizhuan Wang. Underwater target detection based on improved yolov7. *Journal of Marine Science and Engineering*, 11(3):677, 2023.

[25] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. *CoRR*, abs/2101.03697, 2021. URL https://arxiv.org/abs/2101.03697.

[26] Angelo Baruffi Nogueira et al. Análise de viabilidade no uso de deep learning para contagem de pessoas com câmeras de segurança. 2018.

[27] Arthur Costa Serra, João Vitor Ferreira França, Jefferson Alves de Sousa, Robherson Wector de Sousa Costa, Italo Francyles Santos da Silva, Simara Vieira da Rocha, Anselmo Cardoso de Paiva, Aristófanes Correa Silva, Eliana Márcia G Monteiro, Italo Fernandes S da Silva, et al. Segmentaçao semântica de medidores de energia elétrica e componentes de identificaçao. *Brazilian Applied Science Review*, 4(3):2002–2013, 2020.

[28] Rodolfo RV Leocádio, Alan KR Segundo, Jefferson R Souza, Juliana Galaschi-Teixeira, Paulo de Souza, and Gustavo Pessin. Detecçao de abelhas nativas em colmeias em campo utilizando visao computacional. In *Anais do XII Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais*, pages 59–68. SBC, 2021.