UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE TELEINFORMÁTICA

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

ALEXEI ALVES DE SOUZA

AVALIAÇÃO COMPARATIVA DE MODELOS DE VISÃO COMPUTACIONAL PARA RECONHECIMENTO DE PLACAS DE TRÂNSITO.

FORTALEZA

2023

ALEXEI ALVES DE SOUZA

AVALIAÇÃO COMPARATIVA DE MODELOS DE VISÃO COMPUTACIONAL PARA RECONHECIMENTO DE PLACAS DE TRÂNSITO.

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Yuri Lenon Barbosa Nogueira .

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

Souza, Alexei Alves de.
Avaliação comparativa de modelos de visão computacional para reconhecimento de placas de trânsito / Alexei Alves de Souza. – 2023.
49 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia de Computação, Fortaleza, 2023.
Orientação: Prof. Dr. Yuri Lenon Barbosa Nogueira.

1. Visão computacional. 2. Detecção de objetos. 3. Inteligência artificial. 4. Redes neurais. 5. Veículos autônomos. I. Título.

CDD 621.39

ALEXEI ALVES DE SOUZA

AVALIAÇÃO COMPARATIVA DE MODELOS DE VISÃO COMPUTACIONAL PARA RECONHECIMENTO DE PLACAS DE TRÂNSITO.

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: xx/xx/xxxx.

BANCA EXAMINADORA

Prof. Dr. Yuri Lenon Barbosa Nogueira (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. José Gilvan Rodrigues Maia
Universidade Federal do Ceará (UFC)

Prof. Dr. Alexandre Matos Arruda
Universidade Federal do Ceará (UFC)

À minha família, pelo apoio e presença durante todos os momentos.

AGRADECIMENTOS

À Instituição Universidade Federal do Ceará.

Ao Prof. Dr. Yuri Lenon Barbosa Nogueira, pela excelente orientação.

Aos professores participantes da banca examinadora Prof. Dr. José Gilvan Rodrigues Maia e Prof. Dr. Alexandre Matos Arruda pelo tempo, pelas valiosas colaborações e sugestões.

Aos colegas da turma de graduação, pelas convivência e influência no desenvolvimento interpessoal.

"A automação e a inteligência artificial estão transformando radicalmente a maneira como vivemos e trabalhamos. A questão não é se as máquinas pensarão como os humanos, mas se os humanos pensarão como máquinas." (Garry Kasparov)

RESUMO

A automação inteligente de veículos é uma área de estudo que vem crescendo significativamente com a evolução da inteligência artificial aplicada à visão computacional. Essa área inclui desde sistemas de auxílio ao condutor até o desenvolvimento de veículos completamente autônomos. Nesse contexto, a detecção de placas de trânsito ocupa um papel fundamental para prover a um sistema informações relevantes que auxiliem na condução. Neste trabalho são discutidos, de forma resumida, os modelos de detecção de objetos mais utilizados na literatura atual, bem como as métricas utilizadas para avaliação desses modelos. Assim, este trabalho tem como objetivo comparar e analisar alguns dos modelos de detecção de objetos mais relevantes, de modo a concluir qual abordagem é mais adequada para a a tarefa de detecção de placas de trânsito, fornecendo informações e conhecimento que pode ser expandido para outras aplicações semelhantes.

Palavras-chave: visão computacional; inteligência artificial; detecção de objetos; veículos autônomos

ABSTRACT

Intelligent vehicle automation is an area of study that has been growing significantly with the evolution of artificial intelligence applied to computer vision. This area includes everything from driver assistance systems till the development of completely autonomous vehicles. In this context, the detection of traffic signs plays a fundamental role in providing a intelligent system with relevant information that can assist in driving. In this work, is briefly discussed the object detection models who are most used in current literature, as well as the metrics used to evaluate these models. Therefore, this work aims to compare and analyze some of the most relevant object detection models, in order to conclude which approach is most suitable for the task of detecting traffic signs, providing aditional information and knowledge that can be expanded to other similar applications.

Keywords: computer vision; artificial intelligence; object detection; autonomous vehicles.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADAS - Advanced Driver Assistance System	Sistema Avançado de Assistência ao Condutor
ANN - artificial neural network	Redes Neurais Artificiais
AP - Average Precision	Precisão Média
AR - Average Recall	Recall Médio
CNN - convolutional neural network	Redes Neurais Convolucionais
FPN - Pyramid Feature Network	Rede de Pirâmide de Recursos
GTSDB - german traffic sign detection benchmark	Benchmark Alemão de Placas de Trânsito
IoU - intersection over union	Interseção sobre União
KDD - Knowledge Discovery in Databases	Descoberta de Conhecimento em Bancos de Dados
MLP - multilayer perceptron	Perceptron Multicamadas
NMS - Non Maximum Suppression	Supressão Não Máxima
SSD - Single Shot MultiBox Detector	Detector de Mútiplas Caixas de Disparo Único
TCC	Trabalho de Conclusão de Curso
TSD - traffic sign detection	Detecção de Placas de Trânsito

SUMÁRIO

1 INTRODUÇÃO

1.1 Problematização

A automação inteligente de veículos é uma área de estudo que vem sendo bastante beneficiada com o advento de novas tecnologias relacionadas a visão computacional e inteligência artificial. As Redes Neurais Convolucionais (CNN - convolutional neural network), modelos computacionais que buscam imitar o comportamento do cérebro humano e, assim, permitir a uma "máquina enxergar", trouxeram grandes avanços no reconhecimento visual dos elementos que compõem o trânsito urbano, com destaque para as placas de trânsito (WU *et al.*, 2013).

A detecção e reconhecimento de placas de trânsito (Figura 1) são tarefas fundamentais em sistemas de automação veicular. Empresas como Mercedes-Benz, BMW, etc, têm investido ativamente em sistemas como o Sistema Avançado de Assistência ao Condutor (ADAS - Advanced Driver Assistance System), que incluem detecção e reconhecimento de placas de velocidade, os quais auxiliam o condutor através de alertas a estar atento a velocidade máxima (WANG *et al.*, 2020).

Figura 1 – Exemplo de detecção de placas de trânsito em imagens

Fonte: Adaptado de https://www.mapillary.com/app.

Porém, essa tarefa envolve uma série de complicações e restrições que devem ser consideradas. A posição, o ângulo, e o degaste de uma placa, bem como as condições de iluminação e distorções do cenário em que se encontra, fazendo com que seja necessário um profundo estudo sobre qual metodologia é a mais adequada para mitigar possíveis erros de detecção e reconhecimento (DEEPIKA; V, 2017).

A literatura atual possui uma variedade de algoritmos para detecção de objetos em imagens, os quais vêm sendo otimizados ao decorrer dos anos. Compreender o funcionamento de um conjunto desses modelos e os analisar comparativamente para a tarefa de detecção de placas contribui não somente para entender suas limitações com relação a precisão, robustez e desempenho computacional, como para auxiliar na escolha do modelo mais adequado para uma determinada aplicação (ARCOS-GARCíA *et al.*, 2018).

Para consolidar essas análises, as métricas geralmente utilizadas para validação de modelos de detecção de objetos são a Precisão Média (AP - Average Precision) e o Recall Médio (AR - Average Recall), as quais tem como objetivo fornecer informações como o quão corretas são as regiões detectadas, e se o objeto contido em determinada região corresponde a classe desejada. Porém, essas métricas podem apresentar divergências e variações com relação a implementação, não havendo um consenso definitivo pela comunidade acadêmica e científica. Assim, neste trabalho são utilizadas as métricas padrão de AP e AR, fornecidas pelos *frameworks* utilizados, somadas a uma métrica que calcula a porcentagem total de detecções total que um modelo obteve em um conjunto de placas, a qual provê um critério concreto e simples para análise da precisão do modelo, baseado apenas na quantidade de acertos e erros (PADILLA *et al.*, 2020).

1.2 Objetivos

Objetivos gerais: Realizar uma análise comparativa de modelos de visão computacional aplicados a detecção de placas de trânsito, destacando aqueles que obtiveram os melhores resultados com relação as métricas utilizadas.

Objetivos específicos:

- Comparar os modelos de detecção de objetos em imagens CenterNet, SSD, EfficientDet, Faster-RCNN e YOLO, treinados para um banco de imagens de placas de trânsito brasileiras.
- Realizar análises com relação as métricas de: % total de detecções, AP - Average Precision,

% Falsos positivos, e velocidade de processamento médio.

– Gerar tabelas contendo os modelos que apresentaram os melhores resultados com relação a cada métrica analisada.

1.3 Organização do trabalho

A divisão deste trabalho se da da seguinte forma, sendo dividido em outros cinco capítulos:

– Fundamentação teórica : capítulo que apresenta de forma resumida as arquiteturas dos modelos utilizados e seu funcionamento, bem como as métricas utilizadas nas análises;

– Trabalhos relacionados : capítulo que resume alguns dos trabalhos mais relevantes com relação a detecção de placas de trânsito em imagens, os quais trazem otimizações e conceitos que demonstram o constante avanço nessa área;

– Metodologia : capítulo que apresenta os passos utilizadas na condução da pesquisa, desde a coleta do banco de imagens até a obtenção das tabelas de resultados;

– Resultados : capítulo onde é realizada a apresentação dos resultados, analisando as métricas individualmente de modo a extrair do conjunto de modelos aqueles mais adequadas para um determinado objetivo;

– Conclusão : capítulo onde são realizadas as considerações finais com relação aos resultados obtidos na pesquisa e os possíveis trabalhos futuros que possam complementar e enriquecer a tarefa de detecção de placas de trânsito em imagens.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados, de forma resumida, os principais conceitos que norteiam a detecção de objetos e sua aplicação na Detecção de Placas de Trânsito (TSD - traffic sign detection). Além disso, são também abordadas as diferentes arquiteturas dos algoritmos de detecção de objetos utilizados neste trabalho para fins de comparação e análise : CenterNet, EfficientDET, SSD, Faster R-CNN e YOLO.

2.1 Redes neurais

2.1.1 Redes Neurais artificiais

Redes Neurais Artificiais (ANN - artificial neural network) são sistemas de processamento computacional cujo funcionamento é inspirado em como o sistema nervoso humano opera. A estrutura básica de uma ANN é mostrada na Figura 2, sendo composta por neurônios, organizados em camadas (O'SHEA; NASH, 2015).

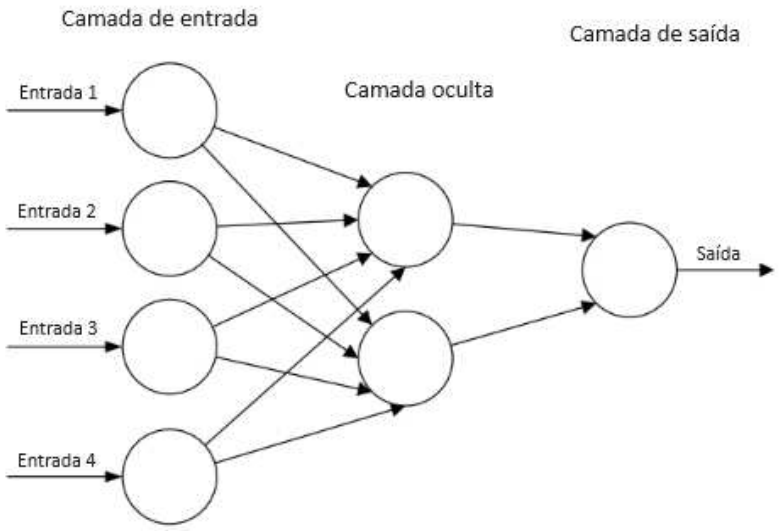
Neurônios são nós computacionais interconectados que realizam cálculos nas entradas a eles conectadas e geram saídas. Cada neurônio possui um peso, ou seja, um parâmetro ajustável que influencia no resultado gerado na saída (O'SHEA; NASH, 2015).

Já as camadas são agrupamentos de neurônios, podendo, em uma ANN, serem classificadas em três tipos principais:

1. Camada de entrada : Esta é a camada que recebe os dados iniciais, geralmente vetores multidimensionais;
2. Camada oculta: É a camada intermediária, podendo haver uma ou mais camadas desse tipo. Nela é onde são tomadas as decisões com relação aos dados da camada anterior, de modo a avaliar como os pesos utilizados melhoram ou pioram o resultado final desejado;
3. Camada de saída: É a camada que retorna a saída da rede neural. Nessa camada é comum a aplicação de funções de ativação no resultado obtido, como por exemplo a função sigmoide que produz valores entre 0 e 1, que podem ser interpretados como probabilidades em problemas de classificação binária, e a função softmax, para problemas de classificação multiclasse, que retorna um vetor com as probabilidades para cada classe.

O treinamento de uma ANN se dá a partir de dois paradigmas: aprendizagem supervisionada e aprendizagem não supervisionada. Na aprendizagem supervisionada o conjunto

Figura 2 – Uma rede neural feedfoward (FNN) simples de três camadas



Fonte: (O'SHEA; NASH, 2015)

de dados de entrada e suas respectivas saídas já estão rotulados. Assim, em cada iteração do treinamento, os pesos das camadas da rede são atualizados até que sejam adequados para que os resultados de saída obtidos pela rede sejam iguais ou muito próximos dos resultados desejados já rotulados. Já na aprendizagem não supervisionada, os dados de entrada não estão rotulados, sendo bem pouco utilizada em tarefas relacionadas ao reconhecimento de imagens, e sim em problemas de agrupamento, redução de dimensionalidade, e outras aplicações que visam entender a disposição implícita dos dados (O'SHEA; NASH, 2015).

Quando abordamos problemas de reconhecimento de imagens, um modelo básico que pode ser utilizado é o Perceptron Multicamadas (MLP - multilayer perceptron), uma ANN com múltiplas camadas ocultas que recebe como entrada a matriz da imagem vetorizada. Porém, essa abordagem possui alguns problemas quando relacionada ao processamento de imagens pois redes MLP não são invariantes por translação e tendem ao *overfitting*. Além disso, a alta dimensionalidade dos dados pode exigir uma rede intratável, dado o elevado número de parâmetros exigidos.

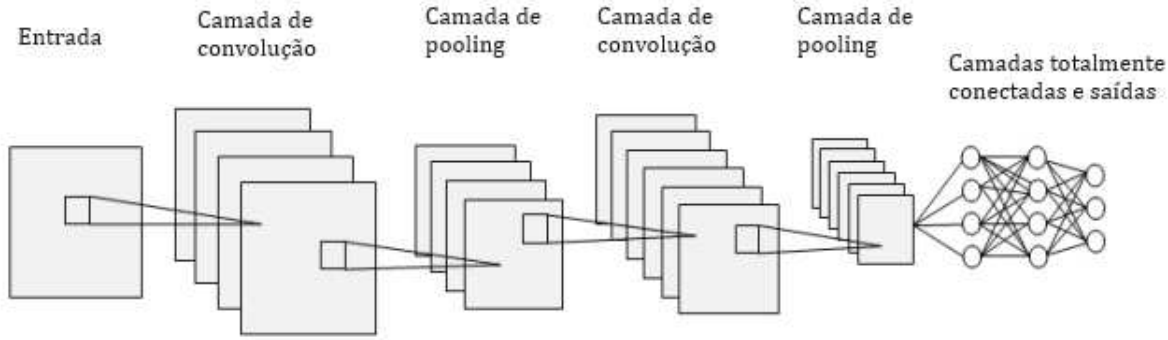
Não ser invariante por translação significa que a rede retorna resultados diferentes e possivelmente incorretos se o conteúdo principal da imagem for alterado, por exemplo: objetos de uma imagem deslocados. Já *overfitting* se refere ao modelo se tornar propenso a superajustar os dados de treinamento, tornando-se incapaz de processar corretamente imagens diferentes que aquelas utilizadas no treinamento. Isto ocorre pois quando uma imagem é submetida a uma rede MLP ela deve ser vetorizada, ou seja, cada pixel da imagem irá ocupar um neurônio de

entrada. Uma imagem pequena de tamanho por exemplo 224x224x3 precisaria então de 150.528 neurônios na camada de entrada, e considerando uma camada oculta modesta com 3 camadas de 128 neurônios cada, os parâmetros treináveis chegariam a centenas de bilhões, gerando um *overfitting* devido a imensa quantidade de valores a serem ajustados que exigem muitas passagens nas imagens do conjunto de treino, além de um tempo de treinamento excessivamente grande (LEARNOPENCV, 2023). Devido a isso, foram desenvolvidas as redes neurais convolucionais(CNN), que tornam o processamento de uma imagem no contexto da inteligência artificial bem mais eficaz e eficiente.

2.1.2 Redes neurais convolucionais

Redes Neurais Convolucionais (CNN - convolutional neural network) são um tipo de rede neural profunda do tipo *feedforward* projetada para tarefas relacionadas ao processamento e análise de imagens (HAKIM; FADHIL, 2021). Em comparação com as redes neurais tradicionais, as CNN possuem uma arquitetura especial, com camadas que realizam operações de convolução para extração de padrões, nas quais o compartilhamento de pesos no processamento de diferentes camadas reduz também o número de parâmetros utilizados no treinamento, aumentando significativamente a velocidade e eficiência do modelo (LEARNOPENCV, 2023). Uma arquitetura simplificada de uma CNN é ilustrada na figura 3.

Figura 3 – Arquitetura básica de uma CNN



Fonte: (HAKIM; FADHIL, 2021)

A funcionalidade básica da CNN apresentada acima pode ser reduzida em quatro conceitos chave (O'SHEA; NASH, 2015):

1. Camada de entrada: É a camada que recebe a matriz de pixels correspondente à imagem;
2. Camada de convolução: É a camada que define o nome CNN. Os parâmetros dessa camada tem como seu foco *kernels* treináveis, os quais são aplicados em toda a dimensão da

imagem de entrada, realizando operações de convolução, de modo a extrair um mapa de características contendo padrões;

3. Camada de pooling: É a camada responsável por reduzir a dimensionalidade dos mapas de características gerados pela camada de convolução e assim reduzir o número de parâmetros e a complexidade do modelo. A operação mais comum é o *max pooling*, que extrai o valor máximo de uma região;
4. Camada totalmente conectada: São camadas semelhantes às camadas de uma rede neural tradicional, que recebem as informações extraídas das camadas convolucionais e de pooling e as transformam em saídas para as diferentes classes de interesse.

2.2 Detecção de objetos

A detecção de objetos em uma imagem é uma das muitas utilidades que as CNNs possuem. Essa tarefa apresentou muitos desafios, devido a fatores como a iluminação e ponto de vista, entres outras variações que uma mesma classe de objeto pode apresentar em uma imagem. Assim, foi com o aprofundamento do estudo das CNNs que foram desenvolvidos vários métodos que tiveram sucesso em reconhecer e detectar objetos em imagens (HAKIM; FADHIL, 2021).

É nesse contexto que foi desenvolvido o Tensorflow, uma plataforma de código aberto voltada para tarefas relacionadas a aprendizagem de máquina e inteligência artificial. Através dele é possível ter acesso a diversas ferramentas para criação e treinamento de redes neurais, além de modelos pré-treinados para as mais diversas tarefas relacionadas a visão computacional. Neste trabalho os métodos de detecção de objetos utilizados são aqueles fornecidos pela API de detecção de objetos do Tensorflow, os quais são: CenterNet, EfficientDET, SSD, Faster R-CNN. A esse soma-se o YOLO, mais especificamente a versão cinco desenvolvida pela Ultralytics, uma empresa que desenvolve soluções para problemas de aprendizagem profunda e visão computacional.

2.2.1 CenterNet

CenterNet é uma arquitetura de detecção de objetos de um estágio baseada em *keypoints* (pontos-chave). Ela foi desenvolvida tomando como base a CornerNet (LAW; DENG, 2018), uma arquitetura que detecta caixas delimitadoras como um par de pontos chave (ponto superior esquerdo e ponto inferior direito). Porém, essa arquitetura demonstrou ser muito

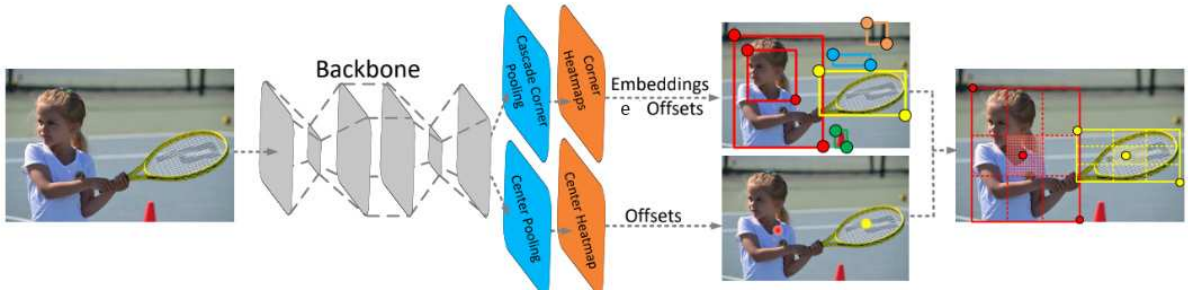
sensível à detecção dos limites dos objetos, possuindo a limitação de não conseguir distinguir eficientemente quais pares de pontos deveriam ser agrupados em um mesmo objeto, o que podia gerar caixas delimitadoras incorretas (DUAN *et al.*, 2019).

Para solucionar esse problema, foi então desenvolvida a CenterNet, que propôs a representação de um objeto não apenas como um par de pontos, mas sim como um trio, sendo esse terceiro o centro. Essa arquitetura utiliza duas estratégias para enriquecer as informações obtidas do centro e dos cantos, respectivamente. A primeira é denominada *center pooling*, a qual auxilia os *keypoints* centrais a estabelecer padrões reconhecíveis dentro dos objetos, e assim facilitar a percepção de sua região central. A segunda estratégia é denominada *cascade corner pooling*, que desempenha a função de reunir informações dos objetos em um mapa de recursos para previsão dos cantos (DUAN *et al.*, 2019).

A arquitetura geral da CenterNet pode ser observada na figura 4, e o método proposto pode ser resumido nos seguintes procedimentos (DUAN *et al.*, 2019):

1. Gerar um número k de caixas delimitadoras utilizando o método proposto pelo modelo CornerNet (LAW; DENG, 2018);
2. Selecionar um número k de *keypoints* centrais, ordenados de acordo com seus scores;
3. Utilizar as coordenadas(offsets) correspondentes para remapear esses *keypoints* centrais para a imagem de entrada;
4. Definir a região central de cada caixa delimitadora e checar se ela possui *keypoints* centrais com a mesma classe da caixa;
5. Se forem detectados *keypoints* centrais na região central, o score da caixa é atualizado para a média dos scores dos três pontos(ponto superior esquerdo, ponto inferior direito e ponto central). Caso não sejam encontrados pontos centrais a caixa em questão é descartada.

Figura 4 – Arquitetura da CenterNet



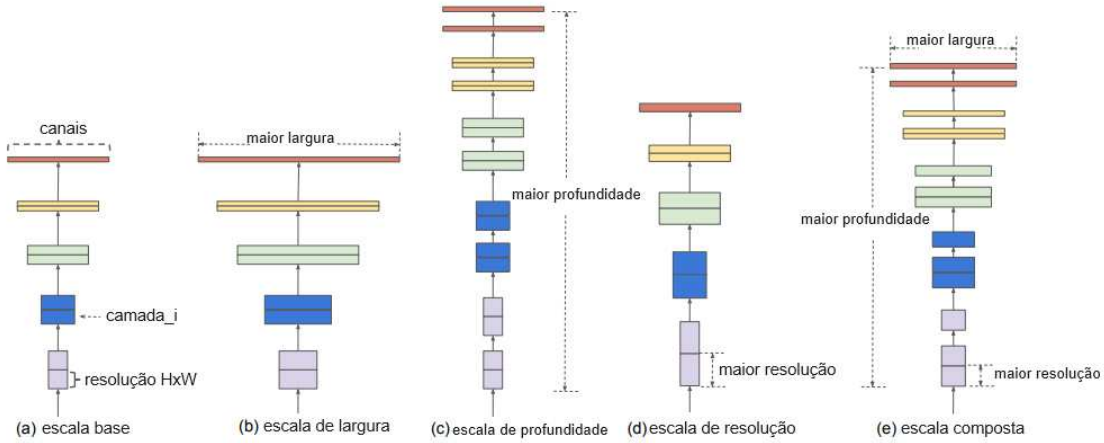
Fonte: (DUAN *et al.*, 2019)

2.2.2 EfficientDet

EfficientDet é uma arquitetura de redes neurais utilizada para detecção de objetos baseada no conceito de escalabilidade, a qual busca otimizar o processo de extração de características equilibrando precisão e eficiência. Sua estrutura é constituída utilizando como *backbone* uma rede EfficientNet acoplada a uma Rede de Pirâmide de Recursos (FPN - Pyramid Feature Network) (TAN *et al.*, 2019).

A rede EfficientNet se trata de uma família de modelos que visa escalar uniformemente a largura, profundidade e resolução de uma rede, e assim se adequar as dimensões das imagens utilizadas na entrada, aumentando a precisão das previsões e a eficiência da rede. Na Figura 5 é mostrado como se dá conceito de escalabilidade em redes neurais (TAN; LE, 2019).

Figura 5 – EfficientNet - Escalonamento do modelo. (a) rede base padrão;(b)-(d) redimensionamento da rede por lagura,profundidade e resolução; (e) escala composta das dimensões uniformemente.



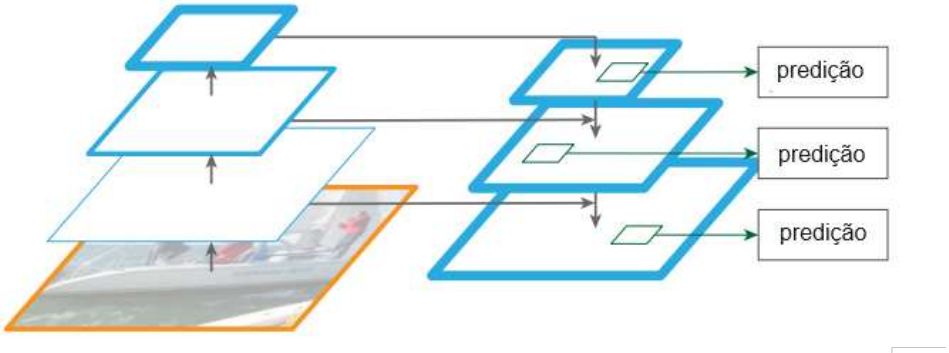
Fonte: (TAN; LE, 2019)

Já as FPNs são estruturas muito utilizadas em sistemas de visão computacional para extrair mapas características de uma imagem em diferentes escalas de resolução (Figura 6). Em detecção de objetos isso permite identificar padrões que se adaptem tanto para identificação de objetos grandes como de objetos pequenos (LIN *et al.*, 2016).

Na EfficientDet é utilizada uma estrutura de FPN com 5 modificações. Essa estrutura é denominada BiFPN (Figura 7), cujas modificações são (TAN *et al.*, 2019):

1. Em vez de apenas uma direção de conexão de cima para baixo , é adicionado outra conexão para fusão de recursos de baixo para cima;
2. Há conexões de salto do mapa de recursos inicial diretamente para os mapas de recursos

Figura 6 – FPN - arquitetura top-down com predições realizadas em cada nível

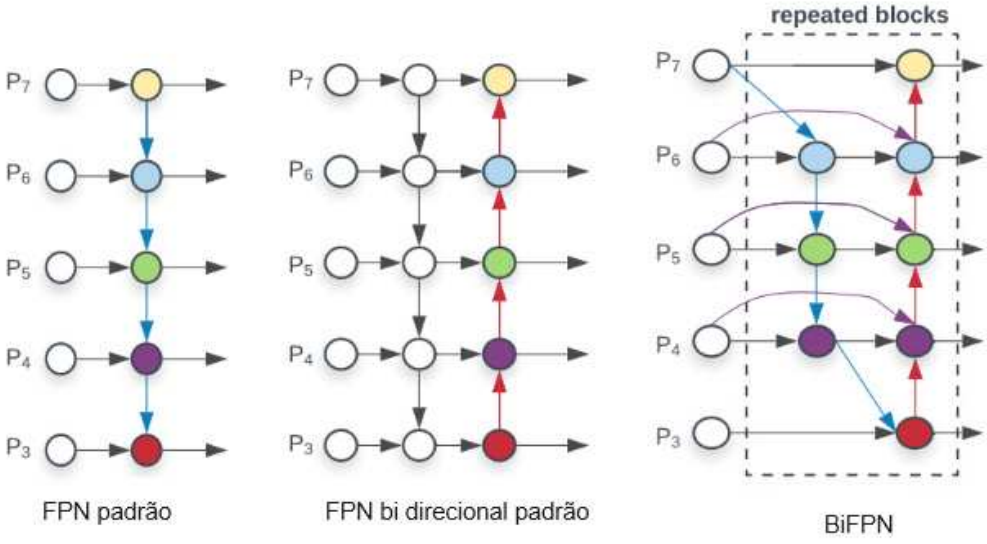


Fonte: (LIN *et al.*, 2016)

fundidos;

3. Nós com apenas uma entrada são removidos, pois não realizam tanta fusão quanto outros nós;
4. Todo o módulo é repetido várias vezes;
5. Os recursos não são somados diretamente; em vez disso, uma média ponderada é usada para que mapas de recursos de resoluções diferentes contribuam para a fusão em capacidades diferentes.

Figura 7 – Arquiteturas de redes de pirâmides de recursos

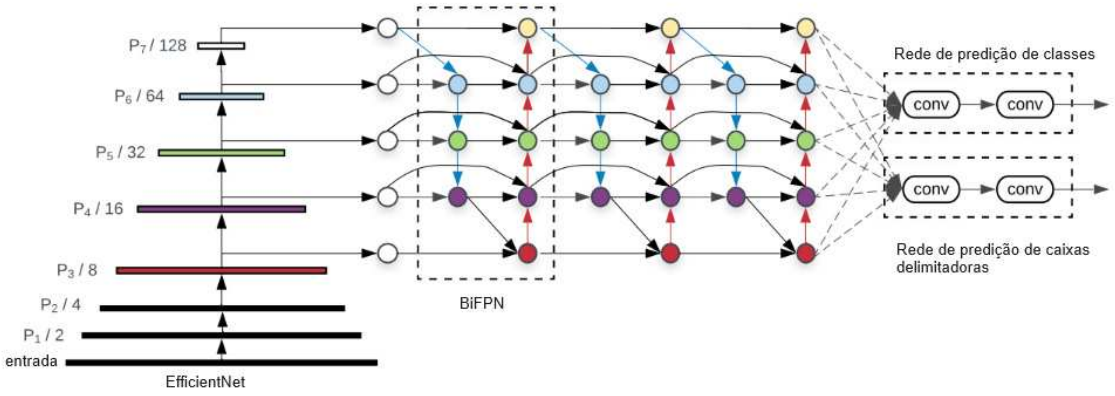


Fonte: (TAN *et al.*, 2019)

A partir da junção dessas estruturas que utilizam de conceitos de redes neurais escaláveis e uma abordagem de extração de recursos em múltiplas resoluções, somadas a uma rede para predição de classes/caixas delimitadoras, é então construído o modelo EfficientDet

como observado na Figura 8

Figura 8 – Arquitetura EfficientDet



Fonte: (TAN *et al.*, 2019)

2.2.3 Faster R-CNN

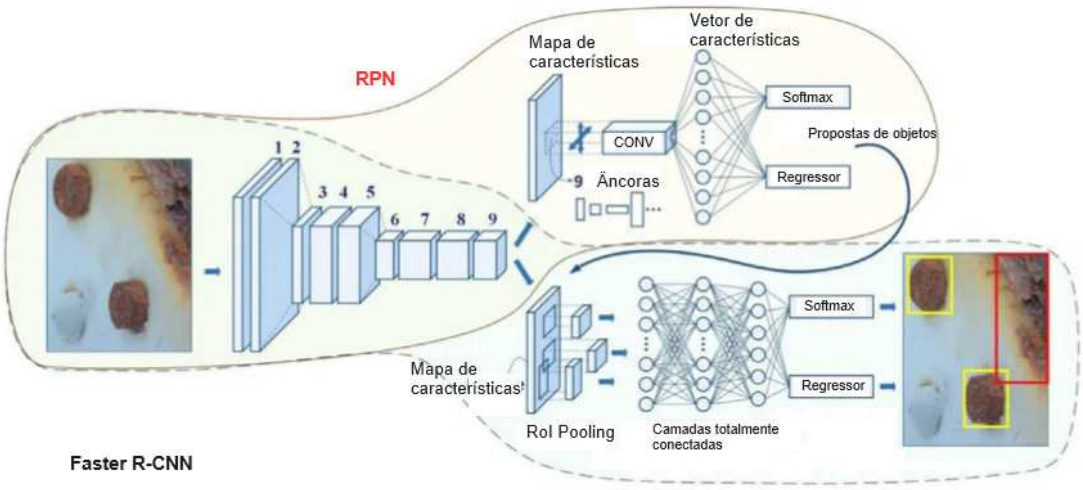
A Faster-RCNN é uma arquitetura (Figura 9) de redes neurais de dois estágios cuja função principal é a detecção de objetos. Ela agrega em sua estrutura uma pequena rede neural para proposta de regiões (RPN - Region Proposal Network), que desliza sobre o mapa de características extraído da última camada de convolução. Esta rede tem o objeto de prever se em determinada região há ou não objetos, além de também prever possíveis caixas delimitadoras para esses objetos, as quais são denominadas âncoras (REN *et al.*, 2015).

A outra estrutura que compõe a arquitetura da Faster R-CNN é uma rede neural, que toma como entrada os objetos propostos pela rede RPN, e os submete a uma camada de *Region of Interest (RoI) pooling* (GIRSHICK *et al.*, 2013). Esta camada extrai pequenos mapas de características para cada caixa candidata e então os submete a uma rede totalmente conectada que realiza operações de classificação, para previsão das classes dos objetos, e regressão, para previsão das caixas delimitadoras (REN *et al.*, 2015).

2.2.4 SSD

O Detector de Mútiplas Caixas de Disparo Único (SSD - Single Shot MultiBox Detector) é um arquitetura (Figura 10) desenvolvida para detecção de objetos em tempo real. Sua abordagem é de uma CNN que produz um conjunto de tamanho fixo de caixas delimitadoras e scores para a presença de determinada classe de objeto nessas caixas, seguida por uma etapa de

Figura 9 – Arquitetura Faster R-CNN



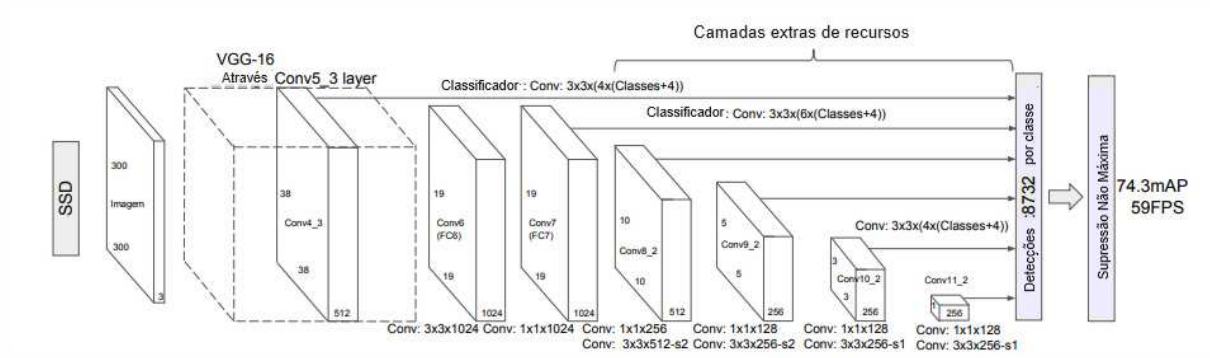
Fonte: (KHAZRI, 2019)

supressão não-máxima para produzir os resultados finais (LIU *et al.*, 2015).

A estrutura do modelo SSD é constituída das seguintes estruturas e recursos principais:

1. Rede neural base: É a parte inicial do modelo. Trata-se de uma CNN utilizada para classificação truncada nas camadas finais que produzem os resultados de classificação;
2. Mapas de características em várias escalas para detecção: No final da rede neural de base truncada são adicionadas camadas convolucionais que decrescem progressivamente e permitem a predição de detecções em múltiplas escalas;
3. Preditores convolucionais para detecção: Cada camada de recursos pode produzir um conjunto de predições de tamanho fixo usando filtros convolucionais;
4. Caixas predefinidas: Ao contrário da Faster R-CNN que utiliza uma rede para previsão de regiões e caixas âncora, o que torna o processamento significativamente mais lento, o modelo SSD possui caixas delimitadoras predefinidas. Essas caixas são aplicadas aos mapas extraídos das camadas de recursos em diferentes resoluções, permitindo, assim, discretizar com eficiência as possíveis dimensões das caixas de saída (LIU *et al.*, 2015);
5. Supressão não-máxima: A Supressão Não Máxima (NMS - Non Maximum Suppression) é uma técnica aplicada após a etapa de proposição para produzir as detecções finais. A ideia da NMS funciona através da comparação das pontuações de confiança das caixas delimitadoras propostas, eliminando aquelas que se sobrepõem a uma respectiva caixa com pontuação mais alta, e assim selecionando as mais relevantes (HOSANG *et al.*, 2017).

Figura 10 – SSD com entrada de dimensão 300x300, e rede neural VGG-16 como base



Fonte: (LIU *et al.*, 2015)

2.2.5 YOLO

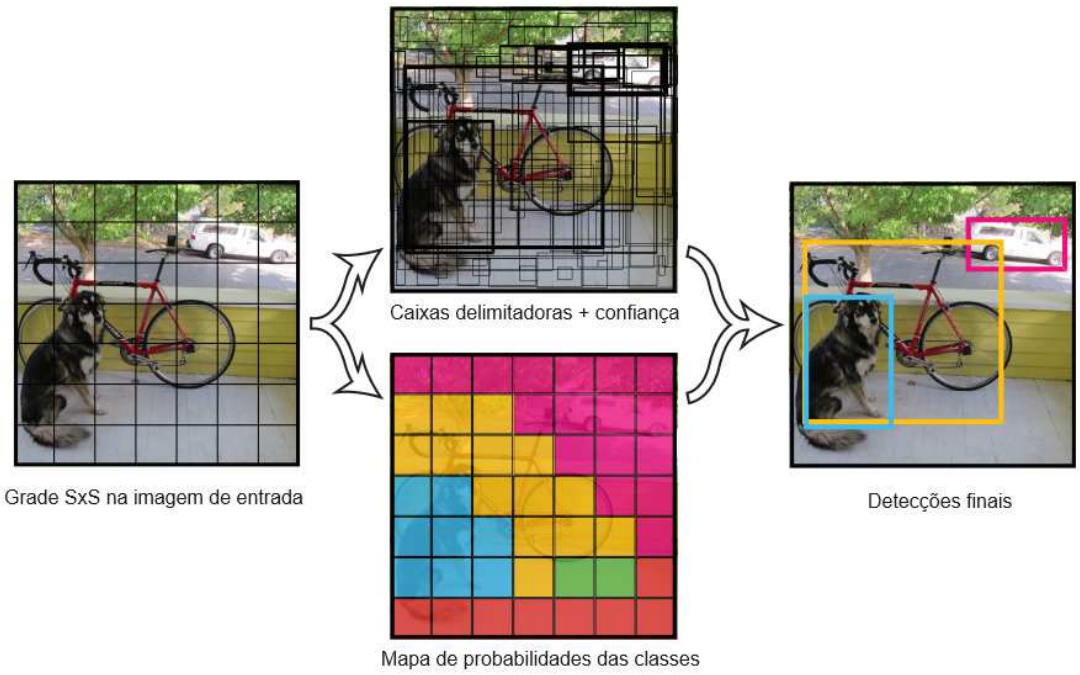
YOLO (You Only Look Once - Você apenas olha uma vez) é um sistema que unifica as estruturas necessárias para detecção de objetos em uma imagem em uma única rede neural. Essa rede atua globalmente em toda a imagem prevendo simultaneamente as caixas delimitadoras para todas as classes (REDMON *et al.*, 2015). Seu principal objetivo é prover uma rede otimizada com performance que equilibra precisão e rapidez para aplicações em tempo real.

Para detectar os objetos o YOLO divide a imagem de entrada em uma grade regular de dimensões SxS, em que cada célula da grade é responsável por prever um certo número de caixas delimitadoras e seus respectivos scores de confiança. Esse score é uma métrica para quantificar a probabilidade da caixa conter um objeto e quão precisa é essa caixa. Em geral o score de confiança é calculado como a Probabilidade(objeto) x Interseção sobre União (IoU - intersection over union) (REDMON *et al.*, 2015).

2.3 Avaliação de modelos de detecção de objetos

Para avaliar métodos de detecção de objetos é necessário compreender os conceitos que definem as métricas utilizadas para análise da precisão e acurácia. Em geral a métrica mais utilizada é a AP (Average Precision - Precisão Média) a qual é construída a partir dos conceitos de verdadeiros positivos, falsos positivos e IoU (Intersection Over Union - Interseção Sobre a União). Nesta serão serão resumidos esses conceitos para enriquecer o entendimento das análises.

Figura 11 – YOLO: funcionamento do modelo



Fonte: (REDMON *et al.*, 2015)

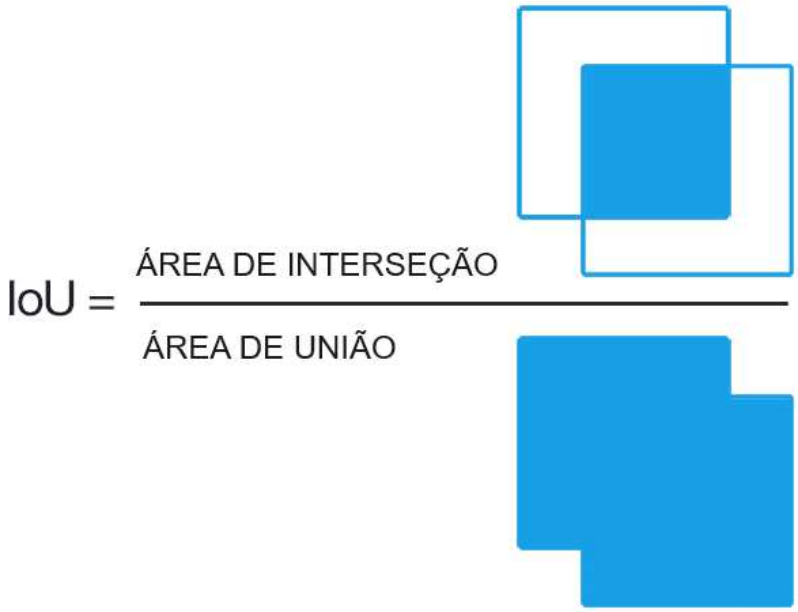
2.3.1 IoU - Interseção sobre a união

A interseção sobre a união é uma métrica baseada no índice de Jaccard que avalia a sobreposição entre duas caixas delimitadoras (HANCOCK, 2004). Para seu cálculo é necessária uma caixa delimitadora real e sua respectiva caixa delimitadora prevista. O valor do IoU é dado pela divisão da área obtida pela sobreposição entre a caixa delimitadora real e a caixa delimitadora prevista pela área obtida da união entre elas (Figura 12) (DUBEY, 2020).

A partir da aplicação da IoU é possível construir a matriz de confusão para a análise de determinado modelo, no qual um limite mínimo para o valor do IoU é estabelecido. Em detecção de objetos a definição de verdadeiros positivos/negativos e falsos positivos/negativos é definida da seguinte forma (GENG, 2022):

- Verdadeiro positivo (TP - True Positive): Detecção correta, ou seja quando o IoU > limite e a classe associada prevista coincide com a real.
- Falso positivo (FP - False Positive): Detecção incorreta, ou seja quando uma caixa é prevista mas não corresponde a nenhuma classe.
- Verdadeiro negativo (TN - True Negative) : Não é relevante para detecção de objetos, pois trata do conjunto de todas as caixas delimitadoras que não deveriam ser detectadas, o que em uma imagem corresponde a todas as áreas que não possuem classe associada.

Figura 12 – IoU: Cálculo da interseção sobre a união



Fonte: (ROSEBROCK, 2016)

– Falso negativo (FN - False Negative): Previsão ausente, quando determinada caixa delimitadora não é detectada pelo modelo para um IoU > limite.

2.3.2 Métricas

A partir dos conceitos discutidos anteriormente é então possível definir as métricas que em geral são utilizadas: precisão, *recall* e AP (DUBEY, 2020).

– Precisão : Precisão = $\frac{TP}{TP+FP} = \frac{TP}{TotalCaixasDetectadas}$
– Recall : Recall = $\frac{TP}{TP+FN} = \frac{TP}{TotalCaixasReais}$
– AP : A precisão média é calculada a partir da média de AP para todos os limites de IoU utilizados. Na ferramenta de validação de modelos tanto do Tensorflow como do YOLO o AP utilizado como parâmetro considera os limites de IoU de 0.5:0.95.
– AR (Average Recall - Recall Médio): Trata-se do recall máximo em relação a um número fixo de detecções por imagem, em que é calculada a média sobre as classes e IoUs (CONTEXT, 2017).
– Score de confiança: É o índice que calcula o quão certo a caixa delimitadora contém um objeto e quão semelhante é a caixa predita em relação a caixa real. Sua fórmula se dá por : $C = Pr(Objeto) * IoU$

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados trabalhos que buscam trazer contribuições para a tarefa de detecção de placas de trânsito em imagens. Vale ressaltar que o objetivo deste Trabalho de Conclusão de Curso (TCC) é realizar uma análise e comparação da arquitetura de modelos de detecção em sua estrutura padrão, e não explorar modelos com modificações em sua arquitetura original, através da inserção de camadas adicionais de processamento, assim preservando conclusões tomadas nos modelos em sua estrutura convencional. Entretanto, como pode ser observado, eles guiam as arquiteturas que devem ser analisadas no problema de detecção de placas de trânsito.

No trabalho de Huang *et al.* (2018) é proposto um método para aperfeiçoar a detecção de objetos pequenos em imagens, no caso de placas de trânsito. Nele é utilizada uma arquitetura formada pela combinação de um modelo de detecção de objetos (Faster-RCNN) e uma rede generativa adversarial (GAN).

Uma GAN é formada por duas redes neurais, uma geradora e uma discriminadora, em que a rede geradora tenta gerar dados artificiais e fazer com que a rede discriminadora os reconheça como verdadeiros. A ideia do método proposto é treinar uma rede geradora que possa converter as características de objetos pequenos em características similares a de objetos grandes, "enganando assim"a rede discriminadora. Assim, com essa estratégia, seria possível então aumentar significativamente a performance da detecção de objetos pequenos quando comparada a Faster R-CNN padrão. Nesse trabalho é apresentado um método viável que poderia ser acoplado a vários modelos de detecção de objetos para aumentar sua precisão em relação a objetos pequenos.

A contribuição do trabalho de Wang *et al.* (2022) é a otimização da detecção de objetos em cenários complexos, em que é comum a existência de ruídos de diversos tipos, como iluminação, climáticos, etc. Nesse método as imagens são submetidas a um camada de pré-processamento especializada, para realizar a atenuação de ruídos. Além disso, é acrescentada uma FPN - Pyramid Feature Network modificada para reter mais informações de imagens de diferentes tamanhos e reduzir as perdas de características. Acrescentar uma camada para pré-processamento de ruídos e normalização das imagens é uma técnica que pode ser acoplada em diversos modelos de detecção de objetos.

O trabalho de Arcos-García *et al.* (2018) busca trazer uma comparação entre os principais modelos de detecção de objetos da atualidade para detecção de placas de trânsito.

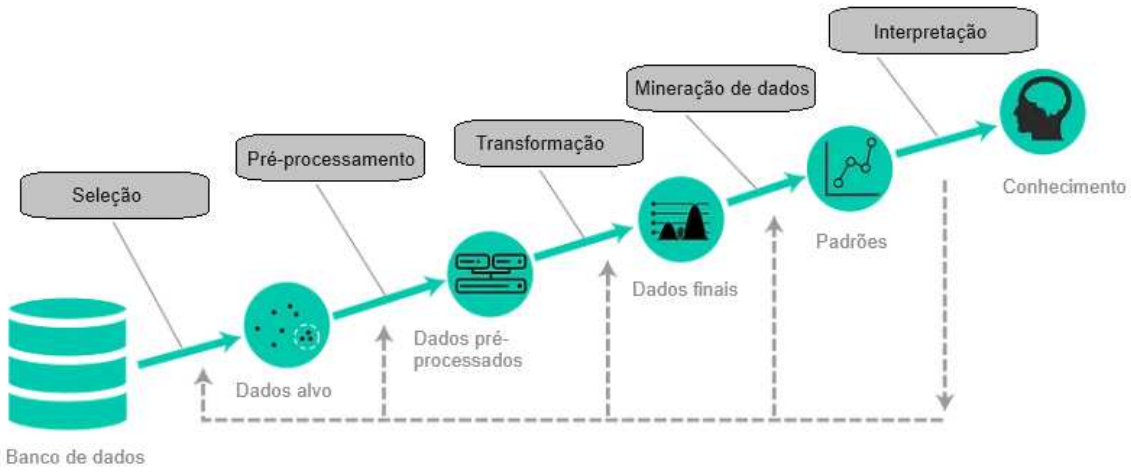
Nele é utilizado o dataset alemão Benchmark Alemão de Placas de Trânsito (GTSDB - german traffic sign detection benchmark), sendo realizadas uma série de análises para determinar a precisão e velocidade de detecção dos modelos testados. A premissa desse trabalho é trazer uma comparação de modelos treinados em um conjunto de imagens de cenários contendo placas de trânsito alemãs, padronizados para uma única resolução de entrada.

O trabalho de Yuan *et al.* (2019) trata da construção de uma arquitetura para detecção de objetos aplicada ao contexto de detecção de placas de trânsito. Nele é utilizada a arquitetura do modelo SSD como base, acrescido de algumas estruturas extras com objetivos específicos. A primeira delas é uma estrutura formada por camadas de deconvolução, a qual tem o objetivo de ampliar os mapas de características e extrair recursos semânticos superiores. Em seguida, a sequência espacial obtida dos mapas de características é codificada por uma rede LSTM, que processa esses recursos e captura as informações relevantes para o contexto. Esse trabalho traz uma arquitetura que utiliza de redes neurais profundas bastante utilizadas em tarefas de visão computacional e outras áreas, para complementar e otimizar a detecção de objetos.

4 METODOLOGIA

Este capítulo tem como objetivo descrever o método utilizado para a condução da pesquisa. O método escolhido foi o da Descoberta de Conhecimento em Bancos de Dados (KDD - Knowledge Discovery in Databases), um método da área de ciência de dados que tem como objetivos extrair informações e *insights* relevantes sobre um banco de dados. Essa escolha se deu pois, além de definir etapas comuns durante o processo de análise de modelos de visão computacional, ele se enfoca na extração de conhecimento a partir de uma consideração implícita dos dados obtidos nos resultados (MATHEUS *et al.*, 1998).

Figura 13 – KDD: Método de descoberta de conhecimento em bancos de dados



Fonte: (RAMIREZ, 2021)

A seguir são apresentadas as etapas do método KDD e como foram adaptadas para a tarefa de detecção de placas de trânsito em imagens.

4.1 Seleção

Essa etapa é uma das mais importantes, pois trata do entendimento do projeto, seus objetivos, bem como sua viabilidade com relação a disponibilidade dos recursos necessários. Neste trabalho foi escolhido o tema de comparação de modelos para detecção de placas de trânsito em imagens, pois é um assunto que vem sendo bastante estudado no contexto atual de tecnologias inteligentes e autônomas. Para conferir a viabilidade do tema, foram realizadas pesquisas com relação tanto à disponibilidade de modelos para uso livre, como da existência de algum conjunto de dados contendo uma quantidade razoável de placas de trânsito brasileiras em

boa resolução.

Os modelos utilizados para análise foram os disponibilizados no Tensorflow Model Zoo, um repositório de código aberto com um conjunto de modelos de detecção de objetos pré-treinados (VIGHNESHBIRODKAR; TEAM, 2021). Foram também utilizados algumas arquiteturas do modelo YOLO para resoluções de entrada de 640 e 1280 pixels. A versão do YOLO utilizada foi a versão 5, desenvolvida pela Ultralytics, a qual é disponibilizada na plataforma do Github [1] para uso livre. Já o conjunto de imagens contendo placas de trânsito foi obtido a partir da API do Mapillary [2], uma plataforma que mapeia as ruas de todo o mundo e possui licença livre para a utilização de suas imagens.

4.2 Pré-processamento

Quando se trata de detecção de objetos, é necessário que o conjunto que será utilizado para treinamento esteja mapeado. Ou seja, que cada placa de trânsito do conjunto de treino tenha as posições de sua caixa delimitadora anotada. Apesar de existirem conjuntos de dados mapeados para outras regiões do mundo, não foram encontrados *datasets* específicos para placas de trânsito brasileiras. Devido a isso, foi necessário realizar o mapeamento e redimensionamento das imagens para um tamanho padrão.

O mapeamento das caixas reais foi realizado de forma manual, para um total de 1252 imagens de treino e 313 imagens de teste, com em média 2 placas por imagens. Já o redimensionamento foi realizado para o tamanho da tela do dispositivo utilizado para desenvolvimento do trabalho, de 1366x768 pixels.

4.3 Transformação

Esta etapa trata das modificações finais realizadas no conjunto de dados, para tornar-los prontos para serem submetidos aos modelos de análise. Para a tarefa de detecção de placas de trânsito foi utilizada neste trabalho uma abordagem que visa prover para os modelos de detecção o máximo de informação possível. A abordagem utilizada foi agrupar placas de características semelhantes em quatro grupos (Figura 14): sendo os dois primeiros a classe de placas de regulamentação e a classe de placas de advertência, e os dois últimos as placas de "Pare"e "Dê a preferência", as quais possuem características bastantes distintas das demais. Essa divisão se deu

[1] https://github.com/ultralytics/yolov5
[2] https://www.mapillary.com/

pois, dividir o conjunto de placas para cada nomenclatura individualmente se mostrou inviável, não somente pela limitação da quantidade de imagens do conjunto de dados base, como o fato de que designar mais classes iria requerer um tempo de mapeamento significativamente maior para atender uma quantidade mínima de placas/classe. Além disso, foi tomado em conta que já existem trabalhos focados apenas em classificação de placas já recortadas (MISHRA; GOYAL, 2022; DALBORGO *et al.*, 2023), incluindo placas brasileiras, ou seja, ao obter uma caixa delimitadora correta, seria possível recortá-la e submetê-la a algum algoritmo de classificação já treinado para obter a nomenclatura da placa obtida.

Figura 14 – Exemplos de Placas de Trânsito Brasileiras

Fonte: Detran CE

4.4 Mineração de dados

Essa é a etapa que trata das técnicas aplicadas para extração de resultados dos dados obtidos. Neste trabalho isso é feito a partir de duas etapas: Treinamento dos modelos e ajuste para o conjunto de dados de placas de trânsito brasileiras e extração/anotação das métricas em tabelas

4.4.1 Treinamento/ajuste dos modelos

Essa etapa foi realizada a partir da utilização do Google Colab [3], uma ferramenta que permite executar códigos em linguagem de programação python através do acesso a maquinas virtuais armazenadas em nuvem. Nela é possível ter acesso a GPUs e TPUs potentes que aceleram significativamente o treinamento de modelos baseados em redes neurais. Além disso, essa ferramenta possui compatibilidade com o Google Drive [4], serviço de armazenamento de arquivos em nuvem, o que permite acessar repositórios virtuais contendo anotações, *checkpoints* do treinamento e bibliotecas essenciais para o treinamento, sem a necessidade de utilizar recursos do computador pessoal.

Para o treinamento dos modelos foi utilizada a versão paga Colab PRO, a qual disponibiliza maior acesso a mémoria e máquinas virtuais com GPUs e TPUs de última geração. Isso foi essencial, pois apenas com os recursos fornecidos pela versão gratuita não era possível treinar modelos para resoluções maiores que 640x640 pixels, devido a problemas de falta de memória.

4.4.2 Extração/anotação das métricas

Nesta etapa é importante ressaltar que além das métricas padrão obtidas pela própria API de treinamento, tanto do Tensorflow como do YOLO, este trabalho utiliza uma métrica que tem como objetivo medir a quantidade total de detecções de cada modelo. Essa métrica trata da observação da quantidade de placas detectadas levando em consideração os seguintes critérios:

- Score de confiança >= 25%;
- IoU >= 50%;
- placas grandes >= 1024 pixels de área;
- placas pequenas < 1024 pixels de área.

Encontrar o score de confiança adequado para se analisar determinado problema não é uma tarefa trivial. Neste trabalho foi optado por esse valor devido aos testes mostrarem que uma quantidade considerável de predições estavam na faixa de 25-50%, e também por dividir o valor máximo de 100% em quatro partes que corresponderiam logicamente ao número de classes. Assim, o que essa métrica faz é calcular o número de detecções verdadeiras obtidas dividido pelo número de placas total existentes.

[3] https://colab.google
[4] https://www.google.com/drive

Além disso, outras métricas são geradas pela *pipeline* de testes, como o número de falsos positivos, caixas duplicadas, e a velocidade de processamento média. Os testes são realizados em um ambiente de Processador AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx (8 CPUs), 2.1GHz, 16GB RAM, o que caracteriza tempos maiores que quando avaliados em um ambiente de GPU. Após coletados os resultados para cada modelo, essas métricas são então anotadas e reunidas em tabelas.

4.5 Interpretação

Essa é a etapa onde os resultados obtidos são analisados e discutidos, de modo a extrair conhecimento e informações. Neste trabalho os modelos são analisados para cada uma das métricas , focando em realizar comparações entre os modelos fornecidos pela plataforma do Tensorflow entre eles e com o YOLO.

O objetivo final da pesquisa é identificar qual(is) modelo(s) são mais adequados para a tarefa de detecção de placas de trânsito em imagens, além de prover informações e conhecimentos das respostas dos modelos em relação as métricas então analisadas.

5 RESULTADOS

Este capítulo apresenta os resultados obtidos a partir da pesquisa e traz uma série de análises comparativas entre os modelos utilizados. Esses resultados são necessários para compreender e extrair conclusões posteriores com relação a detecção de objetos aplicada a detecção de placas de trânsito. Este capítulo é dividido em seções, em que são discutidas as métricas mais relevantes, a partir de sub-tabelas extraídas da tabela base encontrada no apêndice D.

5.1 Modelos com maior detecção

Nesta seção são analisados os modelos que apresentam a maior porcentagem de detecção, ou seja que foram capazes de detectar a maior quantidade de placas com relação ao conjunto total de placas.

Tabela 1 – Tabela de modelos com maior % de placas detectadas

Modelos com maior % de detecção						
Modelo	Placas detectadas(VP)	% Total	Avg process time(s)	FP	% FP	AP
centernet_hg104_1024x1024	499	83,20%	47,52	99	16,56%	71,00%
efficientdet_d5_1280x1280	510	85,00%	17,26	52	9,25%	65,20%
ssd_resnet101_v1_fpn_1024x1024	503	83,83%	12,65	118	19,00%	80,10%
faster_rcnn_resnet101_v1_800x1333	539	89,83%	16,35	236	30,45%	70,20%
Yolov5m_640	446	74,33%	1,22	50	10%	81,20%

Fonte: elaborada pelo autor.

Dos modelos encontrados na API do Tensorflow, o que mostrou a maior porcentagem de detecção foi o Faster R-CNN, com 89,83%, construído sobre a arquitetura base da rede resnet101 para uma resolução de 800x1333 pixels. Porém esse modelo apresentou uma elevada porcentagem de falsos positivos, de 30,45%.

Já o segundo candidato, a rede EfficientDet, apresentou também um alto valor de detecção (85,00%) e uma baixa taxa de falsos positivos (9,25%) para um score confiança de 25%, porém uma precisão média (AP) de apenas 65% , indicando que, quando o score de confiança aumenta, há uma queda significativa na detecção de placas. Assim, dentre os modelos do Tensorflow, o que mostrou melhor desempenho com relação à detecção foi o SSD contruído sobre a resnet101 para uma resolução de 1024x1024 pixels, o qual possui uma porcentagem de detecção de 83,83% uma precisão média (AP) de 80,10%.

Já do YOLO o modelo que apresentou o maior número de placas detectadas foi YoloV5m para uma resolução de 640x640 pixels, com % total de detecção de 74,33% e AP de

81,20%. Assim, foi possível concluir, que o modelo com maior desempenho geral para a métrica de porcentagem de detecções totais foi o resnet101 para uma resolução de 1024x1024 pixels, com 9,5% a mais que o YOLOv5m (resolução 640x640 pixels).

5.2 Modelos com maior precisão média

A precisão média é uma métrica que tem como objeto analisar a quantidade de detecções verdadeiras com relação ao conjunto total de detecções. Nesta seção são observados os modelos que melhor atenderam a essa métrica.

Tabela 2 – Tabela de modelos com maior AP

Modelos com maior AP				
Modelo	AP	AP50	AP75	AR
centernet_hg104_1024x1024	71,00%	94,90%	87,00%	77,10%
efficientdet_d4_1024x1024	68,30%	93,80%	83,10%	73,70%
ssd_resnet101_v1_fpn_1024x1024	80,10%	96,20%	93,40%	84,40%
faster_rcnn_resnet50_v1_800x1333	70,60%	98,10%	85,20%	75,70%
Yolov5m_640	81,20%	96,40%		93,50%
Yolov5m6_1280	82,40%	98,40%		97,50%

Fonte: elaborada pelo autor.

Dentre os modelos da API do tensorflow, o que apresentou a maior AP foi o SSD construído sobre a rede resnet101 para uma resolução de 1024x1024 pixels com 80,10%. Já em relação ao YOLO, a arquitetura Yolov5m para uma resolução de 1280x1280 apresentou o maior AP com 82,4%.

5.3 Modelos com maior velocidade média de processamento

A velocidade de processamento é uma métrica bastante relevante para observar quais modelos são mais adequados para aplicações em tempo real. Nesta seção essa métrica é analisada para uma execução em imagens em um Processador AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx (8 CPUs), 2.1GHz, 16 GB RAM, sem levar em conta processamentos adicionais realizados para aplicações em vídeo.

Dentre as arquiteturas disponibilizadas na API do Tensorflow, é possível observar que a arquitetura SSD construída sobre a rede base MobileNet para uma resolução de 640x640 pixels é a que possui maior desempenho, possuindo a maior velocidade média, de 0.97 segundos, além da maior capacidade de detecção (64%) e precisão média (63,5%).

Já com relação à arquitetura YOLO, o modelo YoloV5n apresentou uma velocidade bastante rápida, de 0,25 segundos, e uma detecção razoável de 57,83%. Em comparação, o

Tabela 3 – Tabela de modelos com maior velocidade de processamento

Modelos mais rápidos com %detecção > 50%						
Modelo	Placas detectadas(VP)	% Total	Avg process time(s)	FP	% FP	AP
centernet_resnet50_v2_512x512	318	53,00%	1,35	298	48,38%	34,80%
efficientdet_d2_768x768	313	52,17%	2,8	62	16,53%	46,20%
ssd_mobilenet_v2_fpnlite_640x640	384	64,00%	0,97	86	18,30%	63,50%
faster_rcnn_resnet50_v1_640x640	358	59,67%	5,77	186	34,20%	42,50%
Yolov5n_640	347	57,83%	0,25	97	21,85%	75,70%
Yolov5m_640	446	74,33%	1,22	50	10%	81,20%

Fonte: elaborada pelo autor.

YoloV5m é mais lento com velocidade de processamento de 1,22 segundos, porém com uma detecção bem maior de 74,33% e precisão média de 81,2%.

5.4 Comparação de modelos para resolução 640x640

A resolução 640x640 é bastante relevante pois é utilizada como entrada para várias arquiteturas. Nesta seção, são comparados os modelos treinados para essa resolução tomando em conta como prioridade as métricas de porcentagem total de detecção e AP .

Tabela 4 – Tabela de modelos com resolução de entrada: 640x640

Comparação modelos para resolução 640x640								
Modelo	Placas detectadas(VP)	% Total	Avg process time(s)	FP	% FP	AP	AP50	AR
efficientdet_d1_640x640	290	48,33%	2,26	70	19,44%	37,60%	71,18%	48,50%
ssd_mobilenet_v1_fpn_640x640	379	63,17%	2,77	48	11,24%	68,90%	90,20%	75,40%
ssd_mobilenet_v2_fpnlite_640x640	384	64,00%	0,97	86	18,30%	63,50%	87,80%	70,80%
ssd_resnet50_v1_fpn_640x640	411	68,50%	4,1	54	11,61%	71,80%	92,30%	77,80%
ssd_resnet101_v1_fpn_640x640	409	68,17%	5,75	32	7,26%	72,00%	91,10%	77,90%
ssd_resnet152_v1_fpn_640x640	388	64,67%	7,56	37	8,70%	69,60%	90,00%	76,40%
faster_rcnn_resnet50_v1_640x640	358	59,67%	5,77	186	34,20%	42,50%	80,50%	53,90%
faster_rcnn_resnet101_v1_640x640	388	64,67%	7,38	139	26,38%	54,50%	84,70%	60,90%
faster_rcnn_resnet152_v1_640x640	380	63,33%	8,69	105	21,65%	57,00%	86,70%	63,60%
Yolov5n_640	347	57,83%	0,25	97	21,85%	75,70%	94,40%	87,70%
Yolov5m_640	446	74,33%	1,22	50	10%	81,20%	96,40%	93,50%

Fonte: elaborada pelo autor.

A partir da tabela 4, é possível observar que dentre os modelos da API do Tensorflow, o que apresentou o melhor desempenho foi o SSD construído sobre a arquitetura Resnet50, com porcentagem total de detecção de 68,5% e AP de 71,80%. Já em relação à arquitetura YoloV5m, observa-se bastante destaque em relação às demais, com porcentagem de detecção de 74,33% e AP de 81,20%

5.5 Comparação dos modelos com melhores resultados

A tabela 5 lista os modelos que apresentaram os melhores resultados com relação às métricas analisadas para a tarefa em questão. O modelo SSD construído sobre a rede resnet101 para uma resolução de 1024x1024 pixels se mostrou o mais adequado para uma aplicação que

desconsidera a necessidade de uma elevada velocidade de processamento. Já o Yolov5m para uma resolução de 640x640 se mostrou bastante eficiente para aplicações que necessitam de um modelo com uma velocidade de processamento elevada.

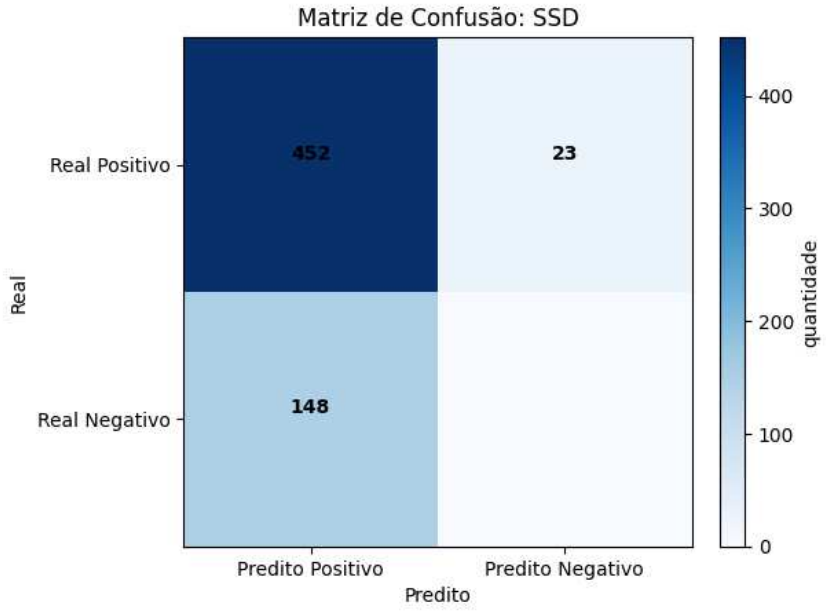
Tabela 5 – Tabela dos modelos com os melhores resultados nas métricas analisadas

Modelos com os melhores resultados							
Modelo	Placas detectadas(VP)	% Total	Avg process time(s)	FP	% FP	AP	AP50
ssd_resnet101_v1_fpn_1024x1024	503	83,83%	12,65	118	19,00%	80,10%	96,20%
Yolov5m6_1280	394	65,67%	1,42	32	7,51%	82,40%	98,40%
ssd_mobilenet_v2_fpnlite_640x640	384	64,00%	0,97	86	18,30%	63,50%	87,80%
Yolov5m_640	446	74,33%	1,22	50	10%	81,20%	96,40%

Fonte: elaborada pelo autor.

Outra análise que contribui para a compreensão dos resultados é a visualização da matriz de confusão desses modelos. Nas figuras 15 e 16 são apresentadas as matrizes de confusão para o modelo SSD e YOLO que apresentaram o melhor desempenho Nessa análise o score de confiança utilizado foi aumentado para 50%, com o objetivo de enriquecer as informações com relação ao funcionamento dos modelos.

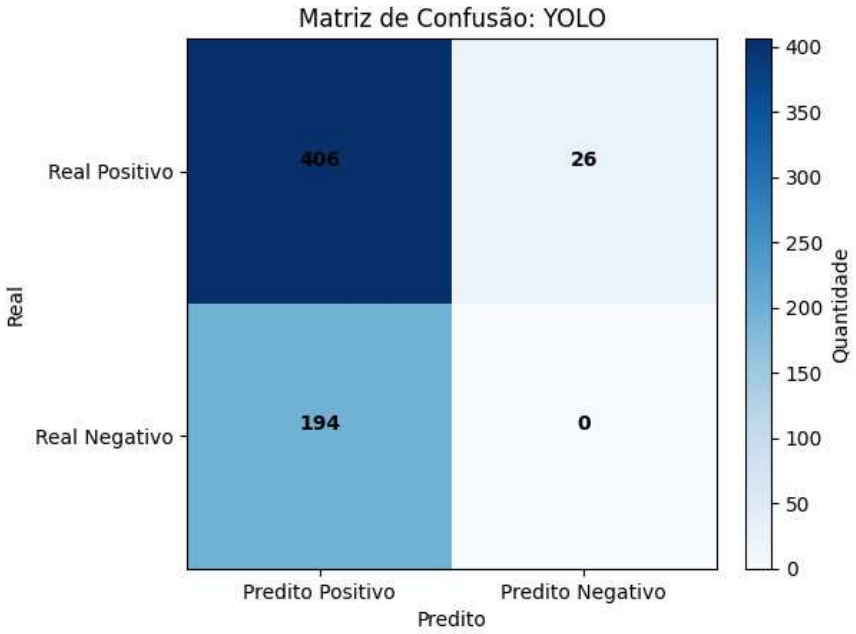
Figura 15 – Matriz de confusão para o SSD (resnet101, 1024x1024): Score de confiança 50%



Fonte: elaborada pelo autor.

A partir da observação da matriz de confusão, é possível a resposta do modelo para o aumento do score de confiança de 25% para 50%. No modelo SSD (resnet101, 1024x1024) é observada uma perda de 8,5% na precisão, caindo para 75,33%, porém com uma redução

Figura 16 – Matriz de confusão para o YOLO (V5m, 640x640): Score de confiança 50%



Fonte: elaborada pelo autor.

significativa de falsos positivos de 19,00% para 4,84%. Já no YOLO(V5m, 640x640) foi observada um perda de precisão de 6,67% caindo para 67,67%, e uma queda não tão expressiva dos falsos positivos como no caso anterior de 10,00% para 6,02%.

5.6 Considerações finais

Assim, a partir dos resultados obtidos das análises, foi possível observar que o modelo que apresentou o melhor desempenho para as principais métricas, de porcentagem de detecção e AP, independente da velocidade de processamento, foi o SSD construído sobre a rede resnet101 para uma resolução de 1024x1024 pixels. Já em relação a análise que toma a velocidade de processamento como significativamente importante o Yolov5m para uma resolução de 640x640 apresentou o melhor desempenho. Na tabela 6 é possível observar os resultados dos modelos citados referentes a essas métricas.

Tabela 6 – Resultados dos modelos YOLO(V5m, 640x640) e SSD (resnet101, 1024x1024)

Métricas	Modelo	% Detecção		AP	% FP		Processamento(seg)
		Score 25%	Score 50%		Score 25%	Score 50%	
% de detecção , AP	ssd_resnet101_v1_fpn_1024x1024	83,83%	75,33%	80,10%	19%	4,84%	12,65
Tempo de processamento	Yolov5m_640	74,33%	67,67%	81,20%	10%	6,02%	1,22

Fonte: elaborada pelo autor.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalhado foram apresentados alguns dos principais conceitos e definições que norteiam a detecção de objetos em imagens: as Redes Neurais Artificiais e suas limitações para tarefas de detecção de objetos; o funcionamento das Redes Neurais Convolucionais que são a base para os muitos dos modelos da literatura atual; uma série de arquiteturas de modelos para detecção de objetos (CenterNet, SSD, EfficientDet, Faster-RCNN e YOLO) e suas principais características; bem como as principais métricas utilizadas para avaliação de modelos de detecção, como IoU, AP, % total de detecções, etc.

Em seguida, foram realizadas análises para descobrir quais modelos obtiveram o melhor desempenho para cada métrica avaliada, com os resultados sendo anotados em tabelas. Das considerações finais, foram obtidos resultados que indicam que os modelos de um estágio SSD (Resnet101, 1024x2024 pixels) e YOLO (V5m, 640x640 pixels) possuem um desempenho bem maior com relação a outros modelos considerados para a detecção de placas de trânsito em imagens.

Assim, essa pesquisa atendeu aos objetivos iniciais de realizar uma análise comparativa de modelos de detecção de objetos aplicados ao problema de detecção de placas de trânsito, bem como de extrair informações e conhecimento com relação aos modelos que apresentaram o maior desempenho em relação ao conjunto total de modelos testados.

6.1 Trabalhos Futuros

É importante ressaltar que esse projeto possui uma série da fatores limitantes, pois o mesmo se baseia em uma análise dos modelos em sua arquitetura padrão. Isso significa que tomando em conta os resultados obtidos, é possível realizar modificações nos modelos que apresentaram os melhores resultados, acoplando camadas de processamento e utilizando-se de técnicas que enfoquem em otimizar fatores pontuais do problema em questão.

Um exemplo disso seria adotar técnicas de pré-processamento para normalizar imagens com relação á fatores como clima, iluminação, etc. Além disso, é possível conduzir modificações mais profundas, como apresentado nos trabalhos relacionados, de utilização de GANs, Camadas deconvolucionais ou FPNs modificadas. Salientando que este trabalho é apenas uma base para nortear a escolha de arquiteturas de modelos, as quais podem ser profundamente estudadas e aperfeiçoadas.

Outra possibilidade de trabalho futuro seria a otimização dos modelos que apresentaram o melhor desempenho, para aplicações em vídeo e dispositivos embarcados, essenciais para projetos relacionados à carros autônomos. Para isso seria necessário realizar modificações nas arquiteturas dos modelos para reduzir sua complexidade, de modo a aumentar a velocidade de processamento, e tamanho, para que seja possível o armazenamento e execução em dispositivos com menor capacidade de recursos computacionais. Existem algumas ferramentas que podem ser úteis para essa tarefa, como por exemplo, o Tensorflow Lite [1], uma ferramenta do próprio Tensorflow que tem como objetivo auxiliar na criação e implantação de modelos em dispositivos móveis, de borda e microcontroladores, para aplicações em tempo real. Além disso, o Tensorflow Lite possui suporte para conversão dos modelos do Tensorflow Model Zoo e YOLO, utilizados neste trabalho, para seu formato otimizado.

[1] https://www.tensorflow.org/lite

REFERÊNCIAS

ARCOS-GARCíA, ; ALVAREZ-GARCIA, J.; MORILLO, L. S. Evaluation of deep neural networks for traffic sign detection systems. **Neurocomputing**, v. 316, 08 2018.

CONTEXT, C. common objects in. **Avaliação de detecção**. [*s. n.*], 2017. Disponível em: https://cocodataset.org/#detection-eval. Acesso em: 01 oct. 2023.

DALBORGO, V.; MURARI, T. B.; MADUREIRA, V. S.; MORAES, J. G. L.; BEZERRA, V. M. O. S.; SANTOS, F. Q.; SILVA, A.; MONTEIRO, R. L. S. Traffic sign recognition with deep learning: Vegetation occlusion detection in brazilian environments. **Sensors**, v. 23, n. 13, 2023. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/23/13/5919.

DEEPIKA, N.; V, S. V. Obstacle classification and detection for vision based navigation for autonomous driving. In: . [*S. l.*: *s. n.*], 2017. p. 2092–2097.

DUAN, K.; BAI, S.; XIE, L.; QI, H.; HUANG, Q.; TIAN, Q. Centernet: Keypoint triplets for object detection. **CoRR**, abs/1904.08189, 2019. Disponível em: http://arxiv.org/abs/1904.08189.

DUBEY, V. **Evaluation Metrics for Object detection algorithms**. [*s. n.*], 2020. Disponível em: https://medium.com/@vijayshankerdubey550/evaluation-metrics-for-object-detection-algorithms-b0d6489879f3. Acesso em: 01 oct. 2023.

GENG, J. **How to Evaluate an Object Detection Model: Explain IoU, Precision, Recall and mAP with examples**. [*s. n.*], 2022. Disponível em: https://zihaogeng.medium.com/how-to-evaluate-an-object-detection-model-iou-precision-recall-and-map-f7cc12e0dcf6. Acesso em: 01 oct. 2023.

GIRSHICK, R. B.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. **CoRR**, abs/1311.2524, 2013. Disponível em: http://arxiv.org/abs/1311.2524.

HAKIM, H.; FADHIL, A. Survey: Convolution neural networks in object detection. **Journal of Physics: Conference Series**, v. 1804, p. 012095, 02 2021.

HANCOCK, J. Jaccard distance (jaccard index, jaccard similarity coefficient). In: ____. [*S. l.*: *s. n.*], 2004. ISBN 9780471650126.

HOSANG, J. H.; BENENSON, R.; SCHIELE, B. Learning non-maximum suppression. **CoRR**, abs/1705.02950, 2017. Disponível em: http://arxiv.org/abs/1705.02950.

HUANG, W.; HUANG, M.; ZHANG, Y. Detection of traffic signs based on combination of gan and faster-rcnn. **Journal of Physics: Conference Series**, v. 1069, p. 012159, 08 2018.

KHAZRI, A. **Faster RCNN Object detection**. [*s. n.*], 2019. Disponível em: https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4. Acesso em: 01 oct. 2023.

LAW, H.; DENG, J. Cornernet: Detecting objects as paired keypoints. **CoRR**, abs/1808.01244, 2018. Disponível em: http://arxiv.org/abs/1808.01244.

LEARNOPENCV. **Understanding Convolutional Neural Network (CNN): A Complete Guide**. [*s. n.*], 2023. Disponível em: https://learnopencv.com/understanding-convolutional-neural-networks-cnn/. Acesso em: 01 oct. 2023.

LIN, T.; DOLLÁR, P.; GIRSHICK, R. B.; HE, K.; HARIHARAN, B.; BELONGIE, S. J. Feature pyramid networks for object detection. **CoRR**, abs/1612.03144, 2016. Disponível em: http://arxiv.org/abs/1612.03144.

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

MATHEUS, C.; CHAN, P.; PIATETSKY-SHAPIRO, G. Knowledge discovery in databases. **IEEE Transactions on Knowledge and Data Engineering**, 12 1998.

MISHRA, J.; GOYAL, S. An effective automatic traffic sign classification and recognition deep convolutional networks. **Multimedia Tools and Applications**, v. 81, n. 13, p. 18915–18934, May 2022. ISSN 1573-7721. Disponível em: https://doi.org/10.1007/s11042-022-12531-w.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. **ArXiv e-prints**, 11 2015.

PADILLA, R.; NETTO, S.; SILVA, E. da. A survey on performance metrics for object-detection algorithms. In: . [*S. l.*: *s. n.*], 2020.

RAMIREZ, M. A. **3 Most Popular Data Science Methodologies**. [*s. n.*], 2021. Disponível em: https://medium.com/@aj.ramirez23/3-most-popular-data-science-metho-e61f6600b83f. Acesso em: 01 oct. 2023.

REDMON, J.; DIVVALA, S. K.; GIRSHICK, R. B.; FARHADI, A. You only look once: Unified, real-time object detection. **CoRR**, abs/1506.02640, 2015. Disponível em: http://arxiv.org/abs/1506.02640.

REN, S.; HE, K.; GIRSHICK, R. B.; SUN, J. Faster R-CNN: towards real-time object detection with region proposal networks. **CoRR**, abs/1506.01497, 2015. Disponível em: http://arxiv.org/abs/1506.01497.

ROSEBROCK, A. **Intersection over Union (IoU) for object detection**. [*s. n.*], 2016. Disponível em: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. Acesso em: 01 oct. 2023.

TAN, M.; LE, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. **CoRR**, abs/1905.11946, 2019. Disponível em: http://arxiv.org/abs/1905.11946.

TAN, M.; PANG, R.; LE, Q. V. Efficientdet: Scalable and efficient object detection. **CoRR**, abs/1911.09070, 2019. Disponível em: http://arxiv.org/abs/1911.09070.

VIGHNESHBIRODKAR; TEAM, T. O. D. **Tensorflow 2 Detection Model ZOO**. [*s. n.*], 2021. Disponível em: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md. Acesso em: 01 oct. 2023.

WANG, L.; SUN, P.; XIE, M.; MA, S.; LI, B.; SHI, Y.; SU, Q. Advanced driver-assistance system (adas) for intelligent transportation based on the recognition of traffic cones. **Advances in Civil Engineering**, Hindawi, v. 2020, p. 8883639, Jun 2020. ISSN 1687-8086. Disponível em: https://doi.org/10.1155/2020/8883639.

WANG, Y.; BAI, M.; WANG, M.; ZHAO, F.; GUO, J. Multiscale traffic sign detection method in complex environment based on yolov4. **Computational Intelligence and Neuroscience**, Hindawi, v. 2022, p. 5297605, Oct 2022. ISSN 1687-5265. Disponível em: https://doi.org/10.1155/2022/5297605.

WU, Y.; LIU, Y.; LI, J.; LIU, H.; HU, X. Traffic sign detection based on convolutional neural networks. In: . [*S. l.*: *s. n.*], 2013. p. 1–7. ISBN 978-1-4673-6128-6.

YUAN, Y.; XIONG, Z.; WANG, Q. VSSA-NET: vertical spatial sequence attention network for traffic sign detection. **CoRR**, abs/1905.01583, 2019. Disponível em: http://arxiv.org/abs/1905.01583.

APÊNDICE A – CÓDIGO A FUNÇÃO PARA OBTENÇÃO DAS CAIXAS DELIMITADORAS DOS ARQUIVOS DE ANOTAÇÕES EM XML

```
# Obtencao das caixas reais do arquivo de anotacoes em XML
# Entrada -> path para o xml file
# Saida   -> Lista de de tuplas para cada objeto contendo [ Label d do objeto , Dimensoes do bbox ]
def object_list_from_xml(path,atype):
    with open(path, 'r') as f:
        data = f.read()
    bs_data = BeautifulSoup(data,'xml')
    objects = bs_data.find_all('object')
    bboxes = []
    for obj in objects:
        lb = obj.find_all('name')[0].text
        xmin = int(obj.find_all('xmin')[0].text)
        ymin = int(obj.find_all('ymin')[0].text)
        xmax = int(obj.find_all('xmax')[0].text)
        ymax = int(obj.find_all('ymax')[0].text)
        dim = (abs(int(xmax) - int(xmin)) * abs(int(ymax) - int(ymin)))
        if atype == 'all':
            bboxes.append([lb,(xmin,ymin,xmax,ymax)])
        elif atype == 'big':
            if dim > 1024:
                bboxes.append([lb,(xmin,ymin,xmax,ymax)])
        elif atype == 'small':
            if dim <= 1024:
                bboxes.append([lb,(xmin,ymin,xmax,ymax)])
    return bboxes
```

APÊNDICE B – CÓDIGO A FUNÇÃO PARA OBTENÇÃO DAS CAIXAS DELIMITADORAS DOS MODELOS DO TENSORFLOW

```
# Entrada -> (Resultados do predict do modelo ( Boxes, Classes/Labels, Scores), Dicionario dos labels, Tamanho da img,
#             Score minimo)
# Saida   -> Lista de de tuplas para cada objeto contendo [ Label d do objeto , Dimensoes do bbox ]

def object_list_from_model(boxes,classes,scores,labels_dict,shape,min_score):
    idx = np.where(scores >= min_score)[0]
    if len(idx) == 0:
        return []
    bboxes = []
    for i in range(len(idx)):
        lb = labels_dict[classes[idx[i]]]
        box_factor = boxes[idx[i]]
        xmin = box_factor[0] * shape[0]
        ymin = box_factor[1] * shape[1]
        xmax = box_factor[2] * shape[0]
        ymax = box_factor[3] * shape[1]

        bboxes.append([lb,(int(ymin),int(xmin),int(ymax),int(xmax))])
    return bboxes
```

APÊNDICE C – CÓDIGO A FUNÇÃO PARA OBTENÇÃO DAS CAIXAS DELIMITADORAS DO MODELO YOLO

```
# Entrada -> (Resultados do predict padr o do YOLO
# Saida   -> Lista de de tuplas para cada objeto contendo [ Label d do objeto , Dimensoes do bbox ]
def object_list_from_model(detections,labels_dict,min_score):
    results = []
    for dtc in detections:
        if float(dtc[4]) < min_score:
            continue
        result = []
        bbox = tuple(dtc.numpy()[0:-2].astype('int'))
        label = labels_dict[int(dtc[5])]
        result.append(label)
        result.append(bbox)

        results.append(result)
    return results
```

APÊNDICE D – TABELA GERAL DE ANÁLISE DOS MODELOS DE DETECÇÃO DE OBJETOS

Neste apêndice é anexada a tabela em formato PDF da tabela geral com os resultados gerais obtidos para todos os modelos analisados.

Tabela Modelos de Detecção de Objetos

Modelo	Métricas Score 25%										Métricas padrão			
	Placas detectadas(VP)	Placas grandes	Placas pequenas	% Total	% Grandes	% Pequenas	Avg process time(s)	FP	Total Pred	% FP	Duplicatas	AP	AP50	AP75	AR
centernet_hg104_512x512	396	292	104	66,00%	77,66%	46,43%	12,3	233	629	37,04%	0	56,80%	85,40%	65,30%	66,70%
centernet_hg104_1024x1024	499	329	170	83,20%	87,50%	76,00%	47,52	99	598	16,56%	0	71,00%	94,90%	87,00%	77,10%
centernet_resnet50_v1_fpn_512x512	370	286	84	61,67%	76,10%	37,50%	1,58	246	616	39,94%	0	44,10%	77,80%	45,70%	54,70%
centernet_resnet101_v1_fpn_512x512	362	271	91	60,33%	72,07%	40,63%	2,48	224	586	38,22%	0	53,20%	83,30%	59,90%	62,30%
centernet_resnet50_v2_512x512	318	256	62	53,00%	68,10%	27,68%	1,35	298	616	48,38%	0	34,80%	62,00%	35,80%	48,90%
efficientdet_d0_512x512	266	223	43	44,33%	59,31%	19,20%	1,43	237	526	47,12%	23	23,70%	52,00%	18,50%	42,10%
efficientdet_d1_640x640	290	250	40	48,33%	66,50%	17,86%	2,26	70	374	19,44%	14	37,60%	71,18%	35,50%	48,50%
efficientdet_d2_768x768	313	280	33	52,17%	74,47%	14,73%	2,8	62	384	16,53%	9	46,20%	79,40%	49,80%	57,00%
efficientdet_d3_896x896	431	332	99	71,83%	88,30%	44,20%	5,35	103	552	19,29%	18	56,90%	88,50%	65,60%	64,50%
efficientdet_d4_1024x1024	451	337	114	75,17%	89,63%	50,89%	9,81	38	492	7,80%	3	68,30%	93,80%	83,10%	73,70%
efficientdet_d5_1280x1280	510	362	148	85,00%	96,28%	66,07%	17,26	52	569	9,25%	7	65,20%	90,02%	84,00%	73,60%
efficientdet_d6_1280x1280	409	318	91	68,17%	84,57%	40,63%	28,15	179	599	30,44%	11	44,50%	66,80%	55,50%	68,50%
ssd_mobilenet_v2_320x320	315	252	63	52,50%	67,02%	28,13%	0,41	1029	1379	76,56%	35	33,10%	65,60%	29,50%	43,60%
ssd_mobilenet_v1_fpn_640x640	379	301	78	63,17%	80,05%	34,82%	2,77	48	432	11,24%	5	68,90%	90,20%	79,20%	75,40%
ssd_mobilenet_v2_fpnlite_320x320	145	120	25	24,17%	31,91%	11,16%	0,56	74	229	33,80%	10	31,10%	57,00%	30,80%	48,80%
ssd_mobilenet_v2_fpnlite_640x640	384	299	85	64,00%	79,52%	37,95%	0,97	86	474	18,30%	4	63,50%	87,80%	74,90%	70,80%
ssd_resnet50_v1_fpn_640x640	411	299	112	68,50%	79,52%	50,00%	4,1	54	467	11,61%	2	71,80%	92,30%	82,70%	77,80%
ssd_resnet50_v1_fpn_1024x1024	436	299	137	72,67%	79,52%	61,16%	9,4	83	519	16,00%	0	78,80%	95,10%	91,80%	83,70%
ssd_resnet101_v1_fpn_640x640	409	306	103	68,17%	81,38%	46,00%	5,75	32	441	7,26%	0	72,00%	91,10%	84,60%	77,90%
ssd_resnet101_v1_fpn_1024x1024	503	344	159	83,83%	91,45%	70,98%	12,65	118	621	19,00%	0	80,10%	96,20%	93,40%	84,40%
ssd_resnet152_v1_fpn_640x640	388	289	99	64,67%	76,86%	44,20%	7,56	37	425	8,70%	0	69,60%	90,00%	82,30%	76,40%
ssd_resnet152_v1_fpn_1024x1024	432	297	135	72,00%	78,99%	60,27%	16,5	37	469	7,90%	0	77,20%	95,40%	92,40%	82,60%
faster_rcnn_resnet50_v1_640x640	358	277	81	59,67%	73,70%	36,16%	5,77	186	547	34,20%	3	42,50%	80,50%	43,10%	53,90%
faster_rcnn_resnet50_v1_1024x1024	419	276	143	69,83%	73,40%	63,84%	8,84	150	604	26,36%	35	55,10%	94,30%	58,30%	64,10%
faster_rcnn_resnet50_v1_800x1333	491	317	174	81,83%	84,30%	77,68%	11,15	168	659	25,49%	0	70,60%	98,10%	85,20%	75,70%
faster_rcnn_resnet101_v1_640x640	388	291	97	64,67%	77,39%	43,30%	7,38	139	529	26,38%	2	54,50%	84,70%	64,00%	60,90%
faster_rcnn_resnet101_v1_1024x1024	467	312	155	77,83%	82,98%	69,20%	12,5	101	568	17,78%	0	64,50%	95,60%	77,60%	71,60%
faster_rcnn_resnet101_v1_800x1333	539	347	192	89,83%	92,29%	85,71%	16,35	236	775	30,45%	0	70,20%	97,70%	84,60%	75,50%
faster_rcnn_resnet152_v1_640x640	380	292	88	63,33%	77,66%	39,29%	8,69	105	486	21,65%	1	57,00%	86,70%	67,60%	63,60%
faster_rcnn_resnet152_v1_1024x1024	403	266	137	67,17%	70,74%	61,16%	15,46	69	472	14,62%	0	68,80%	96,60%	87,50%	74,40%
faster_rcnn_resnet152_v1_800x1333	474	297	177	79,00%	78,99%	79,02%	19,94	236	775	30,45%	0	67,40%	97,50%	83,10%	73,90%
Yolov5n_640	347	242	105	57,83%	64,36%	46,88%	0,25	97	444	21,85%	0	75,70%	94,40%		87,70%
Yolov5m_640	446	306	140	74,33%	81,38%	62,50%	1,22	50	496	10%	0	81,20%	96,40%		93,50%
Yolov5m6_1280	394	287	107	65,67%	76,33%	47,77%	1,42	32	426	7,51%	0	82,40%	98,40%		97,50%