

PAULO SÉRGIO DA SILVA FREITAS JÚNIOR

**DETECÇÃO DE SEMÁFOROS EM AMBIENTES URBANOS:
UMA ANÁLISE COMPARATIVA DAS R-CNNs YOLO**

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Sistemas de Informação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador(a): Prof. Dra. Elloá B. Guedes

Manaus – Março – 2023

Universidade do Estado do Amazonas - UEA

Escola Superior de Tecnologia - EST

Reitor:

André Luiz Nunes Zogahib

Vice-Reitor:

Kátia do Nascimento Couceiro

Diretor da Escola Superior de Tecnologia:

Ingrid Sammyne Gadelha Figueiredo

Coordenador do Curso de Sistemas de Informação:

Marcela Sávia Picanço Pessoa

Coordenador da Disciplina Projeto Final:

Polianny Almeida Lima

Banca Avaliadora composta por:

Data da Defesa: 28/03/2023.

Prof. Dra. Elloá Barreto Guedes da Costa (Orientadora)

Prof. Esp. Giovana Oliveira de Lucca

Prof. Dr. Carlos Maurício Serório Figueiredo

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).
Sistema Integrado de Bibliotecas da Universidade do Estado do Amazonas.

S484dd Freitas Júnior, Paulo Sérgio da Silva
Detecção de Semáforos em Ambientes Urbanos : Uma
Análise Comparativa das R-CNNs YOLO / Paulo Sérgio
da Silva Freitas Júnior. Manaus : [s.n], 2023.
35 f.: color.; 29 cm.

TCC - Graduação em Sistemas de Informação -
Bacharelado - Universidade do Estado do Amazonas,
Manaus, 2023.
Inclui bibliografia
Orientador: Prof. Dra. Elloá B. Guedes

1. Semáforos. 2. Aprendizado Profundo. 3. Detecção
de Objetos. 4. YOLO. I. Prof. Dra. Elloá B. Guedes
(Orient.). II. Universidade do Estado do Amazonas. III.
Detecção de Semáforos em Ambientes Urbanos

Elaborado por Jeane Macelino Galves - CRB-11/463



FOLHA DE APROVAÇÃO

DETECÇÃO DE SEMÁFOROS EM AMBIENTES URBANOS: UMA ANÁLISE COMPARATIVA DAS R-CNNs YOLO

PAULO SERGIO DA SILVA FREITAS JÚNIOR

Trabalho de Conclusão de Curso defendido e aprovado pela banca avaliadora constituída pelos professores:

Prof. Dra. Elloá Barreto Guedes da Costa
Presidente

Profa. Esp. Giovana Oliveira de Lucca
Avaliador

Prof. Dr. Carlos Maurício Seródio Figueiredo
Avaliador

Manaus, 28 de março de 2023.

Resumo

Este trabalho de conclusão de curso contempla o reconhecimento de semáforos urbanos realizando uma análise comparativa de desempenho dos modelos das famílias YOLOv5, YOLOv7 e YOLOv8. Para tal, foram utilizados dois conjuntos de dados: Bosch Small Traffic Lights Dataset e BOSCHv2, sendo o primeiro composto por 13.425 imagens e 24.000 anotações referentes aos semáforos captados nas áreas de Fracisco Bay Area e Palo Alto, Califórnia e o segundo sendo um derivativo com apenas anotações de semáforos consideradas médias e grandes. Os experimentos realizados com validação cruzada holdout evidenciaram que os modelos YOLOv8 Nano e YOLOv8 Small no dataset BOSCHv2 obteram os melhores desempenhos, com a métrica mAP@0.5 de 0.826 e 0.851, respectivamente. As descobertas encontradas mostram haver margens para a exploração de arquiteturas variadas pertencentes a família YOLO, visto que os modelos explorados obtiveram um bom desempenho na tarefa de detecção semafórica mesmo com uma redução drástica dos dados para treinamento.

Palavras-Chave: Semáforos; Aprendizado Profundo; Detecção de Objetos, YOLO.

Abstract

This undergraduate thesis addresses the detection of urban traffic lights by performing a comparative analysis of the model performance from the YOLOv5, YOLOv7, and YOLOv8 models. For this purpose, two sets of data were used: Bosch Small Traffic Lights Dataset and BOSCHv2, the first consisting of 13,425 images and 24,000 annotations regarding traffic lights captured in the Francisco Bay Area and Palo Alto, California, and the second consisting of images and annotations from first dataset filtering only medium and large size traffic light annotation's. The experiments performed using holdout cross-validation showed that the YOLOv8 Nano and YOLOv8 Small models in the BOSCHv2 dataset had the best performance, with the mAP@0.5 metric of 0.826 and 0.851, respectively. The findings show that there is room for exploration of different architectures belonging to the YOLO model since the explored models obtained a good performance in the traffic light detection task even with a drastic reduction of training data.

Keywords: Traffic Lights; Deep Learning; Object Detection, YOLO.

Sumário

Lista de Tabelas	vii
Lista de Figuras	ix
1 Introdução	1
1.1 Objetivos	3
1.2 Justificativa	4
1.3 Metodologia	4
1.4 Organização do Documento	5
2 Fundamentação Teórica	6
2.1 Fundamentos de <i>Deep Learning</i>	6
2.2 Família YOLO	8
2.3 Métricas de Desempenho em Detecção de Objetos	15
2.4 Trabalhos Relacionados	17
3 Materiais e Métodos	20
3.1 Dados Experimentais	20
3.2 Modelos, Parâmetros e Hiperparâmetros	27
3.3 Avaliação de Desempenho	28
3.4 Tecnologias Utilizadas	29
4 Resultados	30

Lista de Tabelas

3.1	Distribuição dos dados no BOSCH <i>dataset</i>	20
3.2	Distribuição de rótulos na partição de treinamento do BOSCH.	24
3.3	Distribuição de rótulos na partição de testes do BOSCH.	24
3.4	Tamanhos das <i>bounding boxes</i> do conjunto de teste em pixels.	24
3.5	Tamanhos das <i>bounding boxes</i> do conjunto de treino em pixels.	24
3.6	Distribuição dos dados nas partições de treino e validação no <i>dataset</i> BOSCHv1. .	25
3.7	BOSCHv1 – Detalhamento dos semáforos quanto ao tamanho na partição de treinamento.	26
3.8	BOSCHv2 – Tamanhos dos rótulos (em px) da partição de treinamento.	26
3.9	BOSCHv2 – Tamanhos dos rótulos (em px) da partição de validação.	27
3.10	BOSCHv2 – Detalhamento dos semáforos quanto ao tamanho na partição de treinamento.	27
3.11	Descrição dos parâmetros das arquiteturas Yolov5's utilizadas.	27
3.12	Descrição dos parâmetros das arquiteturas YOLOv7's utilizadas.	28
3.13	Descrição dos parâmetros das arquiteturas YOLOv8's utilizadas.	28
4.1	Síntese dos resultados experimentais.	30

Lista de Figuras

2.1	Extração de características em uma CNN. Fonte: (KHAN et al., 2018)	7
2.2	Componentes de uma R-CNN. Fonte: (KHAN et al., 2018)	8
2.3	Abordagem da YOLO para detecção de objetos. Fonte: (REDMON et al., 2016) .	9
2.4	Arquitetura da CNN utilizada na primeira versão da YOLO. Fonte: (REDMON et al., 2016)	10
2.5	Comparação da Yolov4 com outras algoritmos de detecção de objetos no <i>dataset</i> MS COCO Fonte: (BOCHKOVSKIY; WANG; LIAO, 2020)	12
2.6	Desempenho Yolov5 no <i>dataset</i> MS COCO Fonte: (JOCHER et al., 2020)	13
2.7	Comparação da Yolov7 com outras algoritmos de detecção de objetos no <i>dataset</i> MS COCO Fonte: (WANG; BOCHKOVSKIY; LIAO, 2022a)	14
2.8	Comparação da Yolov8 com outras algoritmos de detecção de objetos no <i>dataset</i> MS COCO Fonte: (JOCHER et al., 2022)	15
2.9	Representação gráfica do cálculo de IoU (PADILLA; NETTO; SILVA, 2020) . .	15
3.1	Exemplos de imagens do <i>dataset</i> BOSCH Traffic Lights ilustrando as classes: Green, GreenLeft, GreenStraightLeft,GreenStraightRight. Fonte: (BEHRENDT; NOVAK,)	21
3.2	Exemplos de imagens do <i>dataset</i> BOSCH Traffic Lights ilustrando as classes: Red, GreenRight, off,GreenStraight. Fonte: (BEHRENDT; NOVAK,)	22
3.3	Exemplos de imagens do <i>dataset</i> BOSCH Traffic Lights ilustrando as classes: RedLeft, RedRight, RedStraight,RedStraightLeft, Yellow. Fonte: (BEHRENDT; NOVAK,)	23

3.4	Anotação do <i>Bosch Small Traffic Lights Dataset</i>	25
3.5	Exemplo de anotação em padrão YOLO.	25
4.1	YOLOv8 <i>Small</i> – BOSCHv2 – Matriz de confusão	31
4.2	YOLOv8 <i>Medium</i> – BOSCHv2 – Matriz de confusão	32
4.3	YOLOv8 <i>Nano</i> – BOSCHv2.	33
4.4	YOLOv8 <i>Nano</i> – BOSCHv2.	33
4.5	YOLOv8 <i>Small</i> – BOSCHv2.	33
4.6	YOLOv8 <i>Small</i> – BOSCHv2.	34

Capítulo 1

Introdução

Cidades Inteligentes (CIs) são caracterizadas pelo uso estratégico, sistemático e coordenado de Tecnologias da Informação e Comunicação em uma variedade de contextos urbanos, objetivando satisfazer de forma abrangente a necessidade dos seus cidadãos alcançando a sustentabilidade econômica, social e ambiental (SONG et al., 2017). Sob diversas perspectivas, uma CI se propõe a otimizar o uso dos recursos e infraestrutura de uma forma sustentável com vistas a melhorar a qualidade de vida da população que nela habita (KON; SANTANA, 2016).

Giffinger definem seis dimensões a serem consideradas para mensurar o grau de desenvolvimento de uma CI, as quais consideram economia (empresas instaladas, ambiente para empreendedorismo, etc.), população (educação, emprego, renda, etc.), governança (transparência de órgãos públicos, facilidade no uso de serviços públicos, etc.), meio-ambiente (poluição ambiental, eficiência no uso da água, etc.), vida inteligente (entretenimento, segurança, áreas verdes, bibliotecas, etc.) e mobilidade. No tocante à mobilidade, em especial, menciona-se a facilidade de usar diferentes modais de transporte (bicicleta, carro, ônibus, metrô, etc.), quantidade e tamanho de congestionamentos e aplicações para facilitar e incentivar o uso de transporte público e sustentável, dentre outros (GIFFINGER et al., 2007).

Apesar dos progressos na Mobilidade Urbana, problemas elementares ainda são prevalentes, tais como os danos causados ao meio ambiente em razão da emissão de poluentes bem como os impactos na qualidade de vida, tais como o tempo de comutação, a segurança e o conforto. Na evolução para uma Mobilidade Inteligente, a participação humana ativa é gradativamente

substituída por sistemas e tecnologias inteligentes que colaboram para uma autonomia completa de transporte que, em última instância, tornam a função do motorista humano obsoleta (KIRWAN; ZHIYONG, 2020). Neste percurso evolutivo é natural, portanto, pensar primeiramente em soluções inteligentes que apoiem as pessoas para uma mobilidade mais eficiente e segura.

A detecção de semáforos, por exemplo, é uma tarefa importante para sistemas de assistência ao motorista e também para o desenvolvimento de veículos autônomos (KIM; PARK; JUNG, 2018). Esta tarefa reside no âmbito da Visão Computacional (VC) e compreende a localização do semáforo e a classificação do seu estado, as quais são essenciais para uma posterior tomada de decisão. Há que se salientar que em contextos realistas há diversos desafios que não podem ser ignorados no desenvolvimento de soluções inteligentes para a detecção de semáforos, os quais compreendem variações no tamanho, posicionamento e luminosidade, condições de tempo e até mesmo ambiguidades com outros objetos urbanos (BEHRENDT; NOVAK; BOTROS, 2017).

Diversas soluções na literatura para o problema de detecção de semáforos utilizando diferentes estratégias para abordar a tarefa. Os primeiros trabalhos consideraram técnicas clássicas de VC para extração de características, tais como forma e cor, seguidas da classificação com modelos de *Machine Learning*. Tais soluções mostram-se limitadas ao detectarem apenas semáforos de tamanhos e modelos específicos sob *backgrounds* pré-determinados, por exemplo (DIAZ et al., 2015). Com o advento de *Deep Learning*, houve uma melhora no desempenho dessa tarefa sob condições menos controladas, considerando o paradigma do Aprendizado Supervisionado e arquiteturas de Redes Neurais Convolucionais Profundas (CNNs, do inglês *Convolutional Neural Networks*) e CNNs Regionais (R-CNNs, do inglês *Regional CNNs*) (POSSATTI et al., 2019).

Em problemas práticos de detecção de objetos em tempo real, as R-CNNs da família YOLO (acrônimo para *You Only Look Once*), em particular, tem se mostrado eficientes a partir de exemplos oriundos de diferentes bases de dados (BEHRENDT; NOVAK; BOTROS, 2017; JENSEN; NASROLLAHI; MOESLUND, 2017). Uma das explicações para este bom desempenho deve-se ao fato de tal família de modelos utilizar a abordagem *single-shot* para detecção, em

que as caixas delimitadoras (*bounding boxes*) dos objetos são previstas simultaneamente com a classificação associada (REDMON et al., 2016).

No ano de 2021, discentes do Laboratório de Sistemas Inteligentes obtiveram o segundo lugar nacional no *Data Science Challenge* 2021 promovido pelo Instituto Tecnológico de Aeronáutica, cuja terceira etapa envolveu a detecção automática de semáforos (YONEKURA; OLIVEIRA; MENDES, 2021). A solução proposta pelo LSI *Data Team* fez uso da YOLOv5 (JOCHER et al., 2020), uma das atualizações contínuas na literatura desde a proposição original da YOLO. Desde então, foram propostas a YOLOv7 (WANG; BOCHKOVSKIY; LIAO, 2022b) e a YOLOv8 (JOCHER et al., 2022), e é natural conjecturar se o desempenho nesta tarefa pode ser melhorado em consequência do uso de tais modelos. Essa é a perspectiva experimental que foi considerada no escopo deste trabalho de conclusão de curso, incluindo também o esclarecimento das melhorias técnicas e tecnológicas dessas R-CNNs da família YOLO recém propostas na literatura.

1.1 Objetivos

O objetivo geral deste trabalho consistiu em analisar comparativamente as arquiteturas de R-CNNs da Família YOLO na detecção de semáforos em ambientes urbanos. Para alcançar esta meta, alguns objetivos específicos foram consolidados, a citar:

1. Fundamentação de conceitos essenciais de *Deep Learning* para a tarefa de detecção de objetos com R-CNNs da Família YOLO;
2. Preparação, customização e otimização de uma base de dados disponível na literatura para fonte de experiência na tarefa em questão;
3. Condução e elaboração de estudo de caso experimental para treinamento e avaliação comparativa das R-CNNs YOLO;
4. Análise e avaliação de maneira qualitativa e quantitativa dos resultados obtidos.

1.2 Justificativa

O reconhecimento de semáforos é uma tarefa importante para o desenvolvimento de carros autônomos, sistemas de assistência ao motorista e até mesmo para tecnologias assistivas para pessoas com deficiência visual. Todas essas aplicações colaboram para melhorias na mobilidade em Cidades Inteligentes.

Considerando que as técnicas de *Deep Learning* são emergentes, é importante propor trabalhos que possam ajudar a verificar a adequação destas soluções, indicando vantagens e limitações, bem como comparações com o estado da arte. Em particular, neste trabalho será considerado um problema com relevância prática e com dados reais.

Além disto, na perspectiva do bacharel em Sistemas de Informação em formação, o desenvolvimento deste trabalho contribui para a aquisição de conhecimentos sobre *Deep Learning*, uma área de vanguarda, muito dinâmica e cujo potencial está em constante desenvolvimento. Por fim, deve-se mencionar a importância da realização deste trabalho com vistas a colaborar com as atividades desenvolvidas pelo LSI, uma iniciativa do Grupo de Pesquisas em Sistemas Inteligentes da Escola Superior de Tecnologia (EST) da UEA.

1.3 Metodologia

A metodologia utilizada para o desenvolvimento das atividades presentes neste trabalho, com o intuito de atingir os objetivos especificados, é descrita a seguir:

1. Estudo de conceitos relacionados às R-CNNs da família YOLO;
2. Coleta de diferentes bases de dados disponíveis na literatura com exemplos e rótulos para o problema de detecção de semáforos;
3. Realização de análise exploratória das bases de dados para síntese de suas principais características no tocante ao problema considerado;
4. Definição de uma estratégia de validação cruzada para o estudo de caso experimental;

5. Definição das R-CNNs da família YOLO a serem consideradas para os experimentos, bem como seus parâmetros e hiperparâmetros;
6. Preparação dos dados e do ambiente computacional para condução do estudo de caso;
7. Treinamento dos modelos;
8. Aferição das métricas de desempenho no teste dos modelos;
9. Escrita da proposta de Trabalho de Conclusão de Curso;
10. Defesa da proposta de Trabalho de Conclusão de Curso;
11. Escrita do Trabalho de Conclusão de Curso;
12. Defesa do Trabalho de Conclusão de Curso;

1.4 Organização do Documento

Este documento está organizado da seguinte forma. O Capítulo 2 engloba os fundamentos teóricos que foram necessários para o desenvolvimento do projeto, incluindo conceitos de Redes Neurais Convolucionais Regionais, os modelos utilizados da Família YOLO e também uma análise dos trabalhos relacionados. A descrição dos dados experimentais juntamente com a análise dos modelos a serem comparados encontram-se no Capítulo 3. Os resultados obtidos são explanados durante o Capítulo 4. Por fim, as considerações finais do trabalho encontram-se no Capítulo 5.

Capítulo 2

Fundamentação Teórica

Nas seções a seguir encontra-se uma breve apresentação dos conceitos que fundamentaram a realização deste trabalho. Uma visão geral sobre *Deep Learning* aplicado à tarefa de detecção de objetos com R-CNNs encontra-se disponível na Seção 2.1; a Seção 2.2 detalha as arquiteturas de R-CNNs da Família YOLO, utilizadas no escopo deste trabalho, a abordagem utilizada pelas mesmas para detecção de objetos e melhorias efetuadas na evolução para versões; métricas de desempenho na seção 2.3; e, por fim, os trabalhos relacionados encontram-se dispostos e analisados na Seção 2.4.

2.1 Fundamentos de *Deep Learning*

Inteligência Artificial (IA) é uma área abrangente da Computação, a qual engloba as subáreas de *Machine* e *Deep Learning*, nas quais modelos computacionais são desenvolvidos com o intuito de automatizar tarefas que exigiriam esforço intelectual e expertise quando normalmente desempenhada por seres humanos (CHOLLET, 2017).

A subárea de *Machine Learning* (ML), em particular, contempla algoritmos computacionais que aprendem, de forma não explícita, características de um conjunto de dados. Essa área possui como objetivo desenvolver métodos que possibilitem que estes modelos consigam aprender de forma automática usando observações extraídas dos dados de treinamento, sem a definição de regras explícitas por um especialista humano, ou supervisor. *Deep Learning* (DL), por sua vez,

é uma subárea de ML, na qual os modelos aprendem padrões complexos de um conjunto de dados de forma incremental, sequencial e hierárquica por meio de uma camadas. As técnicas de DL constituem atualmente o estado da arte nas tarefas de classificação e detecção de objetos em imagens, sendo utilizadas no reconhecimento de padrões complexos em diagnósticos médicos e tradução de idiomas, por exemplo (KHAN et al., 2018; CHOLLET, 2017).

As Redes Neurais Convolucionais (CNNs, do inglês *Convolutional Neural Networks*) são uma categoria de Redes Neurais Artificiais amplamente utilizada para extração de características em dados multidimensionais, tais como imagens, vídeo e áudio. Funcionam de maneira similar às Redes Neurais Artificiais mediante Aprendizado Supervisionado, mas cada camada em uma CNN possui um ou mais filtros que, por meio da operação de convolução sobre a entrada, permitem a extração de mapas de características, os quais serão utilizados para o aprendizado dos padrões (KHAN et al., 2018). Os parâmetros dos filtros podem ser compartilhados, o que reduz a quantidade de variáveis a serem ajustadas, e aplicação sucessiva dos filtros organizados em camadas permitem a extração de características sucessivamente mais complexas, conforme ilustrado na Figura 2.1.

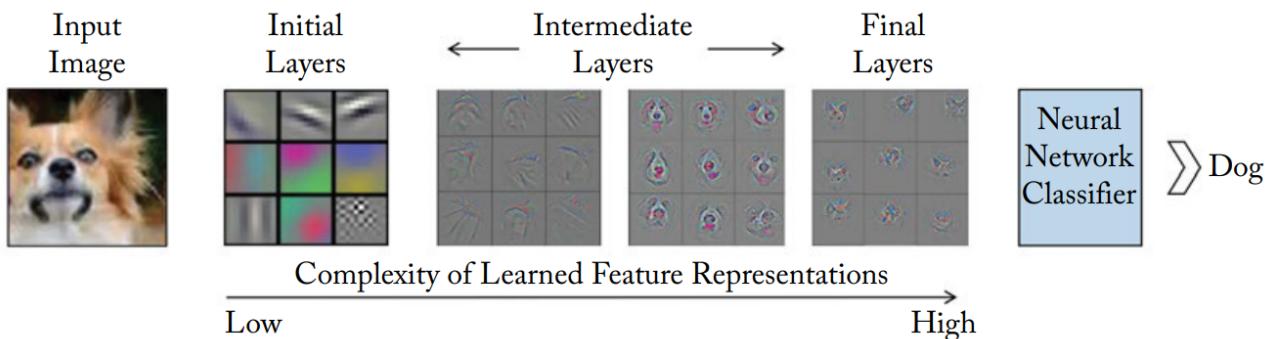


Figura 2.1: Extração de características em uma CNN. Fonte: (KHAN et al., 2018).

No contexto de Visão Computacional, a localização e a detecção de objetos são tarefas consideravelmente mais complexas que classificação, pois além de atribuírem um rótulo à imagem devem prever caixas delimitadoras a respeito da localização dos mesmos na entrada. Para lidar com esta tarefa, foram propostas arquiteturas de Redes Neurais Convolucionais Regionais (R-CNNs, do inglês *Regional Convolutional Neural Networks*) (GIRSHICK, 2015), as quais abordam a tarefa em três etapas ou módulos, conforme ilustrado na Figura 2.2.

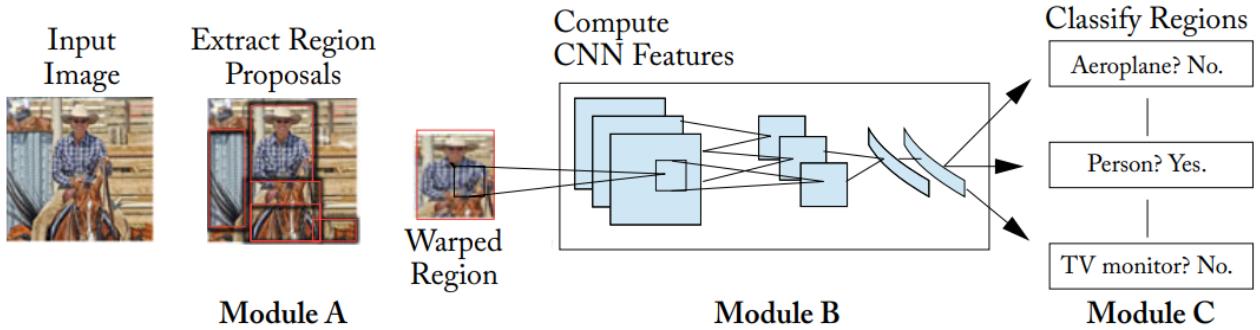


Figura 2.2: Componentes de uma R-CNN. Fonte: (KHAN et al., 2018).

O primeiro módulo, denominado Módulo A, é responsável por gerar propostas de regiões por meio de um algoritmo de busca seletiva. Este algoritmo segmenta a imagem em regiões que representam possíveis objetos a serem detectados pelo algoritmo detector. Após a segmentação, regiões adjacentes são agrupadas por critérios de similaridade e então constituem as chamadas *region of proposals*. Em seguida, o Módulo B, composto por uma CNN (e.g., AlexNet, VGGNet, etc.), é responsável pela extração de características de cada uma das regiões propostas pelo módulo anterior, produzindo um vetor. O terceiro e último módulo, denominado Módulo C, recebe as características extraídas do segundo módulo e, para cada classe, utiliza uma Máquina de Vetores de Suporte previamente treinada que então classifica os objetos que foram previamente localizados (KHAN et al., 2018)..

Apesar da arquitetura R-CNN descrita possuir uma boa acurácia na detecção de objetos segundo seus proponentes, apresentam algumas desvantagens práticas, tais como o alto custo computacional, devido à passagem de cada região proposta pelo módulo de extração de características, e a necessidade de um armazenamento considerável para cada característica extraída. Esse alto custo implica em um ônus de treinamento e em uma baixa velocidade de detecção para aplicações de tempo real.

2.2 Família YOLO

A Família YOLO (acrônimo para *You Only Look Once*) provê uma abordagem para detecção de objetos como um problema de regressão no qual uma única CNN é utilizada para prever as

coordenadas das caixas delimitadoras e suas respectivas classes de forma unificada (abordagem *single-shot*) (REDMON et al., 2016). Em razão dessa estratégia, tais redes conseguem lidar problemas de detecção de objetos em imagens de forma extremamente rápida e eficaz, atingindo mais que o dobro da média em precisão quando comparada a outros sistemas de detecção em tempo real (MICHELucci, 2019).

Conforme ilustrado na Fig. 3.5, a abordagem de detecção de objetos por R-CNNs YOLO consiste nos seguintes passos: primeiramente há a divisão da imagem de entrada em uma grade de dimensões $S \times S$; em seguida, para cada célula, há uma verificação se há um objeto cujo centro é englobado pela mesma; para as células em que tal verificação foi afirmativa, calculase a quantidade B de caixas delimitadoras com seus respectivos coeficientes de confiança – calculados por meio da multiplicação da métrica IoU (do inglês, *Intersection Over Union*) com a probabilidade da célula conter um certo objeto de uma dada classe – refletindo numericamente o quanto acurada está uma dada caixa delimitadora que contém um certo tipo de objeto; por fim, células adjacentes com alta confiança para um mesmo tipo de objeto são unificadas, culminando na previsão das coordenadas das caixas e seus respectivos rótulos.

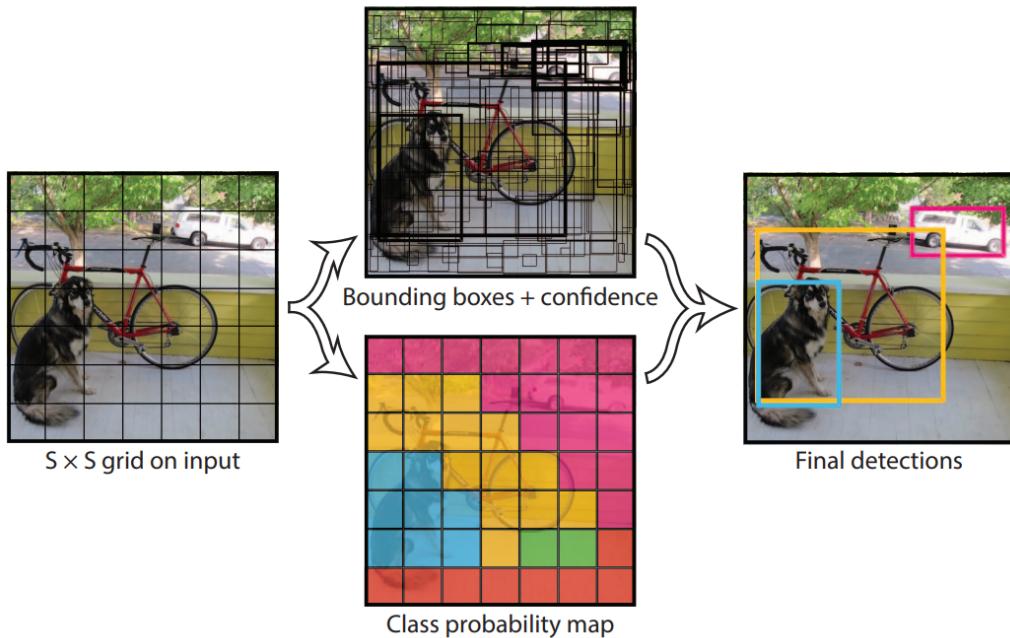


Figura 2.3: Abordagem da YOLO para detecção de objetos. Fonte: (REDMON et al., 2016).

A arquitetura utilizada na CNN da primeira versão do YOLO foi inspirada na classificadora GoogLeNet (Inception) pré-treinada com pesos oriundos da base de dados ImageNet (DENG et al., 2009). Esta rede é constituída por 20 camadas convolucionais seguidas por 2 camadas totalmente conectadas. Ao invés de usar os módulos Inception originais, optou-se por utilizar camadas de *pooling* com filtros de dimensões 1×1 para redução de dimensionalidade seguidos de camadas convolucionais com filtros 3×3 , conforme ilustrado na Fig. 2.6.

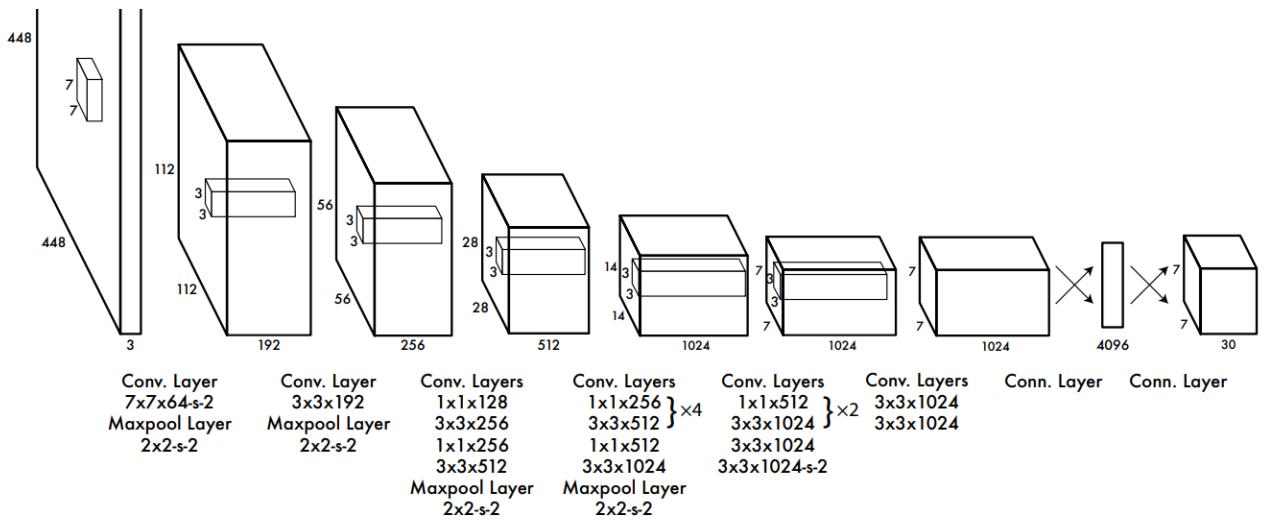


Figura 2.4: Arquitetura da CNN utilizada na primeira versão da YOLO.
Fonte: (REDMON et al., 2016).

Conforme os avanços na experimentação da primeira versão do YOLO, possíveis fragilidades e pontos de melhoria foram sendo descobertos, como por exemplo a baixa precisão da localização de objetos, esses pontos foram motivadores para a evolução do YOLO, resultando em novas versões da arquitetura (JIAO et al., 2019).

A YOLOv2, proposta em 2016, considera várias melhorias em relação à sua primeira versão, a citar: adição de camadas de *batch normalization*, as quais facilitam a convergência da rede e ajudam a regularizar o modelo, prevenindo *overfitting*; utilização de um classificador de alta resolução, com dimensões 448×448 que é treinado por 10 épocas no *dataset* ImageNet para ajuste fino de parâmetros; enquanto a YOLO utiliza as coordenadas previstas das *bounding boxes* para classificação do objeto, a YOLOv2 utiliza convoluções com *anchor boxes* nesta classificação, o que provoca uma melhora na revocação do modelo; e, por fim, a combinação de

características de alta e baixa resolução extraídas da imagem (REDMON; FARHADI, 2017). Além disso, a YOLOv2 propõe um novo *backbone* de classificação: uma rede denominada Darknet-19, com 19 camadas convolucionais e 5 camadas de *max-pooling* que requerem menos operações para processar uma imagem, mas ainda atingindo alta precisão (JIAO et al., 2019).

A Yolov3 implementa uma série de melhorias em relação a sua antecessora, Yolov2, a principal é a mudança na abordagem em relação a extração de características, onde a nova rede denominada DarkNet-53 é composta por 53 camadas convolucionais mais camadas residuais, sendo uma abordagem híbrida do DarkNet-19, porém mais eficiente e robusta (REDMON; FARHADI, 2018). Além disso, a última camada convolucional prevê um tensor tridimensional que codifica as *bounding boxes*, a probabilidade de um objeto existir na *bounding box* e a predição da sua possível classe. Essas modificações incrementaram o desempenho na detecção de *bounding boxes* e objetos pequenos e também o desempenho de forma geral, sem onerar significativamente o tempo de treinamento.

A Yolov4, tem como objetivo principal melhorar a eficiência do algoritmo de detecção de objetos e otimizar a computação paralela. Para tal, foram implementadas melhorias objetivando um aumento de eficiência sem onerar o custo computacional para inferência e treino, como um aumento artificial de dados através de *data augmentation* e *Self-Adversarial Training (SAT)* e aplicação de algoritmos genéticos para uma otimização nos hiperparâmetros. Além disso, utilizou-se a CSPDarkNet53, em substituição da DarkNet-53, para a extração de características. Essas mudanças possibilitaram resultados significativamente melhores do que a Yolov3 e outros algoritmos de detecção de objetos, quando comparados no *dataset* MS COCO e considerando um contexto de detecção de objetos em tempo real, conforme pode ser exemplificado na figura 2.6.

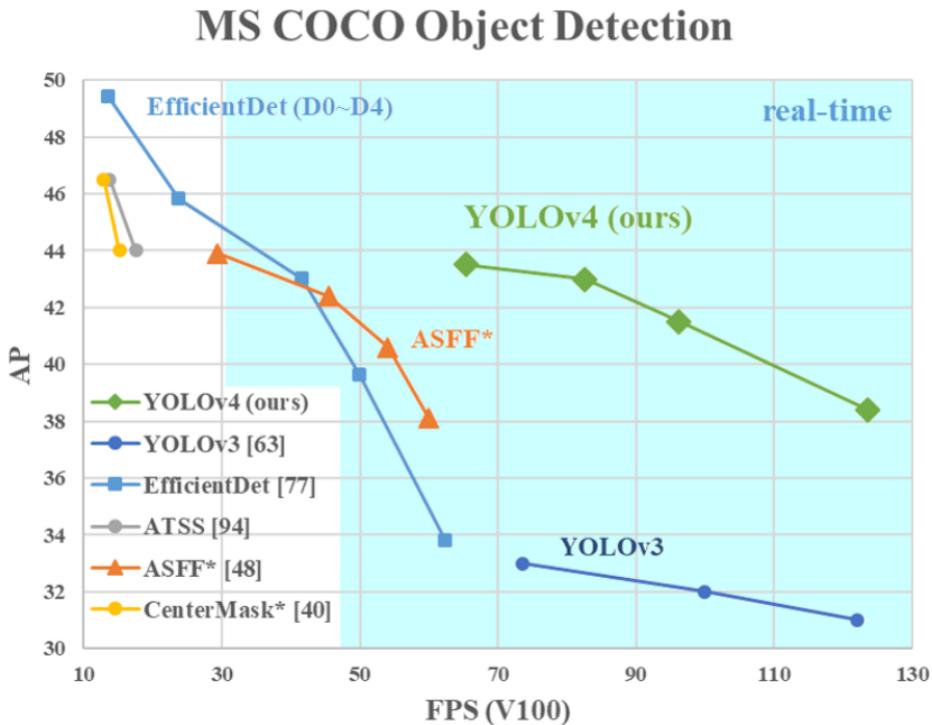


Figura 2.5: Comparação da Yolov4 com outras algoritmos de detecção de objetos no *dataset* MS COCO

Fonte: (BOCHKOVSKIY; WANG; LIAO, 2020)

A Yolov5, visa dar prosseguimento às melhorias implementadas na Yolov4, focando principalmente em facilitar a implementação do algoritmo de detecção de objetos utilizando dados customizados, redução de tempo de treinamento e número de parâmetros sem perda de acurácia (JOCHER et al., 2020). Segundo seus proponentes, o principal objetivo do desenvolvimento da Yolov5 é fazer com que tecnologias de detecção de objetos sejam acessíveis e utilizáveis em uma grande variedade de contextos. Para aumentar a eficiência na detecção de *bounding boxes* em situações diversas, a Yolov5 aprende, através de *K-Means* e *Genetic Learning Algorithms*, as possíveis *bounding boxes* levando em consideração a distribuição dos dados no *dataset*. Quando utilizada no *dataset* MS COCO, as seguintes métricas foram obtidas:

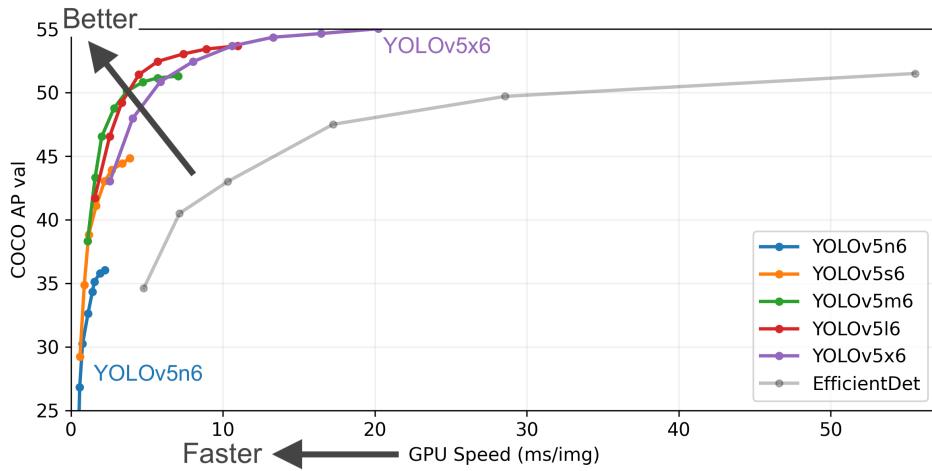


Figura 2.6: Desempenho Yolov5 no *dataset* MS COCO

Fonte: (JOCHER et al., 2020)

A Yolov7, foca em não apenas em otimizar o processo de treinamento juntamente com a arquitetura sem aumentar o custo de inferência, como também responder novos questionamentos levantados pela reparametrização e atribuição dinâmica de *labels* implementada na Yolov6 (WANG; BOCHKOVSKIY; LIAO, 2022a).

Segundo seus propositores, a Yolov7 ultrapassa todos os algoritmos de detecção de objetos tanto em acurácia quanto velocidade, entre 5 a 160 FPS (*Frames Per Second*), as alterações implementadas são melhores do que a antecessora, Yolov6, quando comparadas no *dataset* MS COCO, conforme mostrado na figura 2.7:

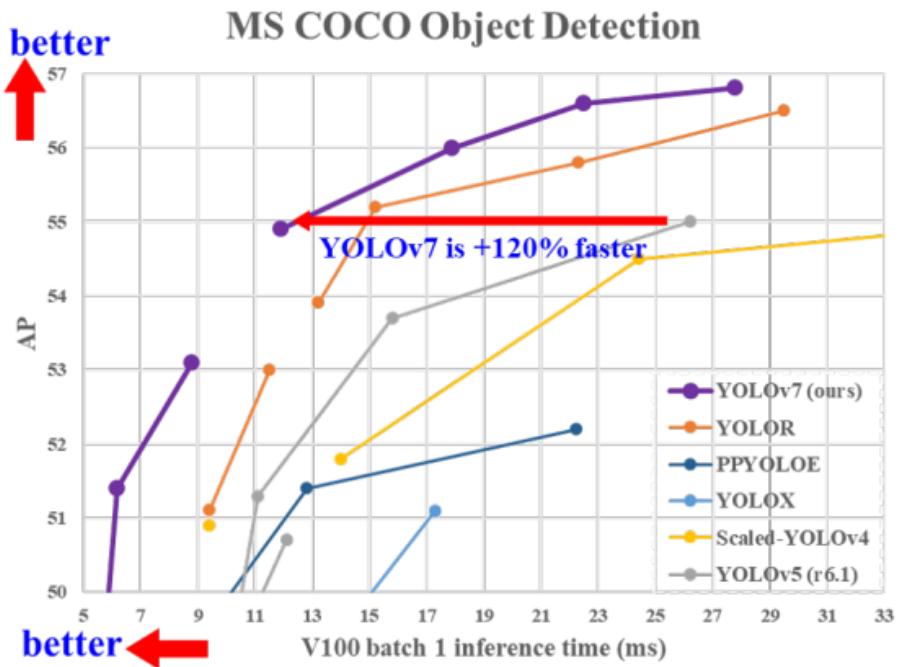


Figura 2.7: Comparação da Yolov7 com outras algoritmos de detecção de objetos no *dataset* MS COCO

Fonte: (WANG; BOCHKOVSKIY; LIAO, 2022a)

A Yolov8, considera várias melhorias em relação a anterior, a citar: mudanças na detecção de *bounding boxes* através da utilização da técnica *anchor free*, que detecta diretamente o centro do objeto ao invés de se basear nas ancoras descritas nas *bounding boxes*, mudanças arquiteturais e melhorias referentes a *data augmentation* principalmente na não utilização da técnica *mosaic* nas últimas dez épocas, uma vez que os autores perceberam que essa técnica em específico estava degradando os possíveis resultados. Essas mudanças possibilitaram resultados significativamente melhores do que a Yolov7, quando comparados no *dataset* MS COCO, conforme pode ser exemplificado na Fig. 2.8.

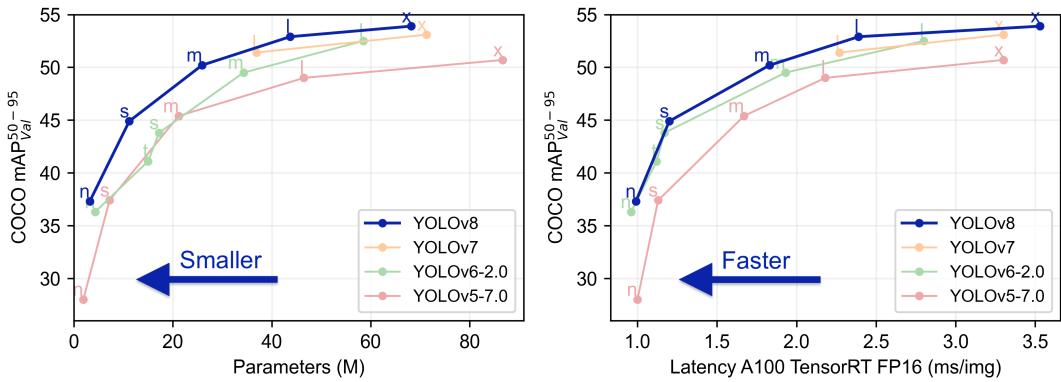


Figura 2.8: Comparação da Yolov8 com outras algoritmos de detecção de objetos no *dataset* MS COCO

Fonte: (JOCHER et al., 2022)

2.3 Métricas de Desempenho em Detecção de Objetos

Com o avanço no desenvolvimento de modelos direcionados a detecção de objetos, foi necessário estabelecer uma série de métricas e parâmetros que são utilizadas para avaliar a qualidade da localização e também classificação nos algoritmos de detecção de objetos. As métricas apresentadas a seguir são referidas em um *survey* disponível na literatura (PADILLA; NETTO; SILVA, 2020).

No contexto de detecção de objetos, é importante que seja estabelecido qual é o conceito de detecção incorreta e correta, para tal utiliza-se a métrica IoU (*Intersection over Union*) que é calculada da seguinte forma e ilustrada de acordo com a figura 2.9:

$$IoU = \frac{area(B_p \cap B_d)}{area(B_p \cup B_d)} \quad (2.1)$$

Onde B_p é a *bounding box* prevista e B_d é a *bounding box* desejada.

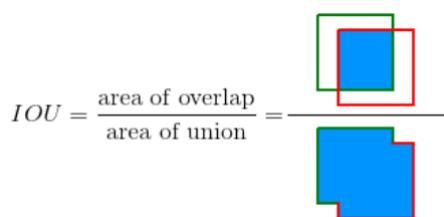


Figura 2.9: Representação gráfica do cálculo de IoU (PADILLA; NETTO; SILVA, 2020)

Ao comparar, o valor calculado com um limite t , é possível saber se a detecção está correta ou incorreta, da seguinte forma: se $\text{IoU} \geq t$ então a detecção está correta, caso $\text{IoU} \leq t$ então a detecção está incorreta. A partir do resultado do IoU é possível três resultados:

- **Verdadeiro Positivo(VP):** Detecção correta de uma *bounding box* desejada.
- **Falso Positivo(FP):** Detecção incorreta de um objeto não existente ou detecção mal posicionada de um objeto existente.
- **Falso Negativo(FN):** Não detecção de uma *bounding box* desejada.

A partir do cálculo dos números de detecções que se encaixam nas categorias acima é possível determinar duas outras métricas importantes na avaliação dos métodos de detecção de objetos, que são: Precisão (P) e Revocação (R) e que podem ser calculadas da seguinte forma:

$$P = \frac{VP}{VP + FP} \quad (2.2)$$

$$R = \frac{VP}{FP + FN} \quad (2.3)$$

Estas métricas são referentes à habilidade do modelo de identificar apenas os objetos considerados relevantes e de encontrar todos estes, respectivamente. Para um modelo para ser considerado bom, é necessário que a Precisão mantenha-se alta conforme a Revocação aumenta, uma forma de visualizar a relação entre essas métricas e extraír as características de um detector é a curva ROC, que descreve a ligação entre a Precisão e Revocação e a troca que ocorre entre elas.

Em casos práticos, a curva ROC apresenta um comportamento em serrilhado, em zig-zag, dificultando o cálculo sob a área da curva ROC, como forma de suavizar este comportamento criou-se uma nova métrica denominada de Average Precision (AP) que pode ser obtida através da média dos valores de uma interpolação de 11 pontos igualmente distribuídos, sendo calculado da seguinte forma:

$$AP_{11} = \frac{1}{11} \sum_{R \in (0, 0.1, \dots, 0.9, 1)} P_{\text{interp}}(R) \quad (2.4)$$

Onde:

$$P_{\text{interp}}(R) = \max P(R) \quad (2.5)$$

Ao invés de usar a Precisão, neste caso utiliza-se a Average Precision, que considera a precisão máxima de todos os valores acima de uma valor de Revocação R, para uma classe em específico.

De forma geral, temos a *Mean Average Precision*, MAP, que é uma métrica utilizada para calcular a acurácia do detector de todas as classes presentes em um dataset específico:

$$MAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (2.6)$$

Onde AP_i , representa o AP da i -ésima classe e N o total de classes. Estas métricas são muito utilizadas para aferir o desempenho de sistemas de detecção de objetos, sendo utilizados em competições relevantes como: *Open Image Dataset*, *Pascal VOC Challenge* e *MS COCO Challenge* (KUZNETSOVA et al., 2020).

2.4 Trabalhos Relacionados

Como mencionado anteriormente, o problema de detecção de semáforos já foi amplamente abordado na literatura perante diferentes estratégias. As primeiras soluções consideravam o *pipeline* clássico de soluções de VC, em que havia a extração de características das imagens, seguida do uso de métodos e modelos tradicionais de *Machine Learning* para classificação. Tais soluções, entretanto, mostravam-se limitadas ao detectarem apenas semáforos de tamanhos e modelos específicos sob certos tipos de *background* pré-determinados, por exemplo (DIAZ et al., 2015). Assim como para outras tarefas, soluções de *Deep Learning* impulsionaram resultados melhores especialmente sob condições menos controladas, colaborando para avanços no estado

da arte (POSSATTI et al., 2019).

Considerando os objetivos desse trabalho, serão consideradas nessa discussão apenas trabalhos da literatura desenvolvidos e voltados para a mesma base de dados aqui considerada, o que propicia uma comparação equânime. Tal base, denominada *Bosch Small Traffic Lights Dataset* (BOSCH) (BEHRENDT; NOVAK; BOTROS, 2017), foi elencada para o escopo deste trabalho por trabalho por ser pública, gratuita, diversa, realística e suas imagens possuírem boa resolução. Uma particularidade dessa base de dados é que a mesma possui mais exemplos de teste que de treinamento, o que visa propiciar um ambiente mais desafiador de preparação dos modelos e aferição de sua capacidade de generalização.

A primeira solução disponível na literatura sobre a base de dados em questão é de autoria dos seus próprios proponentes, os quais utilizaram um modelo YOLO como extrator de características para uma CNN, cujas camadas de classificação e saída foram especificadas pelos próprios autores. Por meio de experimentos, os autores reportaram uma acurácia de 99 % e 95,1 % nas partições de treinamento e teste, respectivamente, mas afirmaram a necessidade de melhorias, pois a métrica IoU mostrou-se particularmente baixa, com valores aferidos entre 0,3 e 0,5 (BEHRENDT; NOVAK; BOTROS, 2017).

Dois trabalhos na literatura foram identificados utilizando, além da base de dados BOSCH, a base de dados de apoio LISA *Traffic Lights Dataset* (PHILIPSEN et al., 2015). O primeiro trabalho utilizou três abordagens distintas para detecção e reconhecimento de semáforos, dentre as quais se sobressaiu uma R-CNN com mAP de 56,31 % no conjunto de dados BOSCH e 76,37 % no LISA (ENNAHHAL; BERRADA; FARDOUSSE, 2019). O segundo trabalho consistiu na proposição de uma Rede Neural De-Convolucional com perda de regressão focal, composta por três sub-redes: uma codificadora, uma decodificadora e uma detetora. A rede decodificadora é responsável por gerar um mapa de características detalhado, porém com poucas informações contextuais, e outro pouco detalhado, mas com muitas informações contextuais, para logo em seguida ambos serem repassados à camada de codificação na qual serão combinados e então submetidos à camada de detecção. O modelo proposto conseguiu um mAP de 7,19 % ~ 42,03 % e de 19,86 % ~ 49,16 % nas bases de dados BOSCH e LISA, respectivamente (LEE; KIM, 2019).

É interessante ressaltar, entretanto, que em face de mais exemplos relevantes para o problema, tem-se o favorecimento do aprendizado, proporcionando um aumento de desempenho geral.

A base de dados mencionada foi também utilizada na terceira etapa de uma competição ocorrida em 2021 intitulada *Data Science Challenge*, promovida pelo Instituto Tecnológico de Aeronáutica. A equipe LSI *Data Team* fez uso do modelo YOLOv5 e obteve um mAP de 0,56, as quais culminaram na classificação em segundo lugar nacional (YONEKURA; OLIVEIRA; MENDES, 2021). Uma das principais motivações deste trabalho foi dar continuidade a este desenvolvimento e viabilizar experimentos para aferir se havia melhoria de desempenho perante novas versões da YOLO.

Capítulo 3

Materiais e Métodos

O trabalho em questão aborda o problema de Visão Computacional considerado como uma tarefa de detecção de objetos multi-classe perante o paradigma de Aprendizado Supervisionado utilizando R-CNNs da Família YOLO como modelos de detecção. Os dados experimentais, modelos e avaliação de desempenho são descritos nas subseções a seguir.

3.1 Dados Experimentais

O conjunto de dados utilizados no experimento é proveniente do *Bosch Small Traffic Lights Dataset* (BOSCH) (BEHRENDT; NOVAK; BOTROS, 2017), que possui originalmente 13.425 imagens captadas em duas localizações diferentes, *Francisco Bay Area* e *Palo Alto*, na Califórnia. Neste *dataset* há cerca de 24.000 anotações que incluem as *bounding boxes* e os estados de cada semáforo. A base de dados é disponibilizada de forma particionada entre treino e teste, o quantitativo de cada partição e a sua respectiva distribuição está descrita na Tabela 3.1.

Tabela 3.1: Distribuição dos dados no BOSCH *dataset*.

	Quantidade de imagens	% imagens	Anotações	% Anotações
Treino	5.093	38%	10.756	44%
Teste	8.334	62%	13.486	56%

As classes presentes em cada partição giram em torno do padrão conhecido para semáfo-

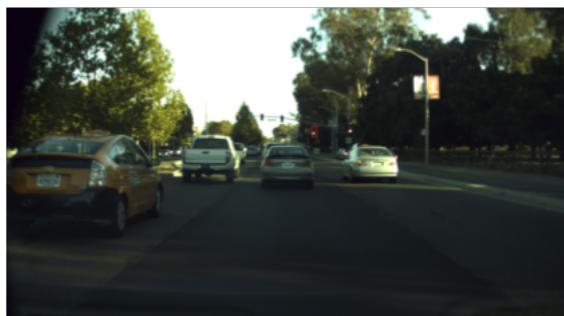
ros urbanos: *Red* (Vermelho), *Yellow* (Amarelo) e *Green* (Verde), mas devido à frequência de captura da câmera aliada à taxa de atualização do semáforo, muitos sinais parecem estar desligados, nesse caso optou-se pela criação da classe *off*. Além disso, em determinadas exemplos, havia rótulos mais específicos que também incluíam direções, tais como *RedLeft* (Vermelho à Esquerda) e *GreenRight* (Verde à Direita). Exemplos ilustrativos da base de dados são apresentados nas Figs. 3.1-3.3.



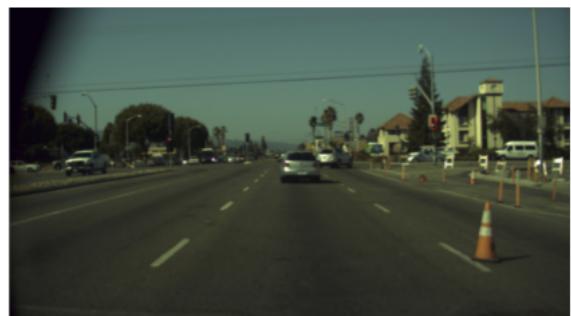
(a) Green



(b) GreenLeft

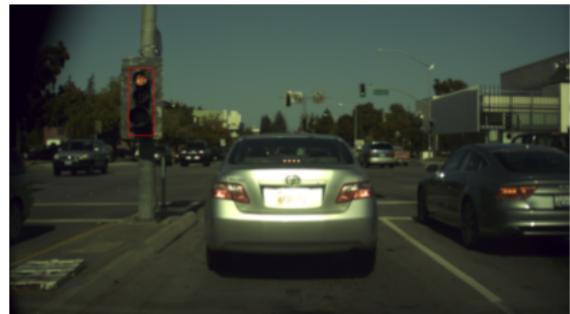


(c) GreenStraightLeft



(d) GreenStraightRight

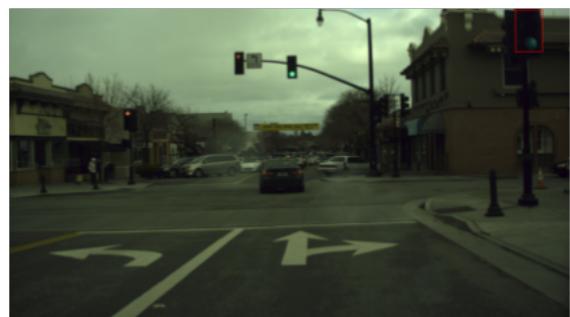
Figura 3.1: Exemplos de imagens do *dataset* BOSCH Traffic Lights ilustrando as classes: Green, GreenLeft, GreenStraightLeft, GreenStraightRight. Fonte: (BEHRENDT; NOVAK,)



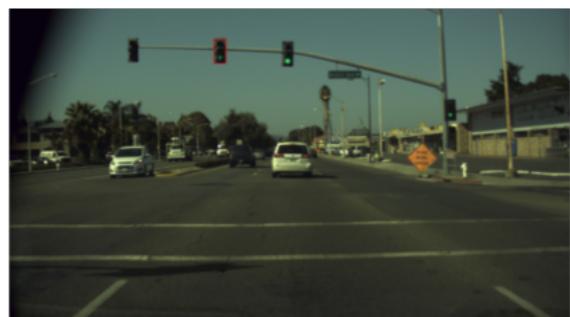
(e) Red



(f) GreenRight

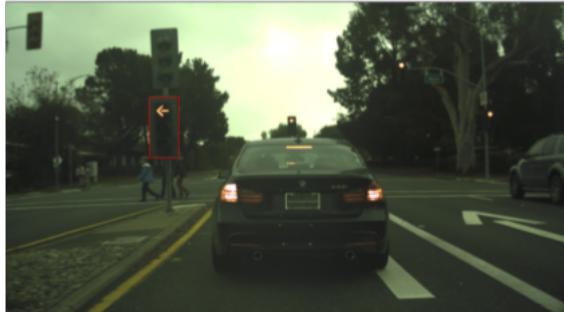


(g) off



(h) GreenStraight

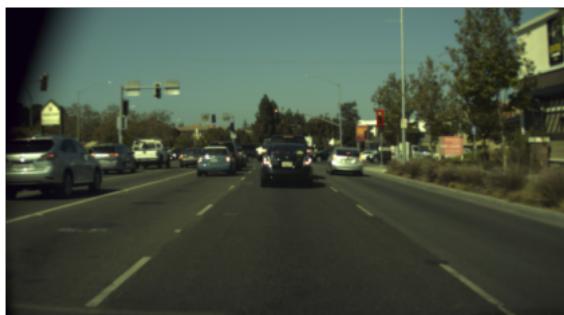
Figura 3.2: Exemplos de imagens do *dataset* BOSCH Traffic Lights ilustrando as classes: Red, GreenRight, off,GreenStraight. Fonte: (BEHRENDT; NOVAK,)



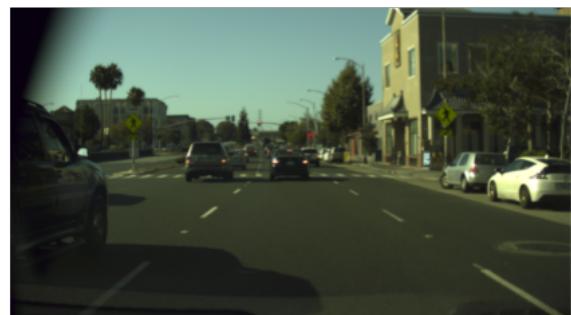
(I) RedLeft



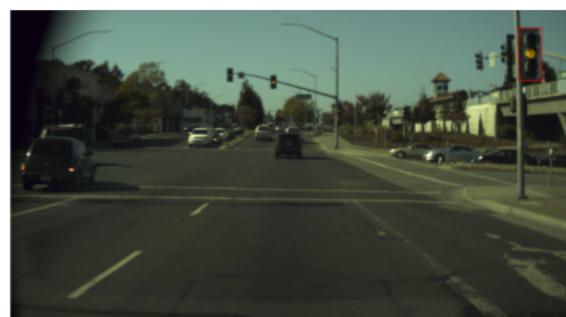
(J) RedRight



(K) RedStraight



(L) RedStraightLeft



(M) Yellow

Figura 3.3: Exemplos de imagens do *dataset* BOSCH Traffic Lights ilustrando as classes: RedLeft, RedRight, RedStraight, RedStraightLeft, Yellow. Fonte: (BEHRENDT; NOVAK,)

Ao examinar o quantitativo de exemplos de semáforos com direção, percebeu-se que este era extremamente desbalanceado e escasso para algumas classes na partição de treinamento. Para prover experiência relevante ao modelo e propiciar a generalização na partição de testes, a especificação de direção foi eliminada, mantendo-se apenas o rótulo padrão do semáforo no tocante ao seu estado. Esse quantitativo nas partições de treino e teste é detalhado nas Tabelas

3.2 e 3.3.

Tabela 3.2: Distribuição de rótulos na partição de treinamento do BOSCH.

Classe	Quantidade	em %
Green	5421	50,4
Red	4164	38,7
Yellow	444	4,1
off	726	6,8

Tabela 3.3: Distribuição de rótulos na partição de testes do BOSCH.

Classe	Quantidade	em %
Green	7569	56,12
Red	5321	38,78
Yellow	154	1,14
off	442	3,27

De modo geral, as *bounding boxes* extraídas dos metadados fornecidos no *dataset*, possuem as características descritas na Tabela 3.4:

Tabela 3.4: Tamanhos das *bounding boxes* do conjunto de teste em pixels.

	Mínima	Média	Mediana	Máxima
Largura	1,88	9,43	8,50	4,84
Altura	6,25	26,77	24,50	104,50
Área	11,72	313,58	212,44	4.734,00

Tabela 3.5: Tamanhos das *bounding boxes* do conjunto de treino em pixels.

	Mínima	Média	Mediana	Máxima
Largura	1,75	11,18	8,55	98,00
Altura	2,62	24,33	18,93	207,00
Área	4,96	404,55	158,80	20.286,00

O conjunto de dados disponibilizados, apesar de já estar previamente particionado em treino e teste, possuía um padrão próprio de anotações, conforme ilustrado na Figura 3.4. Essa rotulação não segue padrões mais amplamente adotados para detecção de objetos com *Deep Learning*,

tais como o PASCAL VOC (EVERINGHAM et al., 2010) ou as rotulações normalizadas adotadas pelos modelos da Família YOLO. Assim, foi necessário o desenvolvimento de *scripts* na linguagem Python para converter os rótulos numéricos para o padrão da Família YOLO, conforme resultado ilustrado na Figura 3.5.

```

1 - boxes:
2   - {label: Green, occluded: false, x_max: 752.25, x_min: 749.0, y_max: 355.125,
  y_min: 345.125}
3   path: 24068.png

```

Figura 3.4: Anotação do *Bosch Small Traffic Lights Dataset*.

```

1 3 0.17500000000000002 0.3604166666666666 0.0765625 0.28750000000000003
2
3 3 0.5134765625000001 0.3956597222222223 0.0166015625 0.063541666666666666
4
5 4 0.7359365 0.1943576388888889 0.021875000000000002 0.0842013888888889
6
7 4 0.915917968750001 0.3729166666666667 0.021484375 0.079166666666666666

```

Figura 3.5: Exemplo de anotação em padrão YOLO.

Após a preparação dos rótulos, os dados pertencentes à partição de treino foram distribuídos de maneira aleatória e disjunta em duas partições de treino e validação, com 80 % e 20 % dos exemplos, respectivamente. A descrição deste particionamento pode ser visualizada na Tabela 3.6.

Tabela 3.6: Distribuição dos dados nas partições de treino e validação no *dataset* BOSCHv1.

	Quantidade de imagens	Anotações
Treino	3566	7611
Validação	1527	3144

Ao considerar um motorista humano, os semáforos que ocupam pouco espaço no campo visual podem também ser difíceis de classificar, necessitando de uma aproximação para tomada de decisão quanto ao seu estado. No *dataset* original verificou-se muitos semáforos pequenos conforme critério do MS COCO (a partir de 5% da área total), conforme exibido na Tabela 3.7, o que levantou o questionamento se estes foram rotulados em tempo real ou *a posteriori*.

Tabela 3.7: BOSCHv1 – Detalhamento dos semáforos quanto ao tamanho na partição de treinamento.

Classes	Pequeno	Médio	Grande	Total
Green	4734	681	6	5448
Red	3861	302	1	4164
Yellow	410	34	0	444
Off	659	67	0	726
Total	9664	1084	7	10782

Dada a suposição previamente mencionada e também a dificuldade dos modelos de detecção aprenderem padrões de objetos arbitrariamente pequenos distinguindo-os de ruído ou do *background*, avaliou-se a hipótese de treinar os modelos apenas com semáforos médios e grandes, o que diminui os custos computacionais, no intuito de aferir se tal estratégia de aprendizado é eficaz e eficiente para generalização posterior perante objetos pequenos. Como consequência, foi produzida uma versão modificada da partição de treino do BOSCH, a qual será BOSCHv2, apenas com objetos médios e grandes, com descrição estatística conforme Tabela 3.8 e distribuição qualitativa conforme as Tabelas 3.10 e 3.9. Enfatiza-se que não houve alterações na partição de testes.

Tabela 3.8: BOSCHv2 – Tamanhos dos rótulos (em px) da partição de treinamento.

	Mínima	Média	Mediana	Máxima
Largura	16,76	29,39	26,84	98,0
Altura	23,75	63,10	58,23	207,00
Área	1.024,42	1.993,72	1.543,58	20.286,00

Tabela 3.9: BOSCHv2 – Tamanhos dos rótulos (em px) da partição de validação.

Classe	Quantidade	em %
Green	220	64,52
Red	87	25,51
off	25	7,33
Yellow	9	2,64

Tabela 3.10: BOSCHv2 – Detalhamento dos semáforos quanto ao tamanho na partição de treinamento.

Classe	Quantidade	em %
Green	687	63,0
Red	303	27,8
off	67	6,1
Yellow	34	3,1

3.2 Modelos, Parâmetros e Hiperparâmetros

No escopo deste trabalho serão consideradas diferentes versões da arquitetura YOLO, a citar: versões 5(JOCHER et al., 2020), 7(WANG; BOCHKOVSKIY; LIAO, 2022b) e 8(JOCHER et al., 2022), cada uma com um número diferente de parâmetros. Em todas as versões utilizadas, foi aplicada a técnica de *Early Stopping* que verifica se houve melhoria na generalização no conjunto de validação; persistência em disco do melhor conjunto de pesos obtidos durante o treinamento perante o conjunto de validação. Em relação aos hiperparâmetros, os valores padrões foram utilizados exceto em relação ao tamanho do batch e o número de épocas que variaram de acordo com cada modelo.

Para a YOLOv5 serão avaliadas as versões *Nano*, *Small* e *Medium* com os seguintes parâmetros, hiperparâmetros e características:

Tabela 3.11: Descrição dos parâmetros das arquiteturas Yolov5's utilizadas.

Modelo	Parâmetros	Camadas	Tamanho do Batch	Épocas
Yolov5 Nano	3.107.836	281	16	500
Yolov5 Small	7.041.205	214	16	500
Yolov5 Medium	20.899.605	291	16	500

Para a YOLOv7 serão avaliadas as versões *YOLOv7*, *YOLOv7-X* e *YOLOv7-W6* com os seguintes parâmetros, hiperparâmetros e características:

Tabela 3.12: Descrição dos parâmetros das arquiteturas YOLOv7's utilizadas.

Modelo	Parâmetros	Camadas	Tamanho do Batch	Épocas
Yolov7	36.900.000	105	24	600
Yolov7-X	71.300.000	121	16	600
Yolov7-W6	70.400.000	122	16	600

Devido a complicações causadas pela limitação do *hardware* utilizado, tornou-se necessário a utilização de *batches* diferentes dos que foram utilizados anteriormente, com o intuito de incrementar a velocidade de treinamento.

Para a YOLOv8 serão avaliadas as versões *Nano*, *Small* e *Medium* com os seguintes parâmetros, hiperparâmetros e características:

Tabela 3.13: Descrição dos parâmetros das arquiteturas YOLOv8's utilizadas.

Modelo	Parâmetros	Camadas	Tamanho do Batch	Épocas
Yolov8 Nano	3.011.043	225	16	600
Yolov8 Small	11.135.987	225	16	600
Yolov8 Medium	25.856.899	295	16	600

3.3 Avaliação de Desempenho

A avaliação de desempenho dos modelos foi efetuada segundo a técnica de validação cruzada *holdout* do conjunto de dados, onde: o treinamento, para cada arquitetura, foi realizado nas partições de treino e validação e posteriormente aferido a partir das previsões efetuadas na partição de testes. As métricas de desempenho foram aferidas na partição de testes e contemplam: Precisão, Revocação, F_1 -Score e mAP com *threshold* $t \geq 0.5$ para o IoU, a qual será denotada como mAP@0.5. Uma explicação detalhada do cálculo e semântica de tais métricas no contexto da detecção de objetos em Visão Computacional encontra-se disponível na Seção 2.3.

3.4 Tecnologias Utilizadas

As tecnologias utilizadas para o desenvolvimento deste trabalho foram a linguagem de programação Python e bibliotecas disponíveis nesta linguagem, sendo escolhida principalmente pela afinidade com os modelos YOLO e por sua grande quantidade de bibliotecas utilizadas durante o decorrer do projeto.

Para a estruturação dos diretórios e manipulação de arquivos utilizou-se a biblioteca OS (OS, 2023). Quanto a conversão das anotações foram utilizadas as bibliotecas yaml, para leitura do arquivo YAML (PYPI, 2021) disponibilizado no *Bosch Small Traffic Lights Dataset*, e sys (SYS, 2023) para a criação e edição dos arquivos equivalentes a cada *bounding box*.

Para a análise de dados foram utilizadas as bibliotecas pandas (PANDAS, 2023), PIL (Pillow) (PILLOW, 2023) e Jupyter Notebook (JUPYTER, 2023) e para o treinamento e teste dos modelos da família YOLO utilizou-se as bibliotecas PyTorch (PYTORCH, 2023) e Ultralytics (ULTRALYTICS, 2023).

Por fim para a execução dos *scripts* de treinamento das redes YOLO foram utilizados servidores disponibilizados pelo Laboratório de Sistemas Inteligentes. A infraestrutura computacional utilizada consistiu em dois servidores com Intel(R) Core(TM) i7-8700 CPU @ 3,20 GHz, 32 GB de memória principal, 2,4 TB de memória secundária e 2 GPUs NVIDIA GeForce GTX 1080 Ti com 11 GB de VRAM para aceleração em *hardware* do treinamento.

Capítulo 4

Resultados

Uma vez estabelecida a metodologia experimental, partiu-se para a execução do treinamento e teste dos modelos, este último perante a partição original da base de dados BOSCH. Os resultados obtidos pelos diferentes modelos e versões de arquiteturas da Família YOLO encontram-se detalhados na Tabela 4.1.

Tabela 4.1: Síntese dos resultados experimentais.

Dataset	Modelo	Tempo de Treinamento	Precisão	Revocação	F-Score	mAP@0.5
BoschV1	YOLOv5 Nano	4h:40min	0.717	0.562	0.630	0.613
	YOLOv5 Small	4h:14min	0.735	0.596	0.658	0.627
	YOLOv5 Medium	3h:44min	0.719	0.611	0.660	0.642
	YOLOv7	10h:55min	0.454	0.248	0.321	0.255
	YOLOv7-X	19h:44min	0.656	0.467	0.546	0.481
	YOLOV7-W6	18h:12min	0.472	0.385	0.424	0.374
	YOLOv8 Nano	4h:24min	0.702	0.476	0.567	0.523
	YOLOv8 Small	5h:11min	0.682	0.480	0.563	0.549
BoschV2	YOLOv8 Medium	6h:35min	0.705	0.485	0.575	0.556
	YOLOv5 Nano	2h:50min	0.697	0.599	0.644	0.635
	YOLOv5 Small	4h:9min	0.763	0.566	0.649	0.644
	YOLOv5 Medium	4h:9min	0.761	0.581	0.658	0.653
	YOLOv7	10h:10min	0.866	0.527	0.655	0.649
	YOLOv7-X	17h:58min	0.740	0.562	0.639	0.648
	YOLOV7-W6	15h:15min	0.800	0.606	0.689	0.657
	YOLOv8 Nano	2h:20min	0.707	0.824	0.761	0.826
	YOLOv8 Small	2h:55min	0.769	0.779	0.774	0.851
	YOLOv8 Medium	4h:34min	0.860	0.708	0.777	0.821

Ao analisar as métricas de desempenho obtidas, concluiu-se que as variantes da arquitetura YOLOv8, especificamente as versões *Nano* e *Small*, obtiveram um bom desempenho, com um

mAP@0.5 de 0,826 e 0,851, respectivamente. Nas matrizes de confusão ilustradas nas Figuras 4.1 e 4.2 percebe-se a prevalência de dados na diagonal principal, atestando uma boa eficiência dos modelos referidos no tocante à tarefa considerada.

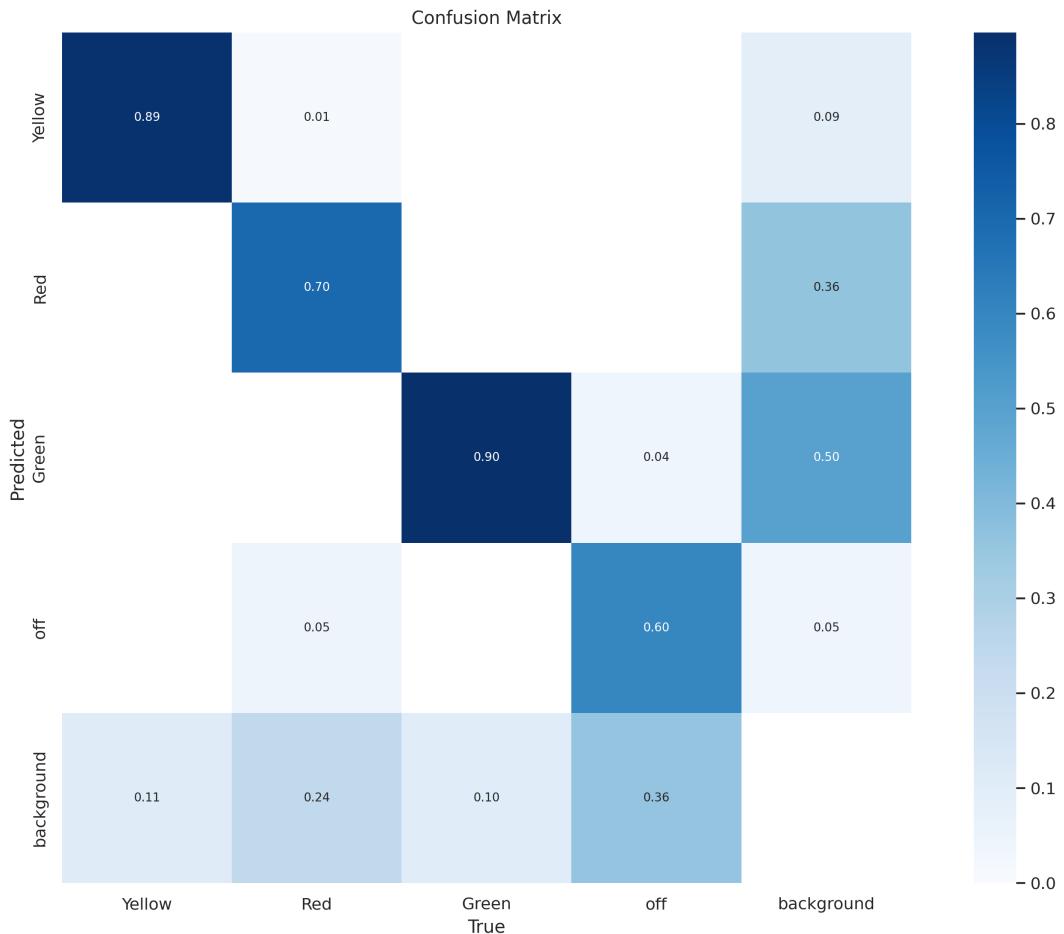


Figura 4.1: YOLOv8 *Small* – BOSCHv2 – Matriz de confusão



Figura 4.2: YOLOv8 *Medium* – BOSCHv2 – Matriz de confusão

Apesar do *dataset* de treinamento BOSCHv2 possuir uma quantidade reduzida de exemplos, todos os modelos treinados sobre o mesmo demonstraram bom desempenho na etapa posterior de testes. Este achado demonstra que a diminuição do ruído, por meio da redução das *bounding boxes* pequenas, foi benéfica para a melhora não apenas das métricas obtidas, como também para a redução do tempo de treinamento, fazendo com que os modelos YOLOv8 *Small* e *Medium* atingissem o *Early Stopping* nas épocas 222 e 270, com um valor padrão de paciência igual a 50, e tempo de treinamento entre 2,5 e 3 horas, respectivamente. Os resultados obtidos neste cenário em específico comprovam que a Família YOLO no geral, possui boas técnicas de generalização e *data augmentation* possibilitando um boa diversidade de dados para o treinamento. Alguns exemplos ilustrativos de detecção são mostrados nas Figuras 4.3-4.6.



Figura 4.3: YOLOv8 *Nano* – BOSCHv2.



Figura 4.4: YOLOv8 *Nano* – BOSCHv2.



Figura 4.5: YOLOv8 *Small* – BOSCHv2.



a) Desejado

b) Previsto

Figura 4.6: YOLOv8 Small – BOSCHv2.

É importante notar que os modelos mais antigos mostraram-se mais flexíveis em relação aos dados de treinamento, como aqueles da YOLOv5, os quais demonstraram um desempenho superior notável quando comparado aos outros modelos no mesmo *dataset* BOSCHv1, evidenciando que melhorias implementadas com foco em *datasets* com imagens com características variadas diferentes das presentes no *dataset* COCO (LIN et al., 2015) também incrementaram as métricas obtidas.

Capítulo 5

Considerações Finais

O objetivo desta proposta de trabalho consiste em avaliar o uso de CNNs da família YOLO para a tarefa de detecção de semáforos e classificação a partir da base de dados *Bosch Small Traffic Light Dataset* (BEHRENDT; NOVAK; BOTROS, 2017) que possui mais de 13.000 imagens captadas. Durante a primeira etapa, foi utilizado a arquitetura YOLOv5 (JOCHER et al., 2020), especificamente suas variantes *Nano*, *Small* e *Medium* e considerou-se duas bases de dados, a *Bosch Small Traffic Light* e uma derivação, denominada BOSCHv2. Os melhores resultados foram obtidos pelas arquiteturas YOLOv5 *Small* e YOLOv5 *Medium*, com um mAP@0.5 de 0,627 e 0,642, respectivamente.

Dando sequência aos experimentos realizados inicialmente, outras versões da família YOLOv7 e YOLOv8 foram implementadas e avaliadas e verificou-se que o desempenho dos modelos *YOLOv8 Nano* e *YOLOv8 Small* foi superior aos demais, com um mAP@0.5 de 0,826 e 0,851, respectivamente, mesmo com uma quantidade de exemplos limitada no *dataset* BOSCHv2, evidenciando a capacidade de generalização da família YOLOv8. Desta forma, apresenta-se como uma solução competitiva para o problema de detecção semafórica.

Em trabalhos futuros, pretende-se ampliar o número de modelos avaliados da família YOLO e realizar novas experimentações. Além disso, considera-se a captação de novos exemplos para a ampliação do *dataset* consolidando um conjunto de dados mais robusto e abrangente e rebalancear a quantidade de imagens obtidas de forma mais eficiente.

Referências Bibliográficas

- BEHRENDT, K.; NOVAK, L. A deep learning approach to traffic lights: Detection, tracking, and classification. In: *IEEE. Robotics and Automation (ICRA), 2017 IEEE International Conference on*. [S.l.].
- BEHRENDT, K.; NOVAK, L.; BOTROS, R. A deep learning approach to traffic lights: Detection, tracking, and classification. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore: IEEE, 2017.
- BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv, 2020. Disponível em: <<https://arxiv.org/abs/2004.10934>>.
- CHOLLET, F. *Deep Learning with Python*. [S.l.]: Manning, 2017. ISBN 9781617294433.
- DENG, J. et al. ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Miami, FL: IEEE, 2009.
- DIAZ, M. et al. A survey on traffic light detection. In: *New Trends in Image Analysis and Processing – ICIAP 2015 Workshops*. Itália: Springer International Publishing, 2015, (Lecture notes in Computer Science). p. 201–208.
- ENNAHHAL, Z.; BERRADA, I.; FARDOUSSE, K. Real time traffic light detection and classification using deep learning. In: *2019 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. [S.l.: s.n.], 2019. p. 1–7.
- EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. *International journal of computer vision*, Springer, v. 88, n. 2, p. 303–338, 2010.
- GIFFINGER, R. et al. *Smart cities - Ranking of European medium-sized cities*. [S.l.: s.n.], 2007. - p.
- GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1440–1448.
- JENSEN, M. B.; NASROLLAHI, K.; MOESLUND, T. B. Evaluating state-of-the-art object detector on challenging traffic light data. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. [S.l.]: IEEE, 2017.
- JIAO, L. et al. A survey of deep learning-based object detection. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 7, p. 128837–128868, 2019.
- JOCHER, G. et al. *ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation*. Zenodo, 2022. Disponível em: <<https://doi.org/10.5281/zenodo.7347926>>.

- JOCHER, G. et al. *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*. [S.l.]: Zenodo, 2020. Disponível em <<https://doi.org/10.5281/zenodo.4154370>>. Acesso em 2 de agosto de 2023.
- JUPYTER. 2023. Disponível em <<https://jupyter.org/>>. Acesso em 2 de agosto de 2023.
- KHAN, S. et al. *A Guide to Convolutional Neural Networks for Computer Vision*. Springer International Publishing, 2018. Disponível em: <<https://doi.org/10.1007/978-3-031-01821-3>>.
- KIM, H.-K.; PARK, J. H.; JUNG, H.-Y. An efficient color space for deep-learning based traffic light recognition. *Journal of advanced transportation*, Hindawi, v. 2018, 2018.
- KIRWAN, C.; ZHIYONG, F. *Smart cities and artificial intelligence*. Philadelphia, PA: Elsevier Science Publishing, 2020. (Smart Cities).
- KON, F.; SANTANA, E. F. Z. Cidades inteligentes: Conceitos, plataformas e desafios. In: JOSÉ CARLOS MALDONADO AND JOSÉ VITERBO AND MARCIO DELAMARO AND SABRINA DOS SANTOS MARCZAK. *Jornadas de atualização em Informática*. Porto Alegre: SBC, 2016. v. 17. Disponível em <<https://doi.org/10.5753/sbc.6.1>>.
- KUZNETSOVA, A. et al. The open images dataset v4. *International Journal of Computer Vision*, Springer, v. 128, n. 7, p. 1956–1981, 2020.
- LEE, E.; KIM, D. Accurate traffic light detection using deep neural network with focal regression loss. *Image and Vision Computing*, v. 87, p. 24–36, 2019. ISSN 0262-8856. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0262885619300538>>.
- LIN, T.-Y. et al. *Microsoft COCO: Common Objects in Context*. 2015.
- MICHELUCCI, U. *Advanced applied Deep Learning: Convolutional neural networks and object detection*. [S.l.]: Appress, 2019.
- OS. 2023. Disponível em <<https://docs.python.org/3/library/os.html>>. Acesso em 2 de agosto de 2023.
- PADILLA, R.; NETTO, S. L.; SILVA, E. A. B. da. A Survey on Performance Metrics for Object-Detection Algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. Niterói, Brasil: [s.n.], 2020. p. 237–242. Disponível em <<http://dx.doi.org/10.1109/IWSSIP48289.2020.9145130>>. Acesso em 2 de agosto de 2023.
- PANDAS. 2023. Disponível em <<https://pandas.pydata.org/>>. Acesso em 2 de agosto de 2023.
- PHILIPSEN, M. P. et al. Traffic light detection: A learning algorithm and evaluations on challenging dataset. In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. [S.l.: s.n.], 2015. p. 2341–2345.
- PILLOW. 2023. Disponível em <<https://pillow.readthedocs.io/en/stable/>>. Acesso em 2 de agosto de 2023.

- POSSATTI, L. C. et al. Traffic light recognition using deep learning and prior maps for autonomous cars. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. Budapest, Hungary: IEEE, 2019.
- PYPI. 2021. Disponível em <<https://pypi.org/project/PyYAML>>. Acesso em 2 de agosto de 2023.
- PYTORCH. 2023. Disponível em <<https://pytorch.org/>>. Acesso em 2 de agosto de 2023.
- REDMON, J. et al. You only look once: Unified, real-time object detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, Estados Unidos: IEEE, 2016.
- REDMON, J.; FARHADI, A. Yolo9000: better, faster, stronger. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Estados Unidos: [s.n.], 2017. p. 7263–7271.
- REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- SONG, H. et al. *Smart cities*. [S.l.]: Standards Information Network, 2017.
- SYS. 2023. Disponível em <<https://docs.python.org/3/library/sys.html>>. Acesso em 2 de agosto de 2023.
- ULTRALYTICS. 2023. Disponível em <<https://ultralytics.com/>>. Acesso em 2 de agosto de 2023.
- WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. arXiv, 2022. Disponível em: <<https://arxiv.org/abs/2207.02696>>.
- WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. [S.l.]: arXiv, 2022. Disponível em <<https://doi.org/10.48550/arxiv.2207.02696>>. Acesso em 2 de agosto de 2023.
- YONEKURA, D. C.; OLIVEIRA, M. V. de; MENDES, L. M. *LSI Data Team no Data Science Challenge at ITA Edição 2021*. São José dos Campos, SP: ITA, 2021. Disponível em <<https://www.youtube.com/watch?v=RmodnKr65aI>>. Acesso em 2 de agosto de 2023.