UNIVERSIDADE FEDERAL DO PAMPA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

JEAN LUCAS DA SILVA CIMIRRO

RECONHECIMENTO DE IMAGENS: USO DO MÉTODO YOLO NO RECONHECIMENTO DE PLACAS DE TRÂNSITO

Bagé
2022

JEAN LUCAS DA SILVA CIMIRRO

RECONHECIMENTO DE IMAGENS: USO DO MÉTODO YOLO NO RECONHECIMENTO DE PLACAS DE TRÂNSITO

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Gerson Alberto Leiria Nunes

Bagé
2022

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

C573r Cimirro, Jean Lucas da Silva
Reconhecimento de imagens: Uso do método Yolo no reconhecimento de placas de trânsito / Jean Lucas da Silva Cimirro.
64 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade Federal do Pampa, ENGENHARIA DE COMPUTAÇÃO, 2022.
"Orientação: Gerson Alberto Leiria Nunes".

1. YOLOv5. 2. Reconhecimento. 3. Placas de trânsito. 4. Redes neurais convolucionais. I. Título.

SERVIÇO PÚBLICO FEDERAL

MINISTÉRIO DA EDUCAÇÃO

Universidade Federal do Pampa

JEAN LUCAS DA SILVA CIMIRRO

RECONHECIMENTO DE IMAGENS: USO DO MÉTODO YOLO NO RECONHECIMENTO DE PLACAS DE TRÂNSITO

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 10 de Agosto de 2022.

Banca examinadora:

_____

Prof. Dr. Gerson Alberto Leiria Nunes
Orientador – Universidade Federal do Pampa

_____

Prof. Dr. Fábio Luís Livi Ramos
Universidade Federal do Pampa

Prof. Dr. Milton Roberto Heinen
Universidade Federal do Pampa

Assinado eletronicamente por **GERSON ALBERTO LEIRIA NUNES**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 17:36, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **FABIO LUIS LIVI RAMOS**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 17:36, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **MILTON ROBERTO HEINEN**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/08/2022, às 22:00, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0903593** e o código CRC **1E71A4DA**.

Referência: Processo nº 23100.017420/2022-16 SEI nº 0903593

RESUMO

O reconhecimento de sinais de trânsito é uma parte essencial dos sistemas de assistência ao motorista, capaz de auxiliar os motoristas a evitar muitos perigos potenciais e melhorar a experiência de dirigir. No entanto, o reconhecimento de sinais de trânsito é uma tarefa cheia de desafios, como ambiente visual, danos físicos e oclusões parciais, entre outros. Para lidar com as restrições, às redes neurais convolucionais (CNN) são acomodadas para extrair o visual características dos sinais de trânsito e classificá-los em classes correspondentes. Nessa pesquisa buscou-se reconhecer sinais de trânsito brasileiros, tendo como principal ferramenta o uso do método You Only Look Once (YOLO), na sua quinta versão. Ele usa de uma única rede neural convolucional que prevê simultaneamente várias caixas delimitadoras e probabilidades de classe para essas caixas, o YOLO treina em imagens completas e otimiza diretamente o desempenho de detecção. O YOLOv5 traz uma melhor acessibilidade para detecção de objetos em tempo real e o desempenho da previsão com base no tempo de resposta ou na precisão do treinamento comparado com sua versão anterior. Foram realizados testes com três arquiteturas do YOLO, *small, large* e *extra large*, obtidas precisões de 92,9%, 92,6% e 92,0%, respectivamente, além da criação de um *dataset* das placas brasileira e uma interface para o usuário.

Palavras-chave: YOLOv5; Reconhecimento; Placas de trânsito; Redes neurais convolucionais.

ABSTRACT

The traffic sign recognition has been an essential part of driver assistance systems, which are able to help drivers avoiding a large number of potential hazards and improving the driving experience. However, recognizing traffic signs is a task that is full of challenges, such as visual environment, physical damage and partial occlusions, among others. To deal with the constraints, convolutional neural networks (CNN) are accommodated to extract the visual characteristics of traffic signs and classify them into corresponding classes. This graduate work, initially intends to recognize traffic signs in Brazil, having as main framework the use of the You Only Look Once (YOLO) method, in its fifth version. Using a single convolutional network that simultaneously predicts multiple bounding boxes and class probabilities for these boxes, YOLO trains on complete images and directly optimizes detection performance. YOLOv5 brings better accessibility for real-time object detection and prediction performance based on training speed or accuracy compared to its previous version. Tests were carried out with three YOLO architectures, small, large and extra large, obtaining accuracies of 92.9%, 92.6% and 92.0%, respectively, in addition to creating a dataset of the Brazilian boards and an user's interface.

Keywords: YOLOv5. Recognition. Traffic signs. Convolutional neural networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

1D	Unidimensional
2D	Bidimensional
3D	Tridimensional
ACM	Association for Computing Machinery
CSUST	*Chinese Traffic Sign Detection Benchmark*
CNN	Rede Neural Convolucional
CNNs	Redes Neurais Convolucionais
CVAT	*Computer Vision Annotation Tool*
FPS	*Frames per Second*
GPU	*Graphical Processing Unit*
IEEE	Institute of Electrical and Electronics Engineers
IA	Inteligência Artificial
mAP	*mean Average Precision*
MIT	*Massachusetts Institute of Technology*
ML	*Machine Learning*
RNA	Rede Neural Artificial
RNAs	Redes Neurais Artificiais
SAT	*Self-Adversarial Training*
SIMD	Single Instruction Multiple Data
TCC	Trabalho de Conclusão de Curso
TT100K	*Tsinghua-Tencent 100K*
UML	*Unified Modeling Language*
VoTT	*Visual Object Togging Tool*)
XML	*eXtensible Markup Language*
YOLO	You Only Look Once

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, a detecção de objetos se tornou um campo significativo de visão computacional. De acordo com Szegedy, Toshev e Erhan (2013), conforme se avança em direção a uma compreensão mais completa da imagem, ter um reconhecimento de objeto mais preciso e detalhado torna-se crucial. Nesse contexto, preocupa-se não apenas em classificar imagens, mas também em estimar com precisão qual sua classe e a localização dos objetos contidos nas imagens, problema conhecido como detecção de objetos. Os principais avanços na detecção de objetos foram alcançados graças à melhorias nas representações de objetos e modelos de aprendizado de máquina.

Para Barelli (2018) os sistemas de Visão Computacional estão cada vez mais presentes em nosso cotidiano, seja nos veículos autônomos, nos robôs industriais ou em equipamentos hospitalares capazes de diagnosticar doenças automaticamente em exames por imagem, permitindo com que máquinas enxerguem o mundo à nossa volta, conseguindo automatizar e solucionar diversos problemas.

De acordo com Houben *et al.* (2013), diversas aplicações de visão computacional do mundo real requerem uma detecção precisa de objetos em imagens e vídeo. Uma destas aplicações é o reconhecimento de sinais de trânsito, em que os algoritmos precisam lidar com ambientes dinâmicos naturais, com isso, se tornando complexos, demandas de alta precisão e restrições de tempo real. Devido possuir uma alta relevância industrial, muitas abordagens para detecção e reconhecimento de sinais de trânsito foram propostas. Sistemas avançados de assistência ao motorista com reconhecimento de sinais de trânsito, geralmente limitados a um subconjunto de sinais possíveis, foram implantados pela indústria automotiva. O processo de reconhecimento de sinais de trânsito se divide em duas principais etapas: detecção do sinal em uma imagem ou fluxo de vídeo e o reconhecimento subsequente (ou seja, classificação) dos sinais detectados.

Segundo Cireşan *et al.* (2011) *Deep Learning* (aprendizado profundo), é um importante método de aprendizado de máquina que, ultimamente, tem sido aplicado com sucesso em diversas tarefas, como reconhecimento de numerais manuscritos, classificação, detecção, rastreamento, processamento de linguagem natural e sistemas inteligentes de perguntas e respostas e alcançou realizações que vão além do alcance de métodos tradicionais. O sucesso do aprendizado profundo se beneficia não apenas de modelos maiores e mais profundos com mais parâmetros.

Redes Neurais Convolucionais (CNN) são arquiteturas que permitem serem

treinadas, CNN é um método de *deep learning*, com inspiração biológica que pode aprender com recursos constantes. De acordo com LeCun, Kavukcuoglu e Farabet (2010), a entrada e a saída de cada estágio são conjuntos de matrizes chamados "mapas de recursos". De maneira semelhante aos processos tradicionais de visão computacional, uma CNN é capaz de aplicar filtros em dados visuais, de forma que a relação de vizinhança entre os *pixels* da imagem ao longo do processamento da rede seja mantida. Este tipo de rede vem sendo amplamente utilizado, principalmente nas aplicações de classificação, detecção e reconhecimento em imagens e vídeos.

Uma rede neural convolucional consiste em múltiplas partes com funções diferentes. Inicialmente é comum aplicar sobre o dado de entrada camadas ditas de convolução. Uma camada de convolução é composta por diversos neurônios, cada um responsável por aplicar um filtro em um pedaço específico da imagem. Atribuindo cada neurônio para um conjunto de *pixels* da camada anterior e que a cada uma dessa conexão se atribui um peso. A combinação das entradas de um neurônio, utilizando os pesos respectivos de cada uma de suas conexões, produz uma saída passada para a camada seguinte. Os pesos atribuídos às conexões de um neurônio podem ser interpretados como uma matriz que representa o filtro de uma convolução de imagens no domínio espacial (conhecido como *kernel* ou máscara) (VARGAS; PAES; VASCONCELOS, 2016).

Os humanos olham para uma imagem e sabem, instantaneamente, quais objetos estão na imagem, onde estão e como interagem. O sistema visual humano é rápido e preciso, permitindo-nos realizar tarefas complexas como dirigir com pouco pensamento consciente. Algoritmos rápidos e precisos para detecção de objetos permitiriam que os computadores dirigissem carros sem sensores especializados, habilitaram dispositivos auxiliares para transmitir informações de cena em tempo real para usuários humanos e desbloquearam o potencial para sistemas robóticos responsivos de uso geral. Os sistemas de detecção atuais adaptam os classificadores para realizar a detecção. Para detectar um objeto, esses sistemas pegam um classificador para aquele objeto e o avaliam em vários locais e escalas em uma imagem de teste. Sistemas como modelos de peças deformáveis usam uma abordagem de janela deslizante em que o classificador é executado em locais com espaçamento uniforme em toda a imagem.

O *You Only Look Once* (YOLO) usa de uma única rede convolucional que prevê simultaneamente várias caixas delimitadoras e probabilidades de classe para essas caixas. O YOLO treina em imagens completas e otimiza diretamente o desempenho de detecção. Este modelo unificado tem vários benefícios sobre os métodos tradicionais de detecção de

objetos. O YOLO raciocina globalmente sobre a imagem ao fazer previsões. Ao contrário de janela deslizante e técnicas baseadas em proposta de região, o YOLO vê a imagem inteira durante o treinamento e, o tempo de teste, portanto, codifica implicitamente as informações contextuais sobre as classes, bem como sua aparência (REDMON *et al.*, 2016).

1.1 Objetivos

Neste sentido, a metodologia de pesquisa refere-se a uma abordagem de pesquisa exploratória e experimental, iniciando por um estudo teórico de pesquisa bibliográfica, norteado pelo seguinte problema de pesquisa: Uma rede neural convolucional seria capaz de identificar e catalogar as placas de trânsito? Tendo como objetivo geral realizar a verificação da possibilidade de redes neurais convolucionais de identificar placas de trânsito em estradas usando como método o YOLOv5. Como objetivos específicos, buscou-se:

- Realizar uma revisão bibliográfica acerca do tema;
- Buscar artigos correlatos;
- Adquirir base de dados de imagens/vídeos;
- Validar as imagens da base de dados;
- Efetuar pré-processamento dos vídeos e imagens;
- Validar as redes neurais obtidas através do treinamento através dos 20% do *dataset* de imagens que não adicionadas no treino;
- Realizar o treinamento da rede neural para identificação das placas de trânsito;
- Testar a ferramenta;
- Analisar os resultados obtidos.

Primeiramente realiza-se um estudo preliminar sobre o tema, objetivando uma aproximação e entendimento do fenômeno, através do procedimento técnico de uma análise bibliográfica. Portanto, este processo foi realizado anteriormente a esta pesquisa com a leitura e produção de artigos e pesquisas científicas, com a descrição dos tipos e modos.

1.2 Importância e motivação da pesquisa

Para Wen *et al.* (2016) os sinais de trânsito são elementos fundamentais e essenciais nos sistemas de transporte rodoviário, pois, além de sua importância para a segurança no trânsito, orientam os motoristas e demais usuários das vias. Hoje em dia, a expansão contínua das cidades cria uma demanda crescente para a instalação de novas placas e manutenção das existentes com base em inventário e fiscalização. Um processo eficiente de inventário de sinais de trânsito é necessário para registrar com precisão os dados dos sinais, como o número de sinais, tipo, condição e localização geográfica de cada sinal, bem como as relações espaciais entre os sinais.

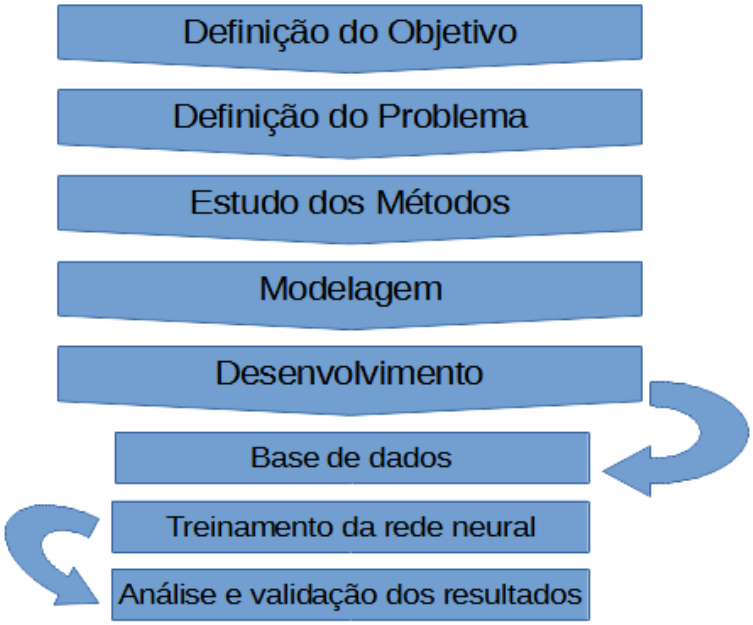
1.3 Organização do texto

Nos capítulos restantes que compõem esta pesquisa, está organizado de forma sequencial da seguinte maneira, no Capítulo 2 apresentando a metodologia utilizada para a pesquisa, no Capítulo 3 é abordada a revisão bibliográfica composta pelas definições dos conceitos utilizados pelos principais autores da área. Na sequência o Capítulo 4, composto pelos trabalhos correlatos sendo, artigos, monografias, dissertações e teses. Ademais, no Capítulo 5 é apresentada a descrição do projeto e suas etapas de desenvolvimento, juntamente da análise de dados, os respectivos resultados e as discussões a respeito dos mesmos. Por fim, no Capítulo 6 são expostas as conclusões e sugestões para trabalhos futuros.

2 METODOLOGIA

Para atingir os objetivos pretendidos, foi utilizada uma abordagem exploratória através de procedimentos experimentais, recorrendo a métodos qualitativos e soluções técnicas práticas (algoritmos de inteligência artificial) de natureza aplicada. A justificativa, métodos, procedimentos e ferramentas utilizados são descritos abaixo. A sequência de etapas planejadas para execução deste projeto será descrita na Figura 1 que representa o conjunto de atividades desenvolvidas.

Figura 1 – Atividades de Metodologia



Fonte: Autor (2021)

Foi tomada então como referência na execução das atividades, Pradonov e Freitas (2013), seguindo a metodologia descrita. Para o primeiro item das atividades, a definição do objetivo, se fez necessário a realização de um estudo preliminar acerca do tema, uma pesquisa exploratória, tendo como objetivo um maior entendimento e aproximação referente ao estudo realizado, utilizando como base a análise bibliográfica. De acordo com Pradonov e Freitas (2013) a pesquisa exploratória procura buscar mais informações a respeito do assunto que será investigado, auxiliando na definição e delimitação do tema da pesquisa, formulando as hipóteses. Em geral, está envolvida com a forma de pesquisa bibliográfica e estudos de caso, proporcionando uma maior familiaridade com o problema.

Para a definição do problema de pesquisa, foi verificado que o método mais adequado seria o método de pesquisa experimental que se dá após a delimitação do

tema, o pesquisador procura refazer as condições de um fato a ser estudado, usando de um ambiente controlado com finalidade de demonstrar como um fato é produzido. É caracterizada pela manipulação das variáveis relacionadas com o objetivo de estudo, buscando as causas e os efeitos de um determinado fenômeno, através da criação de situações de controle, de forma que não haja interferência de variáveis intervenientes. Interferimos diretamente na realidade, manipulando a variável independente, a fim de observar o que acontece com a dependente. (PRADONOV; FREITAS, 2013)

Como terceira atividade, de estudo dos métodos de resolução, foi realizado um estudo, sobre os métodos, ferramentas e algoritmos já existentes para a identificação de placas de trânsito. Suas eficiências e características foram analisadas, para assim adequar e selecionar o mais indicado a esta proposta.

Após a pesquisa experimental ser realizada para obter o embasamento teórico, se deu início ao processo de seleção dos trabalhos correlatos, para essa pesquisa foi feita a busca por documentos científicos que utilizem técnicas de detecção de placas de trânsito utilizando de inteligência artificial. Segundo Neiva e Silva (2016) existem alguns passos para se realizar uma revisão sistemática, sendo eles:

1. A definição das questões de pesquisa, sendo a principal intenção o uso de inteligência artificial para detecção de placas de trânsito.
2. A definição das palavras-chave, são elas, reconhecimento, placas de trânsito, redes neurais convolucionais, inteligência artificial e YOLOv5.
3. A definição de *strings* de busca, que consiste em separar as palavras-chave, concatená-las com seus sinônimos utilizando o conector *OR*, separá-las por grupos de sinônimos, e conectá-las com *AND*, como é mostrado na Tabela 1.
4. Definir a bases de busca, escolhendo o *Google Scholar*[1] por ser uma plataforma que consegue retornar resultado de diversos congressos e eventos.
5. Após um primeiro teste utilizando a *string* de busca na base selecionada, foi necessário realizar um refinamento nela, pelo fato de retornar muitos resultados, 482 publicações, após seu refinamento ficou conforme a Tabela 2.
6. Ao realizar a execução da *string* refinada na base selecionada foi obtido um total de 88 resultados.
7. Baixar e armazenar o resultado das buscas
8. Para os critérios de inclusão e exclusão, foram filtradas publicações a partir de 2016,

[1] https://scholar.google.com.br/

devido ser o ano em que a ferramenta YOLO foi apresentada.

9. Para a seleção dos artigos, foram selecionados artigos nos idiomas português e inglês, pois a busca retornou alguns em Mandarim, se tornando inviável sua leitura para o autor.
10. Após foi analisado o título e *abstract*, diversas publicações foram descartadas devido não possuírem relevância com o tema ficando com 28 dos 88.
11. Uma segunda etapa de seleção foi feita, sendo realizada a leitura de introdução e conclusão destes trabalhos, sendo realizado mais alguns cortes, restando 13 trabalhos.
12. Na terceira e última fase de seleção, realizando a leitura completa, restaram 6 publicações, que serão utilizadas para embasar, justificar e responder à questão de pesquisa. Estes trabalhos selecionados são citados no Capítulo 4

Tabela 1 – *String* de busca pré refinamento

("YOLO" OR "YOU ONLY LOOK ONCE") AND ("TRANSIT" OR "TRANSITO") AND ("SIGNAL" OR "PLACAS" OR "PLACA")

Fonte: Autor (2021)

Tabela 2 – *String* de busca após refinamento

("YOLOv5") AND ("TRAFFIC" OR "TRANSITO" OR "ROAD") AND ("PLATE" OR "PLACAS" OR "PLACA" OR "SIGNS")

Fonte: Autor (2021)

A Tabela 1 mostra a primeira *string* de busca utilizada para o levantamento dos trabalhos correlatos, ela precisou passar por um refinamento devido seu resultado retornar muitos trabalhos (482), se tornando inviável a análise de todos. Já com a Tabela 2 foi possível diminuir esse número para um total de 88 publicações, e utilizando como período de pesquisa publicações a partir de 2016. Com essa diminuição significativa, foi possível fazer uma análise inicialmente superficial destes trabalhos, até chegar aos com maior relevância para essa pesquisa.

Dando continuidade ao quarto item com base na Figura 1, como o tema em questão existem diversas técnicas e métodos existentes, como citado anteriormente, foi decidido por utilizar o método YOLOv5, que pelos estudos indica ser eficaz para a solução do problema, além de ser um modelo utilizado em diversos problemas para a detecção de objetos.

A quinta atividade trata do desenvolvimento prático do modelo proposto, tendo início na criação da base de dados, seguido do treinamento da rede e coletas dos dados para análise e validação dos resultados. Devido ao alto poder computacional necessário para o treinamento dos algoritmos de Inteligência Artificial, foi escolhido o *Google Colab* como ambiente de desenvolvimento para realizar o treinamento citado, uma vez que disponibiliza o processamento em GPU (Graphical Processing Unit) de alto desempenho sendo executado em nuvem. A arquitetura do modelo foi desenvolvida em linguagem *Python* com o uso do código da YOLOv5. O controle de versionamento foi feito através do Git via repositório GitHub[2], disponível publicamente, juntamente com o *dataset* utilizado.

Para a atividade da base de dados, foi criado uma *dataset* próprio, devido não haver algum disponível com tamanho razoável de placas de trânsito brasileiras, e como citado anteriormente, se encontra no mesmo repositório, para que demais pesquisadores possam utilizá-lo em seus projetos. Contendo três classes, regulamentação, advertência e indicação, tendo um total de 3283 imagens, divididas entre treinamento e validação. Esta base de dados será usada para treinar o algoritmo e após o treinamento será realizada a coleta dos dados referentes aos resultados alcançados pelo modelo.

[2]https://github.com/jean2612/Dataset-Placas-Transito

3 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentadas as definições dos conceitos relacionados aos assuntos abordados neste trabalho, articulando com a revisão da literatura. Sendo eles, princípios de inteligência artificial, *machine learning, deep learning*, redes neurais, redes neurais convolucionais. Apresentando o método YOLO e requisitos de *software*, finalizando com conceitos de *datasets*.

3.1 Inteligência artificial

A Inteligência Artificial (IA) pode ser definida como o ramo da ciência da computação que se empenha em estudos da automação do desempenho inteligente, tendo como principal interesse realizar de forma efetiva o entendimento e a aplicação inteligente de técnicas para solucionar problemas, para planejamento e a comunicação com inúmeros problemas práticos (LUGER, 2013). Apesar da variedade de problemas tratados pela inteligência artificial, o autor traz oito características importantes que, segundo ele, parecem ser comuns a todas as divisões da área, são elas:

1. O uso do computador para executar raciocínio, reconhecimento de padrões, aprendizado ou outras formas de inferência.
2. Um foco em problemas que não respondem a soluções algorítmicas. Isso implica a utilização de busca heurística como uma técnica de IA para a solução de problemas.
3. Um interesse na solução de problemas utilizando informação inexata, faltante ou insuficientemente definida, e o uso de formalismos representacionais que possibilitem ao programador compensar esses problemas.
4. Raciocínio que utiliza as características qualitativas significativas de uma situação.
5. Uma tentativa de tratar de questões que envolvem tanto significado semântico como forma sintática.
6. Respostas que não são nem exatas, nem ótimas, mas que são “suficientes” em um certo sentido. Isso é resultado de se utilizar essencialmente métodos de solução de problemas baseados em heurísticas em situações em que resultados ótimos, ou exatos, são caros demais ou mesmo impossíveis.
7. O uso de grandes quantidades de conhecimento específico de um domínio para resolver problemas. Essa é a base dos sistemas especialistas.

8. O uso de meta conhecimento para produzir um controle mais sofisticado sobre as estratégias de resolução de problemas. Embora esse seja um problema muito difícil, tratado por relativamente poucos sistemas, ele vem emergindo como uma área essencial de pesquisa.

3.2 Machine Learning

Machine Learning (aprendizado de máquina) procura otimizar um critério de desempenho usando dados de exemplo ou experiências anteriores. Temos um modelo definido até alguns parâmetros, e aprendizagem é a execução de um *software* para otimizar os parâmetros do modelo usando os dados de treinamento ou experiência. O modelo pode ser preditivo para fazer previsões no futuro, ou descritivo para obter conhecimento dos dados, ou ambos. O aprendizado de máquina usa a teoria da estatística na construção de modelos matemáticos, porque a tarefa principal é fazer inferências a partir de uma amostra. O papel da ciência da computação é duplo: primeiro, no treinamento, é preciso possuir algoritmos eficientes para resolver o problema de otimização, bem como para armazenar e processar a grande quantidade de dados que geralmente se possui. Em segundo lugar, uma vez que um modelo é aprendido, sua representação e solução algorítmica para inferência também precisam ser eficientes. Em certas aplicações, a eficiência do algoritmo de aprendizagem ou inferência, nomeadamente, a sua complexidade no espaço e no tempo, pode ser tão importante quanto a sua precisão preditiva. (ALPAYDIN, 2020)

3.3 *Deep Learning*

Deep Learning (aprendizado profundo) é o subcampo da Inteligência Artificial que se concentra na criação de grandes modelos de rede neural que conseguem tomar decisões baseadas em dados precisos. O aprendizado profundo é particularmente adequado para contextos onde os dados são complexos e onde existem grandes conjuntos de dados disponíveis. Hoje, a maioria das empresas online e tecnologias de consumo de ponta usam aprendizado profundo. Entre outras coisas, o Facebook[3] usa aprendizado

[3] www.facebook.com

profundo para analisar texto em conversas online. Google[4], Bing[5], entre outras, usam aprendizado profundo para pesquisa de imagens e também para tradução automática. Alguns *smartphones* mais modernos possuem sistemas de aprendizado profundo em execução; por exemplo, o aprendizado profundo agora é a tecnologia padrão para reconhecimento de fala e também para detecção de rosto em câmeras digitais. No setor de saúde, o aprendizado profundo é usado para processar imagens médicas (radiografias, tomografia computadorizada e ressonância magnética) e diagnosticar problemas de saúde. Ademais, o aprendizado profundo está no centro dos carros autônomos, onde é usado para localização, mapeamento, planejamento, direção de movimento e percepção do ambiente, bem como rastrear o estado do motorista.

O aprendizado profundo surgiu a partir de pesquisas em inteligência artificial e aprendizado de máquina. O campo moderno do aprendizado de máquina baseia-se nos dois últimos tópicos: computadores que podem aprender com exemplos e pesquisa de redes neurais. O aprendizado de máquina envolve o desenvolvimento e a avaliação de algoritmos que permitem a um computador extrair (ou aprender) funções de um conjunto de dados (conjuntos de exemplos). Para entender o que significa aprendizado de máquina, precisamos entender três termos: conjunto de dados, algoritmo e função. Em sua forma mais simples, um conjunto de dados é uma tabela em que cada linha contém a descrição de um exemplo de um domínio e cada coluna contém as informações de um dos recursos em um domínio. Um algoritmo é um processo (ou receita, ou fluxo de instruções) que um computador pode seguir. No contexto do aprendizado de máquina, um algoritmo define um processo para analisar um conjunto de dados e identificar padrões recorrentes nos dados. Uma função é um mapeamento determinístico de um conjunto de valores de entrada para um ou mais valores de saída. O fato de o mapeamento ser determinístico significa que, para qualquer conjunto específico de entradas, uma função sempre retorna as mesmas saídas (KELLEHER, 2019).

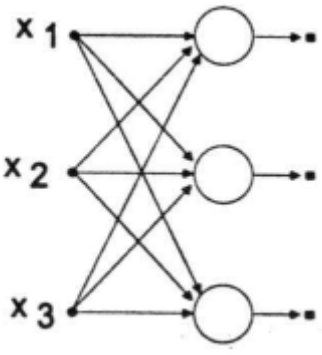
3.4 Redes Neurais

Para Braga, Carvalho e Ludermir (2000) Redes Neurais Artificiais (RNAs) ou apenas redes neurais, compostos por simples unidades de processamento, denominadas nodos, cujo objetivo é calcular funções matemáticas. Essas unidades de processamento

[4] www.google.com
[5] www.bing.com

estão distribuídas em uma ou mais camadas interligadas por diversas conexões. Cada conexão conta com um peso, que serve para armazenar o conhecimento representado pelo modelo e encaminhar cada entrada para um neurônio da rede como na Figura 2. Cada um dos nerônios possuem uma função de ativação, que define se o neurônio vai propagar ou inibir o sinal. Durante o processo de treinamento são atribuídos pesos para cada conexão, que servirão como parte do processo de decisão, de quais neurônios serão ativados na próxima camada, simulando as sinapses de um sistema nervoso animal. A função de ativação define a intensidade da ativação, que neste caso possui diversos tipos, onde a escolha desta determina como devem ser as saídas de cada neurônio. Para a solução de problemas com uso de RNAs, eles passam inicialmente por um processo de aprendizagem, onde é basicamente apresentado uma série de exemplos para a rede, fazendo com que consiga extrair de forma automática características necessárias para poder representar a informação fornecida. Estas características anteriormente extraídas, são as usadas para gerar a solução para o problema. Haykin (2007) complementa que com a capacidade natural de armazenar o conhecimento experimental e torná-lo disponível para uso. Assemelhando-se ao cérebro, que o conhecimento adquirido pela rede é através de um processo de aprendizagem e que as forças de conexão entre neurônios são usadas para armazenar o conhecimento adquirido.

Figura 2 – Rede de camada única



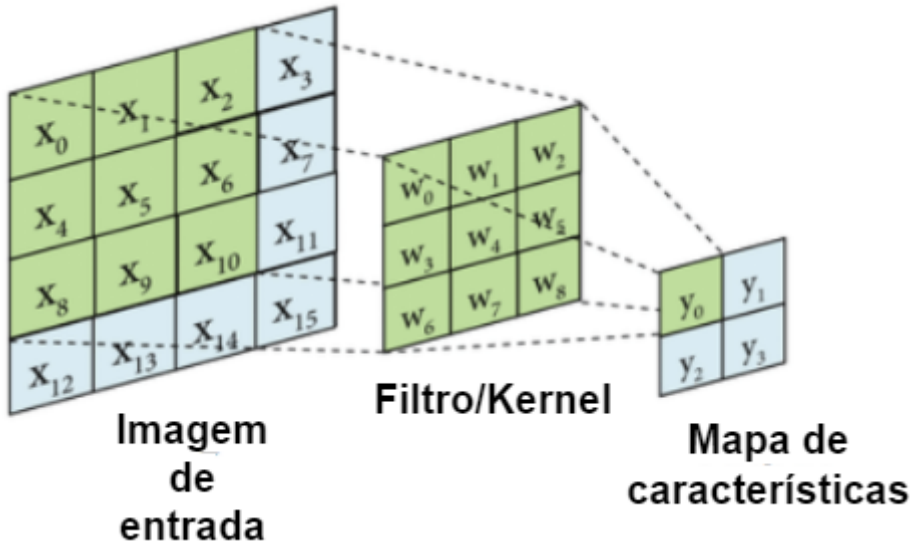
Fonte: Adaptado de Braga, Carvalho e Ludermir (2000)

3.5 Redes Neurais Convolucionais (CNNs)

São um tipo especializado de rede neural para processamento de dados com uma topologia conhecida, como uma grade. Os exemplos incluem dados de série temporal, que podem ser considerados como uma grade 1D que coleta amostras em intervalos de

tempo regulares, e dados de imagem, que podem ser considerados como uma grade 2D de *pixels*. As redes convolucionais têm sido extremamente bem-sucedidas em aplicações práticas. O nome “rede neural convolucional” indica que a rede emprega uma operação matemática chamada convolução. A convolução é um tipo especializado de operação linear. As redes convolucionais são simplesmente redes neurais que usam convolução no lugar da multiplicação geral da matriz em pelo menos uma de suas camadas, no contexto de imagens, esse processo pode ser associado como um filtro (também chamado de *kernel* nesse caso), que transforma uma imagem de entrada, onde esses filtros são representados por uma matriz a qual é utilizada numa operação entre as matrizes, como visto na Figura 3 de modo que é feita uma multiplicação dos *pixels* da imagem com o *pixel* do filtro e após essas multiplicações feitas, sua soma é armazenada em um mapa de características. Esta operação é aplicada diversas vezes em diferentes regiões da imagem. A cada aplicação, a região é alterada por um parâmetro conhecido como *stride*. *Stride* é um parâmetro do filtro da rede neural que modifica a quantidade de movimento sobre a imagem ou vídeo. Por exemplo, se o passo de uma rede neural for definido como 1, o filtro moverá um *pixel*, ou unidade, por vez, o que significa que a transformação será aplicada em todos os *pixels* da imagem (GOODFELLOW; BENGIO; COURVILLE, 2016).

Figura 3 – Convolução



Fonte: Adaptado de https://deepai.org/

Conforme Michelucci (2019) para o contexto de redes neurais, a convolução é feita entre tensores. A operação obtém dois tensores como entrada e produz um tensor como saída. Sendo a convolução o resultado da multiplicação de elementos dos dois

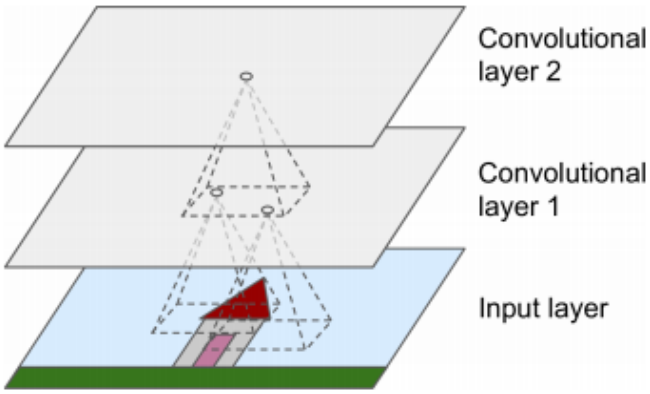
tensores na mesma posição e posteriormente fazendo um somatório dessas multiplicações, considerando um filtro do tamanho da imagem. Em uma imagem maior que o filtro, deve-se posicionar o filtro bem a esquerda e acima na imagem, aplicar a convolução, fazer um deslocamento para direita e aplicar convolução novamente até o fim da linha, então retornando para primeira coluna, mas pulando para próxima linha e assim sucessivamente até o filtro passar em toda imagem.

Haykin (1998) *apud* (LECUN; BENGIO, 1995) a forma estrutural de uma CNN é dividida em três, extração de características, mapeamento de características e subamostragem.

1. **Extração de características:** Para cada neurônio leva suas entradas sinápticas da camada anterior, e assim forçar a extração do local de características;
2. **Mapeamento de características:** Cada camada computacional é composta por vários mapeamentos de características, com cada mapeamento sendo um plano com os neurônios individuais compartilhando o mesmo conjunto de pesos sinápticos;
3. **Subamostragem:** Cada camada convolucional é seguida por uma camada computacional que realiza uma média local e subamostragem onde o tamanho do mapeamento de características é reduzido.

Nos últimos anos, graças ao aumento do poder computacional, as CNNs conseguiram atingir um desempenho sobre-humano em algumas tarefas visuais complexas. Potencializam os serviços de pesquisa de imagens, carros autônomos, sistemas automáticos de classificação de vídeo, entre outros. As CNNs não se restringem à percepção visual, elas também são bem-sucedidas em outras tarefas, como reconhecimento de voz ou processamento de linguagem natural. Os campos receptivos de diferentes neurônios podem se sobrepor e, juntos, eles revestem todo o campo visual. O bloco de construção mais importante de uma CNN é a camada convolucional, os neurônios na primeira camada de convolução não estão conectados a cada *pixel* na imagem de entrada, mas apenas aos *pixels* em seus campos receptivos, observado na Figura 4. Por sua vez, cada neurônio na segunda camada convolucional está conectado apenas a neurônios localizados dentro de um pequeno retângulo na primeira camada. Essa arquitetura permite que a rede se concentre em pequenos recursos de nível inferior na primeira camada oculta, depois os monte em recursos maiores de nível superior na próxima camada oculta e sucessivamente. Essa hierarquia é comum em imagens reais, um motivo das CNNs funcionarem bem para reconhecimento de imagens (GÉRON, 2019).

Figura 4 – Camadas CNN com campos receptores locais retangulares

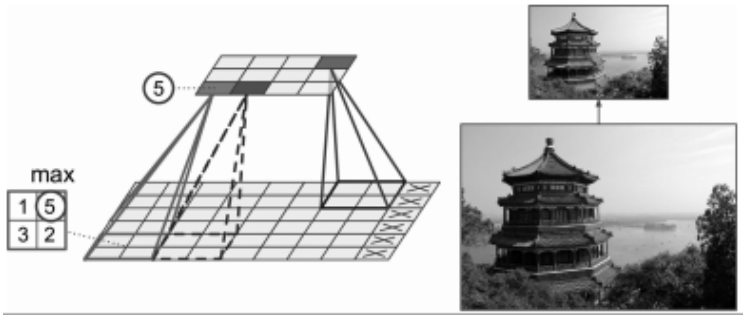


Fonte: Géron (2019)

O *pooling* tem o objetivo de diminuir a imagem de entrada para reduzir a carga computacional, o uso de memória e o número de parâmetros. Assim como nas camadas convolucionais, cada neurônio na camada *pooling* é conectado às saídas de um número limitado de neurônios na camada anterior, localizados dentro de um pequeno campo receptivo retangular. Um neurônio de *pooling* não tem pesos, ele apenas une as entradas, a camada de *pooling* funciona de forma independente em todos os canais de entrada, portanto, a profundidade de saída é a mesma de entrada (GÉRON, 2019).

O *pooling* ajuda a fazer com que a representação se torne aproximadamente invariável a pequenas traduções da entrada. A invariância para a tradução significa que, se a entrada for traduzida em uma pequena quantidade, os valores da maioria das saídas combinadas não mudam. A invariância para a tradução local pode ser uma propriedade muito útil se nos preocupamos mais se algum recurso está presente ou exatamente onde ele está. O uso de *pooling* pode ser visto como uma adição infinitamente forte de que a função que a camada aprende deve ser invariável para pequenas traduções. Quando essa suposição está correta, ela pode melhorar muito a eficiência estatística da rede.

Figura 5 – Camada *MaxPooling* (*kernel pooling* 2×2, *stride* de 2, sem *padding*)



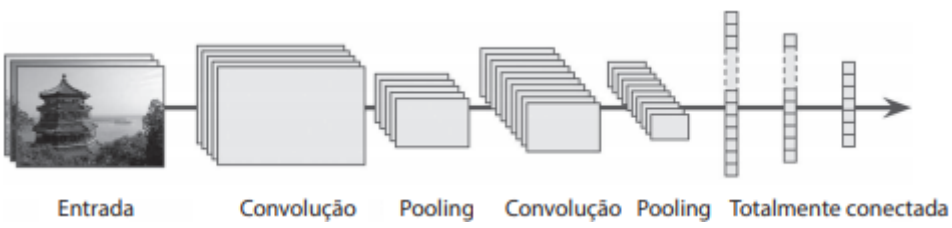
Fonte: Géron (2019)

A Figura 5 mostra que a operação de *MaxPooling* retira o maior elemento de determinada região da matriz, no exemplo é utilizado um filtro de 2x2, um *stride* de 2 e um preenchimento zero denominado *padding*. Tendo assim que o valor máximo de cada *kernel* chegue para a próxima camada, reduzido em 75% o tamanho da imagem, já que para cada 4 *pixels* é extraído apenas 1 (o de maior valor devido ser *MaxPooling*) para a próxima camada.

O *pooling* em regiões espaciais produz invariância para a translação, mas se agruparmos as saídas de convoluções parametrizadas separadamente, os recursos podem aprender a quais transformações se tornar invariáveis. Assim, como o *pooling* resume as respostas em toda uma vizinhança, é possível usar menos unidades de *pooling* do que unidades de detector, relatando estatísticas resumidas para regiões de *pool* espaçadas k *pixels* em vez de 1 *pixel*. Isso melhora a eficiência computacional da rede porque a próxima camada tem cerca de k vezes menos entradas para processar. Quando o número de parâmetros na próxima camada é uma função de seu tamanho de entrada (como quando a próxima camada está totalmente conectada e com base na multiplicação da matriz), essa redução no tamanho de entrada também pode resultar em eficiência estatística aprimorada e requisitos de memória reduzidos para armazenar os parâmetros.

Para muitas tarefas, o *pooling* é essencial para lidar com entradas de tamanhos variados. Por exemplo, se quisermos classificar imagens de tamanho variável, a entrada para a camada de classificação deve ter um tamanho fixo. Isso geralmente é realizado variando-se o tamanho de um deslocamento entre as regiões de *pooling*, de modo que a camada de classificação sempre receba o mesmo número de estatísticas de resumo, independentemente do tamanho da entrada. Por exemplo, a camada final de *pooling* da rede pode ser definida para produzir quatro conjuntos de estatísticas de resumo, um para cada quadrante de uma imagem, independentemente do tamanho da imagem (GOODFELLOW; BENGIO; COURVILLE, 2016).

Figura 6 – Arquitetura CNN típica



Fonte: Géron (2019)

Na etapa após o último o *pooling* temos a etapa de *flattening*(achatamento),

como observamos na Figura 6 onde mostra a arquitetura típica da CNN onde temos a rede totalmente conectada, como o próprio nome sugere, ocorre uma transformação da matriz resultado das operações de convolução e *pooling*, transformando-a num vetor unidimensional. Em seguida, a camada densa recebe como entrada o resultado da etapa de *flattering* sendo composta por neurônios logísticos que calculam os pesos e realizam a predição sobre a saída. (BARROS, 2021), (DU *et al.*, 2016)

3.6 Método YOLO

Segundo Redmon *et al.* (2016) o método YOLO (*You Only Look Once*) reformula a detecção de objetos ao transformá-la em um problema de regressão, partindo unicamente dos *pixels* de uma imagem, resultando nas probabilidades por classe, nas coordenadas e nas dimensões das predições que delimitam os objetos. Além de simples, já que utiliza uma abordagem fim-a-fim com uma única rede convolucional, também apresenta desempenho competitivo em função do tempo. Diferente de outros métodos de detecção de objetos baseados em redes convolucionais, YOLO necessita apenas de uma propagação da entrada pela rede para gerar predições, apresentando desempenho compatível com métodos de detecção de objetos em tempo real, mantendo a acurácia preditiva.

A primeira versão do YOLO possui diversas restrições espaciais nas previsões de caixa delimitadora, uma vez que cada célula da grade prevê apenas duas caixas e pode ter apenas uma classe. Essa restrição espacial limita o número de objetos próximos que ele consegue prever. Usa recursos relativamente grosseiros para prever caixas delimitadoras, devido a arquitetura possuir várias camadas de redução da resolução da imagem de entrada. A principal fonte de erro são as localizações incorretas.

Usando um novo método de treinamento em várias escalas, o mesmo modelo YOLOv2 pode ser executado em tamanhos variados, oferecendo uma troca fácil entre velocidade e precisão, um dos principais focos de mudança foi em melhorar a recuperação e a localização, mantendo a precisão da classificação. A visão do computador geralmente tende a redes maiores e mais profundas. O melhor desempenho geralmente depende do treinamento de redes maiores ou da combinação de vários modelos. Em vez de aumentar a rede, ela foi simplificada e fazendo com que ela tenha uma maior facilidade de aprendizado.

O melhor desempenho geralmente depende do treinamento de redes maiores ou

da combinação de vários modelos. No entanto, com o YOLOv2[6], se tem um detector mais preciso e rápido. Em vez de aumentar a rede, ela foi simplificada, tornando sua representação mais fácil de aprender. A primeira versão do YOLO usa uma resolução de entrada de 448×448. Com a adição de caixas de âncora, foi mudada a resolução para 416x416. No entanto, como o modelo usa apenas camadas convolucionais e de *pooling*, ele pode ser redimensionado em tempo real. Com o intuito de que o YOLOv2 seja robusto para rodar em imagens de tamanhos diferentes, então foi treinado isso no modelo. Em vez de fixar o tamanho da imagem de entrada, a rede é mudada a cada poucas iterações.

É esperado que a detecção de objetos seja precisa, mas também que seja rápida. A maioria das aplicações que usam detecção, como robótica ou carros com direção automática, dependem de previsões de baixa latência. A fim de maximizar o desempenho, foi projetado o YOLOv2 para ser rápido desde o início. O *framework* YOLO usa uma rede customizada baseada na arquitetura GoogLeNet[7].

Sua proposta é de um mecanismo de treinamento em conjunto de classificação e detecção de dados. Este método usa imagens classificadas para aprender informações específicas de detecção, como previsão de coordenadas de caixa delimitadora, bem como classificar objetos comuns. Usando as imagens apenas com rótulos de classe para expandir o número de categorias que pode detectar. Em seu treinamento houve uma mistura de imagens entre os conjuntos de dados de detecção e classificação. Quando a rede identifica uma imagem rotulada para detecção, ela é capaz de retropropagar com base na função de perda completa do YOLOv2 . Quando ele identifica uma imagem de classificação, é feita apenas a retropropagação de perda das partes específicas da classificação da arquitetura. Essa abordagem apresenta alguns desafios. Os conjuntos de dados de detecção têm apenas objetos comuns e rótulos gerais, como “cachorro” ou “barco”. Os conjuntos de dados de classificação têm uma gama muito mais ampla e profunda de rótulos. Para realizar o treinamento em ambos os conjuntos de dados, é necessária uma maneira coerente para mesclar esses rótulos. A maioria das abordagens de classificação usa uma camada *softmax* em todas as categorias possíveis para calcular a distribuição de probabilidade final. O uso de uma *softmax* assume que as classes são mutuamente exclusivas. Isso apresenta problemas para combinar conjuntos de dados. Em vez disso, foi usado um modelo com vários rótulos para combinar os conjuntos de dados que não pressupõem exclusão mútua. Essa abordagem ignora toda a estrutura que conhecemos sobre os dados, por exemplo, que todas as classes COCO (*Common Objects*

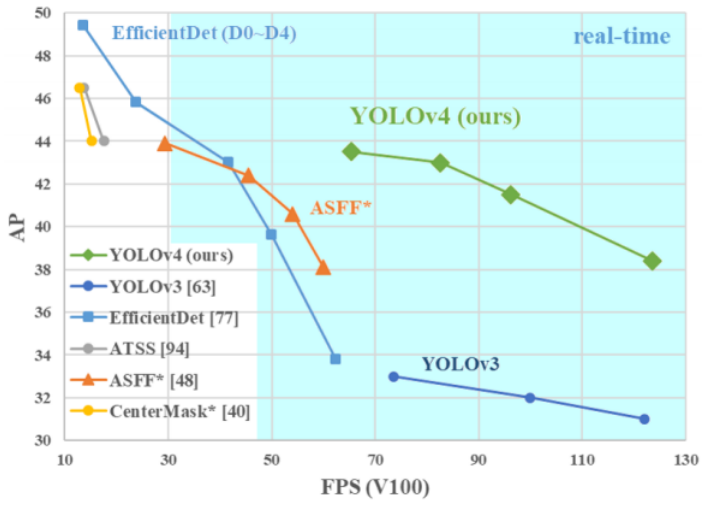
[6]https://pjreddie.com/darknet/yolov2/
[7]https://arxiv.org/abs/1409.4842v1

in Context) são mutuamente exclusivas. O YOLOv2 até o lançamento de seu sucessor foi mais rápido do que outros sistemas de detecção em uma variedade de conjuntos de dados de detecção. Além disso, ele pode ser executado em uma variedade de tamanhos de imagem para fornecer uma compensação suave entre velocidade e precisão (REDMON; FARHADI, 2017).

Já o YOLOv3[8] prevê uma pontuação de objetividade para cada caixa delimitadora usando regressão logística. Deve ser onde a caixa delimitadora anterior se sobrepõe a um objeto mais do que as outras caixas delimitadoras anteriores. Se a caixa delimitadora anterior não é a melhor, mas se sobrepõe a um objeto de verdade fundamental em mais do que algum limite, ignoramos a previsão a seguir. Usa-se o limite de 0,5. Ao contrário do sistema, apenas atribui uma caixa delimitadora antes de cada objeto de verdade terrestre. Se uma caixa delimitadora anterior não for atribuída a um objeto de verdade fundamental, não há perda de coordenadas ou previsões de classe, apenas objetividade.

Isso indica que o YOLOv3 é um detector muito forte que se destaca na criação de caixas delimitadoras para objetos. Portanto, o desempenho cai à medida que o limite de *Intersection over Union* (IoU) aumenta, indicando que o YOLOv3 se esforça para alinhar as caixas perfeitamente com o objeto. No passado, YOLO possuía dificuldades em detectar pequenos objetos. No entanto, agora há uma reversão nessa tendência. Com as novas previsões em várias escalas, se nota que o YOLOv3 tem um desempenho relativamente alto. No entanto, ele tem um desempenho comparativamente pior em objetos de tamanho médio e grande. (REDMON; FARHADI, 2018)

Figura 7 – Comparativo YOLOv4 e principais detectores



Fonte: Adaptado de Bochkovskiy, Wang e Liao (2020)

[8]https://pjreddie.com/darknet/yolo/

A Figura 7 mostra a comparação do YOLOv4[9] proposto pelos autores com outros detectores de objetos de última geração. O YOLOv4 roda duas vezes mais rápido do que *EfficientDet* com desempenho comparável. Tem uma melhoria no mAP (*mean Average Precision*) e *FPS* (*Frames per Second*) comparado ao YOLOv3 em 10% e 12%, respectivamente.

Bochkovskiy, Wang e Liao (2020) propuseram realizar um equilíbrio ideal entre a resolução da rede de entrada, o número da camada convolucional, o número do parâmetro e o número de saídas da camada. Selecionar blocos adicionais para aumentar o campo receptivo para obter o melhor método de agregação de parâmetros de diferentes níveis de *backbone* para diferentes níveis de detector. A fim de tornar o detector projetado mais adequado para o treinamento em uma única *GPU*, foram feitos *designs* e melhorias adicionais da seguinte forma:

Mosaic representa um novo método de aumento de dados que combina quatro imagens de treinamento. Assim, quatro contextos diferentes são misturados, enquanto anteriormente era usado combinações com apenas duas imagens de entrada. Isso permite a detecção de objetos fora de seu contexto normal. Além disso, a normalização em lote calcula as estatísticas de ativação de quatro imagens diferentes em cada camada. Isso reduz significativamente a necessidade de um tamanho grande de mini lote.

Self-Adversarial Training (SAT) também representa uma nova técnica de aumento de dados que opera em dois estágios para frente e para trás. No primeiro estágio, a rede neural altera a imagem original em vez dos pesos da rede. Dessa forma, a rede neural executa um ataque adversário a si mesma, alterando a imagem original para criar a ilusão de que não há nenhum objeto desejado na imagem. No segundo estágio, a rede neural é treinada para detectar um objeto nesta imagem modificada da maneira normal (BOCHKOVSKIY; WANG; LIAO, 2020).

3.7 Requisitos de *Software*

Nesta seção será apresentado o que são os requisitos de *software*, de acordo com Sommerville (2011) os requisitos de *software* sendo frequentemente classificados como requisitos funcionais e requisitos não funcionais:

1. Requisitos funcionais: São especificações de quais serviços o sistema deve

[9] https://arxiv.org/abs/2004.10934

conseguir fornecer, como ele deve reagir para as entradas específicas e como deve ser seu comportamento para uma determinada situação. É comum em alguns casos que os requisitos funcionais especificam o que o sistema não deve fazer.

2. Requisitos não funcionais: São restrições aos serviços ou funções oferecidas pelo sistema. Sendo incluídas nessas restrições, tempo, restrições de processo de desenvolvimento, restrições impostas por normas. De forma distinta às características individuais ou serviços do sistema, os requisitos não funcionais, diversas vezes são aplicados ao sistema na totalidade.

Em prática, notar a diferença entre os tipos de requisitos não é uma tarefa tão fácil quanto a definição anterior. Pois um requisito relacionado a proteção de usuário, por exemplo, como a limitação de acessos em determinados setores do sistema, pode parecer um requisito não funcional, no entanto, ao ser desenvolvido em mais detalhes, pode ser preciso gerar outros requisitos funcionais, como a necessidade de realizar a autenticação do usuário no sistema.

Com isso é notório que os requisitos não são independentes, eles em diversas vezes gerando ou restringindo outros requisitos. Desta forma, os requisitos de sistema não são apenas para especificar ou caracterizar a necessidade do sistema, sendo feita a especificação das funcionalidades necessárias para garantir que esses serviços sejam entregues de forma correta.

3.7.1 Requisitos funcionais

Os requisitos funcionais de um sistema visa descrever o que ele deve fazer. Eles são baseados no tipo de *software* que está sendo desenvolvido, de quem são os possíveis usuários e de qual será a abordagem geral adotada pela organização ao serem escritos os requisitos. Quando são descritos com requisitos de usuário, os requisitos funcionais tem a característica de serem descritos de forma mais simplificada, para ser de fácil entendimento para o usuário do sistema. Contudo, requisitos funcionais mais específicos devem descrever de forma detalhada as funções do sistema, suas entradas e saídas, além de suas exceções. Os requisitos funcionais do sistema variam de requisitos gerais, que devem abranger qual o funcionamento do sistema, até requisitos muito específicos, que retratam como deve ser a forma de trabalho de uma organização. Esses requisitos funcionais dos usuários definem os recursos específicos a serem fornecidos pelo sistema.

Uma imprecisão durante a especificação dos requisitos pode acarretar diversos problemas de engenharia de *software*. Porém, é compreensível certas divergências, devido o desenvolvedor do sistema interpretar de maneira ambígua a forma de implementação para o sistema, sendo necessário realizar ajustes durante o processo, podendo ocasionar atraso na entrega e aumento nos custos de produção.

Presumivelmente, a especificação dos requisitos funcionais deve ser feita de forma completa e consistente, completa no sentido de que todos os serviços requisitados pelo usuário sejam definidos e consistentes de forma que os requisitos não tenham definições contraditórias. Porém, de forma prática, para sistemas grandes e complexos, se torna praticamente impossível atingir uma descrição completa e consistente dos requisitos. Um motivo para este fato se dá devido à especificação de um sistema ser complexo, se facilita cometer erros e omissões.

Outro motivo é que em um sistema de grande porte se faz necessário a existência de muitos *stakeholders*. Um *stakeholder* é uma pessoa ou uma organização que, de alguma maneira, é afetado pela ação do sistema. As partes interessadas têm necessidades diferentes e muitas vezes inconsistentes. Essas inconsistências podem não ser aparentes no início, quando os requisitos são especificados, portanto, são incluídos na especificação. Podem surgir problemas após uma análise mais aprofundada ou entrega do sistema ao cliente. (SOMMERVILLE, 2011)

3.7.2 Requisitos não funcionais

Como o nome sugere, requisitos não funcionais são requisitos que não estão diretamente relacionados ao serviço específico que um sistema fornece a seus usuários. Eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e ocupação de área. Uma alternativa a esse cenário seria os requisitos que definem restrições sobre a implementação do sistema, como as capacidades dos dispositivos de E/S ou as representações de dados usadas nas interfaces com outros sistemas.

Os requisitos não funcionais, como desempenho, proteção ou disponibilidade, normalmente especificam ou restringem as características do sistema na totalidade. Requisitos não funcionais são frequentemente mais críticos que requisitos funcionais individuais. Os usuários do sistema podem, geralmente, encontrar maneiras de contornar uma função do sistema que realmente não atenda a suas necessidades. No entanto, deixar

de atender a um requisito não funcional pode significar a inutilização de todo o sistema.

Embora seja muitas vezes possível identificar quais componentes do sistema implementam requisitos funcionais específicos, é frequentemente mais difícil relacionar os componentes com os requisitos não funcionais. A implementação desses requisitos pode ser difundida em todo o sistema. Há duas razões para isso:

1. Requisitos não funcionais podem afetar a arquitetura geral de um sistema em vez de apenas componentes individuais. Por exemplo, para assegurar que sejam cumpridos os requisitos de desempenho, será necessário organizar o sistema para minimizar a comunicação entre os componentes.
2. Um único requisito não funcional, tal como um requisito de proteção, pode gerar uma série de requisitos funcionais relacionados que definam os serviços necessários no novo sistema. Além disso, também podem gerar requisitos que restrinjam requisitos existentes.

Os requisitos não funcionais surgem por meio das necessidades dos usuários, devido a restrições de orçamento, políticas organizacionais, necessidade de interoperabilidade com outros sistemas de *software* ou *hardware*, ou a partir de fatores externos, como regulamentos de segurança ou legislações de privacidade. Os requisitos não funcionais podem ser provenientes das características requeridas para o *software* (requisitos de produto), da organização que desenvolve o *software* (requisitos organizacionais) ou de fontes externas:

1. Requisitos de produto. Esses requisitos especificam ou restringem o comportamento do *software*. Exemplos incluem os requisitos de desempenho quanto à rapidez com que o sistema deve executar e quanta memória ele requer, os requisitos de confiabilidade que estabelecem a taxa aceitável de falhas, os requisitos de proteção e os requisitos de usabilidade.
2. Requisitos organizacionais. Esses são os requisitos gerais de sistemas derivados das políticas e procedimentos da organização do cliente e do desenvolvedor. Exemplos incluem os requisitos do processo operacional, que definem como o sistema será usado, os requisitos do processo de desenvolvimento que especificam a linguagem de programação, o ambiente de desenvolvimento ou normas de processo a serem usadas, bem como os requisitos ambientais que especificam o ambiente operacional do sistema.
3. Requisitos externos. Esse tipo abrange todos os requisitos que derivam de fatores

externos ao sistema e seu processo de desenvolvimento. Podem incluir requisitos reguladores, que definem o que deve ser feito para que o sistema seja aprovado para uso, por um regulador, tal como um banco central; requisitos legais, que devem ser seguidos para garantir que o sistema opere dentro da lei; e requisitos éticos, que assegurem que o sistema será aceitável para seus usuários e o público em geral.

3.8 *Datasets*

Para Deng *et al.* (2009) a era digital trouxe consigo uma enorme explosão de dados. As últimas estimativas colocam um número de mais de 3 bilhões de fotos no Flickr[10], um número semelhante de videoclipes no YouTube[11] e um número ainda maior de imagens no banco de dados da pesquisa de imagens do Google[12]. Modelos e algoritmos mais sofisticados e robustos podem ser propostos pela exploração dessas imagens, resultando em melhores aplicativos para os usuários indexarem, recuperarem, organizarem e interagirem com esses dados. Mas exatamente como esses dados podem ser utilizados e organizados ainda é um problema a ser resolvido. Cada conceito significativo no WordNet[13], possivelmente descrito por várias palavras ou frases de palavras, é chamado de "conjunto de sinônimos" ou "*synset*". Existem cerca de 80.000 *synsets* de substantivos no WordNet. No ImageNet[14], tem como pretensão fornecer em média $500 - 1000$ imagens para ilustrar cada synset. ImageNet, portanto, oferecerá dezenas de milhões de imagens ordenadas de forma limpa.

De acordo com Sharma *et al.* (2018) a descrição automática da imagem é a tarefa de produzir um enunciado em linguagem natural que descreva corretamente o conteúdo visual de uma imagem. Esta tarefa se tornou uma potente aliada nas soluções propostas com base em arquiteturas de aprendizagem profunda. As aplicações práticas dos sistemas de descrição automática de imagens incluem o aproveitamento de descrições para indexação ou recuperação de imagens e ajuda para pessoas com deficiência visual, transformando sinais visuais em informações que podem ser comunicadas por meio da tecnologia de texto para fala. O desafio científico é visto como alinhar, explorar e levar adiante as melhorias mais recentes na interseção da Visão Computacional e do

[10] https://www.flickr.com/
[11] www.youtube.com
[12] images.google.com
[13] https://wordnet.princeton.edu/
[14] https://www.image-net.org/

Processamento de Linguagem Natural.

Existem duas categorias principais de avanços responsáveis pelo aumento do interesse nesta tarefa. O primeiro é a disponibilidade de grandes quantidades de dados classificados. Conjuntos de dados relevantes incluem o conjunto de dados *ImageNet*, com mais de 14 milhões de imagens e 1 milhão de anotações de caixa delimitadora, e o conjunto de dados MS-COCO[15] (*Microsoft Common Objects in Context*), com 120.000 imagens e 5 vias anotações de legenda de imagem. O segundo é a disponibilidade de mecanismos de modelagem poderosos, como Redes Neurais Convolucionais, capazes de converter *pixels* de imagem em recursos de alto nível sem nenhuma engenharia manual de recursos.

3.8.1 *Softwares* para criação de *datasets*

Atualmente, as classificações de imagem têm alta demanda devido ao rápido crescimento e ao uso generalizado de Inteligência Artificial (IA) e Aprendizado de Máquina (ML). Os desenvolvimentos nessas áreas levaram a mais e mais imagens sendo classificadas e estão sendo usadas não apenas para atrações de imagens, mas também para previsões futuras em diferentes cenários. A visão computacional que os modelos IA e ML têm, pode visualizar diferentes objetos através de vídeos e fotos; por isso, é melhor que sejam classificados usando a técnica especial, e se realizados em grande escala, podem ser usados como dados para treinamento adicional.

Algumas das principais ferramentas gratuitas são:

- LabelImg;
- LabelMe;
- CVAT (*Computer Vision Annotation Tool*);
- VoTT (*Visual Object Togging Tool*);
- ImageLab.

O *software* LabelImg[16] é uma ferramenta de rotulagem de imagens de código aberto e gratuita. Após ser instalada, pode ser utilizada de forma *off-line*, tornando mais simples e rápida rotular as imagens e recuperar as salvas. Caso contrário, é uma

[15] https://cocodataset.org/
[16] https://github.com/tzutalin/labelImg

ferramenta muito simples e não tem nada de avançado. Além disso, ele suporta apenas caixas delimitadoras e nenhum outro tipo de método de rotulagem.

LabelMe[17] é considerada uma das ferramentas clássicas de rotulagem de imagens da indústria, construída pelo MIT (*Massachusetts Institute of Technology*) em um formato de código aberto. Sua melhor técnica de etiquetagem é usar a forma poligonal de etiquetagem, porém, seu nível de precisão provou ser extremamente baixo e desigual.

A ferramenta de anotação de visão computacional (CVAT[18]) é desenvolvida pela Intel. O software reitera a incorporação do *OpenCV*, lançado há 2 décadas pela gigante da tecnologia. Como pode ser esperado pelo software da Intel, o CVAT vem com ferramentas de anotação poderosas e de última geração. E embora familiarizar-se com todos os recursos e funcionalidades do software possa levar algum tempo, vale a pena o esforço para se acostumar com as toneladas de recursos oferecidos pela plataforma para anotação. Como um aplicativo da web, o software é fácil de instalar e escalar e oferece muitos instrumentos de automação, incluindo; *TensorFlow, API Object Detection*, anotação automática e outros. A única desvantagem da ferramenta é a interface do usuário complicada, que pode levar vários dias para ser dominada.

A ferramenta de marcação de objeto visual (VoTT[19]) é outra ferramenta poderosa de anotação de imagens. Desenvolvido pela *Microsoft*, o software traz uma interface interativa e intuitiva, facilitando para o usuário dominar as diversas funcionalidades e recursos oferecidos pela ferramenta. Além disso, a ferramenta facilita para os usuários criar um projeto sem mergulhar nos detalhes da documentação. VoTT é escrito em *React Language* com codificação limpa, além disso, o uso de algoritmos de aprendizagem profunda permite a detecção rápida e precisa de objetos. O VoTT está disponível como um aplicativo da web e também como um aplicativo eletrônico.

ImgLab[20] fornece vários tipos de rótulos, como pontos, círculos, caixas delimitadoras e polígonos. Ele também suporta vários formatos de arquivos de saída, incluindo dlib, XML, Pascal VOC e COCO. Além de ter a versão de *Desktop* via *download*, conta também com uma versão online que é independente de plataforma, rodando diretamente no navegador e sem pré-requisitos. Devido a versão online rodar diretamente no navegador, faz com que necessite de um mínimo de processamento e memória da máquina.

[17] https://github.com/wkentaro/labelme
[18] https://github.com/openvinotoolkit/cvat
[19] https://github.com/microsoft/VoTT
[20] https://github.com/NaturalIntelligence/imglab

4 TRABALHOS CORRELATOS

Durante as buscas, foram selecionados alguns trabalhos que utilizam de abordagem de reconhecimento de imagens, tanto de pedestre, quanto placas de trânsito, para as mais diversas aplicações. Nessa seção são abordados os principais deles.

4.1 Detecção de sinais de trânsito em tempo real com base no modelo de rede YOLO

Os autores Yang e Zhang (2020) traz uma breve comparação das versões da ferramenta e sua evolução ao longo dos anos, porém, o principal objetivo do trabalho é realizar uma comparação no desempenho entre o YOLOv3 e o YOLOv4 usando como teste sinais de trânsito chineses, pois optaram por uma melhor aplicação à situação da China, não utilizando o conjunto de dados alemão, embora seja popular. O conjunto de dados usado neste experimento é CSUST (*Chinese Traffic Sign Detection Benchmark*) onde foi selecionado cerca de 4000 imagens, divididas em quatro categorias de placas, são elas, as de aviso, de limite de velocidade, de instruções e as de proibição. Com este estudo eles concluíram que devido ao pequeno número de conjuntos de dados, a melhoria da precisão não é tão alta quanto o esperado, mas mesmo assim o YOLOv4 possui uma vantagem sobre seu antecessor.

A contribuição que ele traz é que mostra um número de imagens para se ter como base para o treinamento da rede. Um ponto negativo é que o trabalho possui poucas informações mais detalhadas. A única diferença deste trabalho para o que está sendo apresentado aqui é a utilização de um *dataset* do padrão de placas brasileiro e a utilização da CNN YOLOv5.

4.2 Reconhecimento de sinais de trânsito usando aprendizado profundo

Conforme os autores Qin e Yan (2021), eles trazem inicialmente em seu trabalho algumas das importâncias de se pesquisar sobre esse tema, pois os sinais de trânsito auxiliam efetivamente os motoristas no processo de direção e os mantêm dirigindo com muita segurança, informando os motoristas sobre o estado da estrada e os perigos potenciais. O reconhecimento de sinais de trânsito, como uma das partes importantes do sistema de assistência ao motorista, além de que se tornou muito valioso e muitos

trabalhos de pesquisa relevantes surgiram recentemente.

Na sequência relatam que geralmente existem duas etapas para um reconhecimento de sinais de trânsito típico. O primeiro é localizar e obter as informações dos sinais de trânsito em imagens oriundas de cenários reais, o que é conhecido como detecção de sinais de trânsito. A segunda etapa é realizar a classificação dos sinais de trânsito detectados. Embora o reconhecimento de sinais de trânsito tenha ganhado muita popularidade em sistemas de assistente de motorista, ainda existem inúmeras dificuldades para identificar sinais de trânsito do mundo real usando algoritmos de computador devido aos vários tamanhos de objetos visuais, deterioração de cor e oclusão parcial.

Para lidar com esses obstáculos, muitas abordagens e algoritmos têm sido propostos, como métodos de aprendizagem profunda que estão surgindo, várias abordagens de ponta têm sido amplamente aplicadas a essa área, como redes neurais convolucionais (CNNs). As CNNs trouxeram a possibilidade de aprender recursos a partir de uma quantidade de dados sem pré-processamento, evitando o processo de criação de recursos feitos à mão e absorvendo recursos generalizados. Além disso, a CNN já foi definida como um classificador de objetos em aprendizado de máquina que tem sido alavancado na classificação de sinais de trânsito.

Foi realizado um experimento para reconhecimento de sinais de trânsito com base no modelo de aprendizado profundo mais recente, o YOLOv5 em placas de trânsito da Nova Zelândia. Como primeiro passo realizaram a captura das imagens realistas de sinais de trânsito na cidade de *Auckland*. Para rotular as imagens de seu conjunto de dados, foi utilizada uma ferramenta gráfica de anotação de imagens, o *LabelImg*, que também será usado neste trabalho de conclusão de curso. Após gerados os arquivos de etiqueta com base em nosso conjunto de dados, o próximo passo é organizar os diretórios que salvam as imagens e etiquetas de treinamento e validação. A estrutura do modelo do YOLOv5 é a mesma do detector de objetos de estágio único.

Este trabalho tem grandes contribuições para esta pesquisa, devido ao uso do YOLOv5 e apresentar ser possível realizar o que é proposto neste trabalho de conclusão de curso, além de mostrar que o *LabelImg* também é utilizado para a rotulação de imagens. A principal diferença é que se utilizou um *dataset* próprio com placas de trânsito do padrão brasileiro.

4.3 Yolov4 para reconhecimento avançado de sinais de trânsito com dados de treinamento sintético gerados por várias GAN

O trabalho de Dewi *et al.* (2021) os autores discorrem sobre os *datasets* mais usados do mundo e, além disso, a questão de haver dificuldade em relação ao desenho e a cor dos sinais de trânsito, pois diferem para cada país, o que significa que é necessário recolher e marcar sinais de trânsito de diferentes países. Trazem também o uso de Redes Adversárias Geradoras (GAN) responsáveis por gerar imagens sintéticas. Eles utilizam como base de dados sinais proibitivos de Taiwan, base criada por eles, usando da ferramenta YOLOv3 e YOLOv4, fazendo um comparativo entre os resultados obtidos com elas. As principais contribuições desta pesquisa, segundo os autores, são: Imagens de sinais de proibição de alta qualidade, que são sintetizadas usando DCGAN, LSGAN e WGAN.

O desenvolvimento de uma solução baseada em CNN para tarefas de classificação de sinais de trânsito, bem como o aumento do conjunto de treinamento da CNN usando dados sintéticos criados, a fim de melhorar o desempenho de classificação e reconhecimento. Propuseram também uma configuração experimental com vários estilos GAN para gerar imagem sintética. A avaliação do modelo YOLOv3 e YOLOv4 inclui o mAP, tempo de detecção, IoU e operações de ponto flutuante (FLOPS). Os experimentos mostram que o uso de geração de dados de imagem sintética usando vários GAN pode melhorar a IoU e o desempenho de todos os modelos.

Outro ponto relevante trazido por eles é mostrar quais as etapas de como as ferramentas YOLO fazem para detectar objetos, que são divididas em seis, sendo elas, divisão da imagem de entrada em grades S x S, cada grade gera um número K de caixas delimitadoras baseado no cálculo de caixas de âncora. Após, independentemente do número de caixas, ele detecta apenas um item, prevendo probabilidades da classe condicional C (uma por classe para a similaridade da classe de objeto). Depois destes dois passos iniciais o sistema executa as camadas CNN para obter todos os recursos da imagem. Em sequência, compara a confiança IoU_{pred}^{truth} ideal das K caixas delimitadoras com o limite IoU^{truth}. Se $IoU_{pred}^{truth} > IoU^{truth}$, tem como significado que a caixa delimitadora contém o objeto, a caixa delimitadora não conteria o objeto de outra forma. Por fim, ao descobrir a categoria potencial para a categoria do objeto, o computador identifica o item específico. A última etapa resulta em uma imagem categorizada rotulada com a classe.

4.4 Um novo modelo de rede neural para detecção e reconhecimento de sinais de trânsito em condições extremas

Para os autores Wan *et al.* (2021), a detecção de sinais de trânsito é extremamente importante em sistemas de segurança de condução e transporte autônomos. No entanto, a detecção precisa de sinais de trânsito continua sendo um desafio, especialmente em condições extremas. Os autores propuseram a criação um novo modelo para detecção chamado *Traffic Sign Yolo* (TS-Yolo) baseado em uma rede neural convolucional, para melhorar a precisão de detecção e reconhecimento de sinais de trânsito, especialmente em condições de baixa visibilidade e visão extremamente restrita, utilizando como *dataset* o *Tsinghua-Tencent 100K* (TT100K), que conta com dados de placas de trânsito da China. Um método de aumento de dados de copiar e colar foi usado para construir um grande número de novas amostras com base em conjuntos de dados de sinais de trânsito existentes. Com base em YOLOv5, a convolução de profundidade mista (*MixConv*) foi empregada para misturar diferentes tamanhos de *kernel* em uma única operação de convolução, de modo que diferentes padrões com várias resoluções possam ser capturados. Além disso, o módulo de fusão de recursos de atenção (AFF) foi integrado para fundir os recursos com base na atenção de cenários de mesma camada para camadas cruzadas, incluindo conexões de salto curtas e longas, e até mesmo realizar a fusão inicial com ele mesmo. Os resultados experimentais trazidos por eles, demonstra que, usando o conjunto de dados YOLOv5 com aumento, a precisão foi 71,92, em 34,56 em comparação com os dados sem aumento, e a precisão média mAP 0,5 foi de 80,05, aumentada em 33,11 em comparação com os dados sem aumento.

Este artigo propôs um modelo baseado em CNN, denominado TS-Yolo, para detecção precisa de tráfego em condições climáticas severas, assim como mostrado na Figura 8, onde em (a) é possível ver a imagem em um clima nublado, (b) na neve, (c) ao anoitecer e (d) sofrendo encobrimento por sombra.

4.5 Reconhecimento automático de placas de carro em tempo real com base no detector YOLO

No trabalho de Laroca *et al.* (2018) os autores abordam a utilização do YOLO para reconhecimento em tempo real de placas de veículos usando de redes neurais convolucionais. Parte do trabalho foi a coleta de imagens para o banco de imagens,

Figura 8 – Resultado da detecção usando TS-Yolo

Fonte: Adaptado de Wan *et al.* (2021)

após esta coleta foi feito o treinamento de duas CNNs, uma para reconhecer o veículo de entrada e outra para detecção de sua placa. Eles avaliaram dois tipos de imagem, com cenários simples, e para isso usaram do fast-YOLO que se demonstrou eficaz, e com cenários realistas, onde não foi suficiente para execução da tarefa, para eles usaram do YOLOv2. No trabalho foi relatado que se teve sucesso na identificação do veículo e assim que identificado foi possível determinar onde se encontrava sua placa, de forma efetiva, mas a parte do reconhecimento dos caracteres precisa de uma maior taxa de acerto, porém já é possível conseguir alguns resultados. O trabalho possui relevância para este, devido utilizar CNNs e YOLO, que se pretende utilizar também, eles utilizam o fast-YOLO e o YOLOv2, mais atual na época. Porém, como diferencial será utilizado o YOLOv5 que é foco de pesquisa deste trabalho.

4.6 YOLO-LITE: Um Algoritmo Otimizado de Detecção de Objetos em Tempo Real para Computadores Sem GPU

No trabalho de Huang, Pedoeem e Chen (2018) foi estudado o comportamento próprio do YOLO-LITE, onde se teve como objetivo uma arquitetura que durante sua

execução obtivesse uma razão de 21 *frames* por segundo (FPS) em um computador onde não houvesse uma unidade de processamento gráfica (GPU). Enquanto a outras versões do método YOLO tem como principal objetivo a construção de uma rede neural grande o suficiente para que se possa atingir uma maior chance de acerto, o YOLO-LITE tem como foco utilizar ao limite de precisão a rede já preexistente para que o resultado seja alcançado com maior velocidade em comparação com outros. Como conclusão neste trabalho, foi possível verificar pelos gráficos apresentados que o YOLO-LITE e as suas redes neurais que são pequenas em relação a outros, não houve uma perda de precisão tão grande em relação a comparação feita e apresentou um alto desempenho em relação a quantidade de *fps*, sendo obtidos 21*fps* utilizando a YOLO-LITE em comparação com a *Tiny*-YOLOv2 que obeteve apenas 2,4 *fps* o que torna esta ferramenta de grande relevância para este trabalho que será produzido, pois a mesma poderá ser utilizada com forma de comparação ao YOLOv5 durante os testes de reconhecimento de imagens.

4.7 Considerações sobre os trabalhos correlatos

Ao término da análise dos trabalhos correlatos é visível a existência de diversos autores realizando pesquisa sobre o tema em questão, mostrando assim sua relevância. É possível analisar a utilização de diversas ferramentas para solução do problema, dentre elas o método YOLO, em todas suas versões, porém na abrangência brasileira não foi localizado trabalhos utilizando as placas locais, mas como foi realizado em diversos outros contextos de outros países, se mostrando algo com grande potencial.

A Tabela 3 apresenta quais ferramentas os trabalhos correlatos escolhidos utilizam, qual o país das placas, sendo o objetivo da identificação, e qual o *dataset* utilizado, ou se foi criado pelos autores. Nos trabalhos de Laroca *et al.* (2018) e de Huang, Pedoeem e Chen (2018) mesmo eles não tendo o objetivo de identificar placas de trânsito, foram escolhidos devido em um ser utilizado uma versão mais antiga do YOLO e para identificar não a placa de trânsito e sim a placa do carro, mostrando que há a possibilidade de utilizar a ferramenta para identificar qual a placa de trânsito que está sendo classificada, e não apenas seu tipo. O outro foi escolhido devido trazer uma versão do YOLO otimizada para ser utilizada em computadores sem GPU, fazendo assim com que quem não possui acesso a uma tenha a opção de utilizar a ferramenta para treinamento e detecção.

Tabela 3 – Principais características dos trabalhos correlatos

Trabalho	**Ferramenta utilizada**	**País das placas**	***Dataset* utilizado**
Yang e Zhang (2020)	YOLOv3 e YOLOv4	China	CSUST
Qin e Yan (2021)	YOLOv5	Nova Zelândia	Próprio
Dewi *et al.* (2021)	YOLOv3 e YOLOv4	Taiwan	Próprio
Wan *et al.* (2021)	YOLOv5 e TS-Yolo	China	TT100K
Laroca *et al.* (2018)	FAST-YOLO e YOLOv2	-	-
Huang, Pedoeem e Chen (2018)	YOLO-LITE	-	-
Trabalho proposto	YOLOv5	Brasil	Próprio

Fonte: Autor (2021)

5 DESENVOLVIMENTO DA FERRAMENTA

5.1 Análise dos requisitos

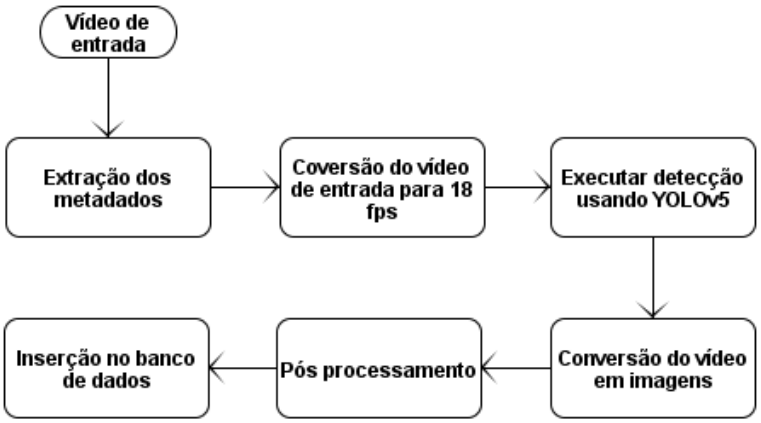
Nesta seção é descrito os requisitos de sistema, a proposta de modelo, os procedimentos de desenvolvimento do projeto experimental deste trabalho. Além de se realizar um estudo teórico sobre o tema e buscar outras que já são usadas na identificação de imagens de placas de trânsito, este projeto propõe um modelo que possa realizar a identificação usando o método YOLOv5, e após sua catalogação o armazenamento em um banco de dados, além de uma interface gráfica para o usuário.

O sistema deve conseguir identificar objetos de interesse, sendo passado como entrada um vídeo gravado com câmera com capacidade de armazenar a localização geográfica do trajeto filmado e sua saída ser a inserção dos objetos capturados em um banco de dados. Tendo como fluxo de trabalho os seguintes itens:

1. Inserir vídeo de entrada;
2. Extrair metadados;
3. Converter vídeo para 18 fps;
4. Converter vídeo para imagens;
5. Detectar objetos de interesse;
6. Salvar posição do objeto em arquivo de texto;
7. Armazenar resultados em banco de dados georreferenciado.

O fluxograma de funcionamento do *software* é observado na Figura 9, onde é inicialmente passado como entrada um vídeo gravado com câmera capaz de armazenar geolocalização de dados de onde está sendo feita a gravação, permitindo o armazenamento das coordenadas de GPS e o caminho exato de onde esse vídeo foi feito, um arquivo de vídeo é formado por uma sequência de imagens, áudio e metadado esse o responsável por armazenar tais dados geográficos.

Figura 9 – Fluxograma do *software* desenvolvido



Fonte: Autor (2022)

Após o vídeo de entrada ser inserido, como primeira etapa de processamento ocorre a extração dos metadados, feita utilizando o *script Python* GoPro2GPX[21] com o comando "gopro2gpx [video.mp4] [nome_saida]", em análise ao arquivo extraído, notou-se que a ocorrência do dado de localização se dava a cada 18 *frames*, por esse motivo, foi necessário realizar a conversão do vídeo de entrada para 18 *fps*, segunda etapa, para que a imagem e o metadado fiquem sincronizados fazendo com que cada *frame* seja atrelado a uma coordenada geográfica e evitando assim imagens desnecessárias.

Na terceira etapa é onde ocorre a detecção com o método YOLOv5 utilizando o vídeo convertido para 18 *fps* e os pesos da rede, que foi previamente treinada para a detecção. Os arquivos de saída da YOLO são arquivos no formato de texto, onde cada arquivo representa um *frame* do vídeo, fazendo necessário a conversão dele para imagens, realizada na quarta etapa. Já na quinta etapa é realizado o pós processamento, que consiste em fazer a comparação de cada *frame* com o arquivo de texto de saída da YOLO para ser feita a exclusão das imagens que não possuem *labels*, otimizando assim o armazenamento, evitando imagens desnecessárias. Finalizando na etapa seis, com a inserção dos dados no banco de dados georreferenciado PostgreSQL.

5.2 Classificação

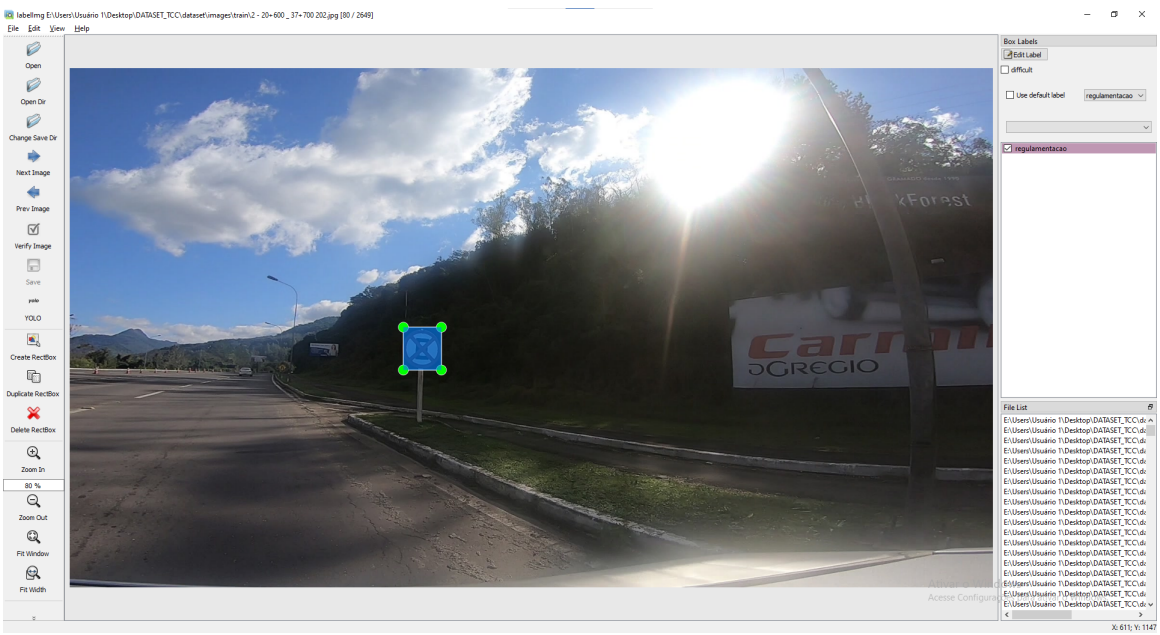
O intuito de se montar um *dataset* próprio é devido o único disponível possuir poucas imagens, fazendo com que fosse necessário a criação de um novo para se obter melhores resultados, ficando com um total de aproximadamente 3000 imagens para

[21] https://github.com/juanmcasillas/gopro2gpx

treinamento e validação, respeitada a proporção adequada de 80% para o treino e 20% para validação, como é recomendado pela literatura. Para a catalogação de imagens foi usado o *software LabelImg* devido sua saída ser no formato de entrada para o YOLOv5 e ser de fácil utilização, não demandar desempenho da máquina, além de uma ferramenta gratuita.

Como primeira etapa do pré-processo de desenvolvimento, foi realizada a criação do *dataset* para treinamento. Utilizando videos de veículos percorrendo rodovias brasileiras disponibilizados no *YouTube*, devido ao *software* de catalogação dos *labels* aceitar imagens e não diretamente vídeos, foi necessário a conversão dos arquivos de formato MP4 para PNG utilizando o *script* em Python FFmpeg [22] utilizando o comando "ffmpeg -i [video.mp4] [%d.png]". Após realizar a conversão é utilizada a ferramenta LabelImg para fazer a rotulação dos objetos de interesse, como pode ser constatado na Figura 10, que mostra um exemplo de uma placa de regulamentação sendo rotulada. A ferramenta tem como saída um arquivo no formato de texto formado pelas coordenadas da localização do objeto na imagem e sua classe, sendo um identificador para o objeto, o arquivo de saída de uma catalogação, pode ser observado na Figura 11.

Figura 10 – Interface do *LabelImg*

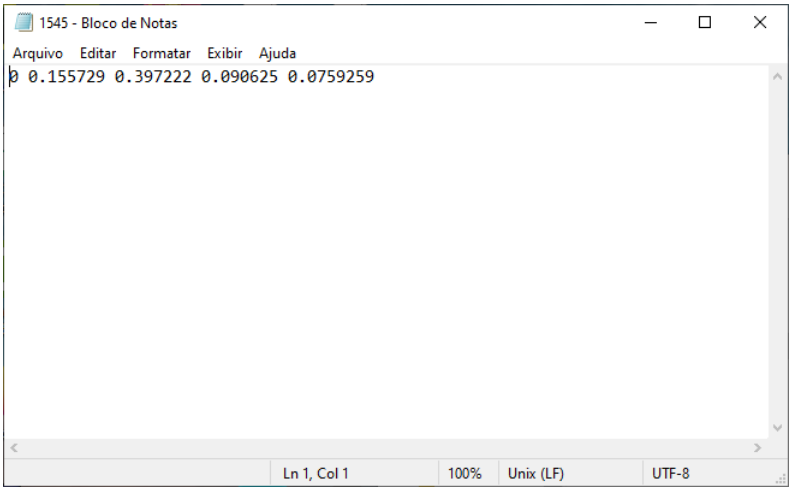


Fonte: Autor (2022)

A composição de dados do *dataset* foi dividida em três classes, sendo elas, as placas vermelhas que são de regulamentação, que podemos observar na Figura 12, são

[22] https://ffmpeg.org/

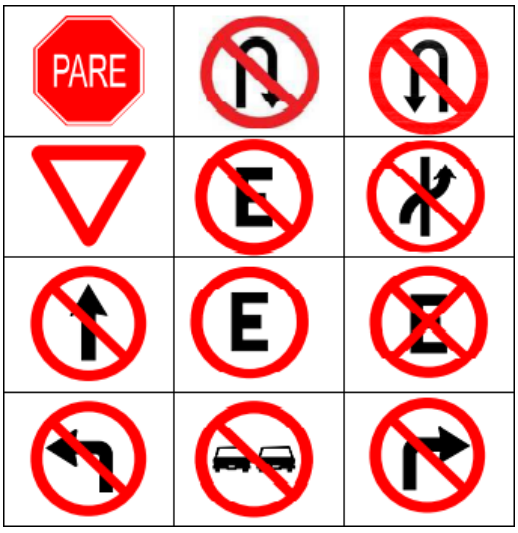
Figura 11 – Saída do arquivo do *LabelImg*



Fonte: Autor (2022)

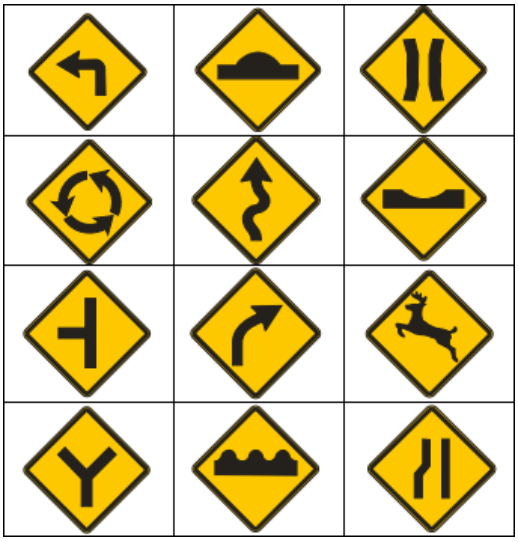
sinais impositivas que exigem atenção redobrada dos motoristas, as placas de trânsito amarelas, na Figura 13, por sua vez, são chamadas de placas de alerta, pois avisam os motoristas e pedestres condições potencialmente perigosas que podem vir pelo caminho e as azuis, vistas na Figura 14, chamadas de placas de serviços auxiliares que indicam aos usuários da via os locais onde os mesmos podem dispor dos serviços indicados, orientando sua direção ou identificando estes serviços.

Figura 12 – Exemplos de placas de trânsito brasileiras vermelhas - placas de regulamentação



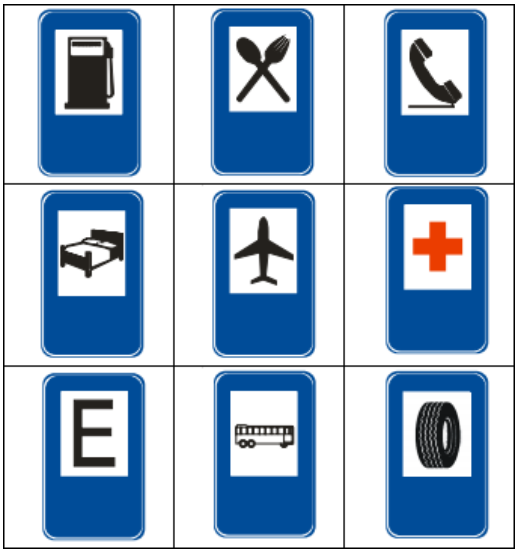
Fonte: Adaptado de Brasil (2010)

Figura 13 – Exemplos de placas de trânsito brasileiras amarelas - placas de advertência



Fonte: Adaptado de Brasil (2010)

Figura 14 – Exemplos de placas de trânsito brasileiras azuis - placas de serviços auxiliares



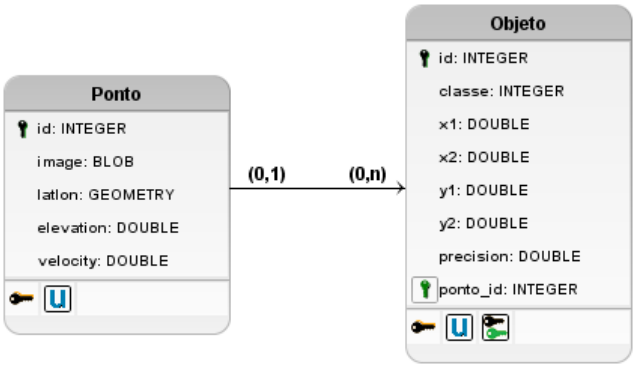
Fonte: Adaptado de Brasil (2010)

5.3 Banco de dados

Nessa sessão é contextualizado a descrição do banco de dados, suas funcionalidades, motivo de sua implementação e como se deu seu planejamento. A especificação do banco é observada em Modelo conceitual na Figura 15, cujo é dividido em duas tabelas, sendo uma dos pontos, tendo como objetivo armazenamento da imagem e dos dados extraídos dos metadados, que consiste no local de onde foi capturado o *frame* da detecção, sendo do tipo *'GEOMETRY'*, passado duas coordenadas, uma de

latitude e outra longitude formado um ponto que pode ser consultado diretamente em um sistema de informação geográfica e a imagem, que fica nessa tabela devido a posição ser atrelada ao *frame* que contém uma ou mais placas de sinalização, demais dados referentes a velocidade e elevação também são armazenados nesta tabela. A segunda tabela é composta pela classe do objeto identificado pela YOLO, sua posição na imagem, formada por quatro pontos, formando sua caixa delimitadora, a precisão da detecção e a classe do objeto. É preciso ocorrer um relacionamento de 1 para n, devido em uma imagem ser possível a ocorrência de mais de uma placa.

Figura 15 – Modelo conceitual do banco de dados



Fonte: Autor (2022)

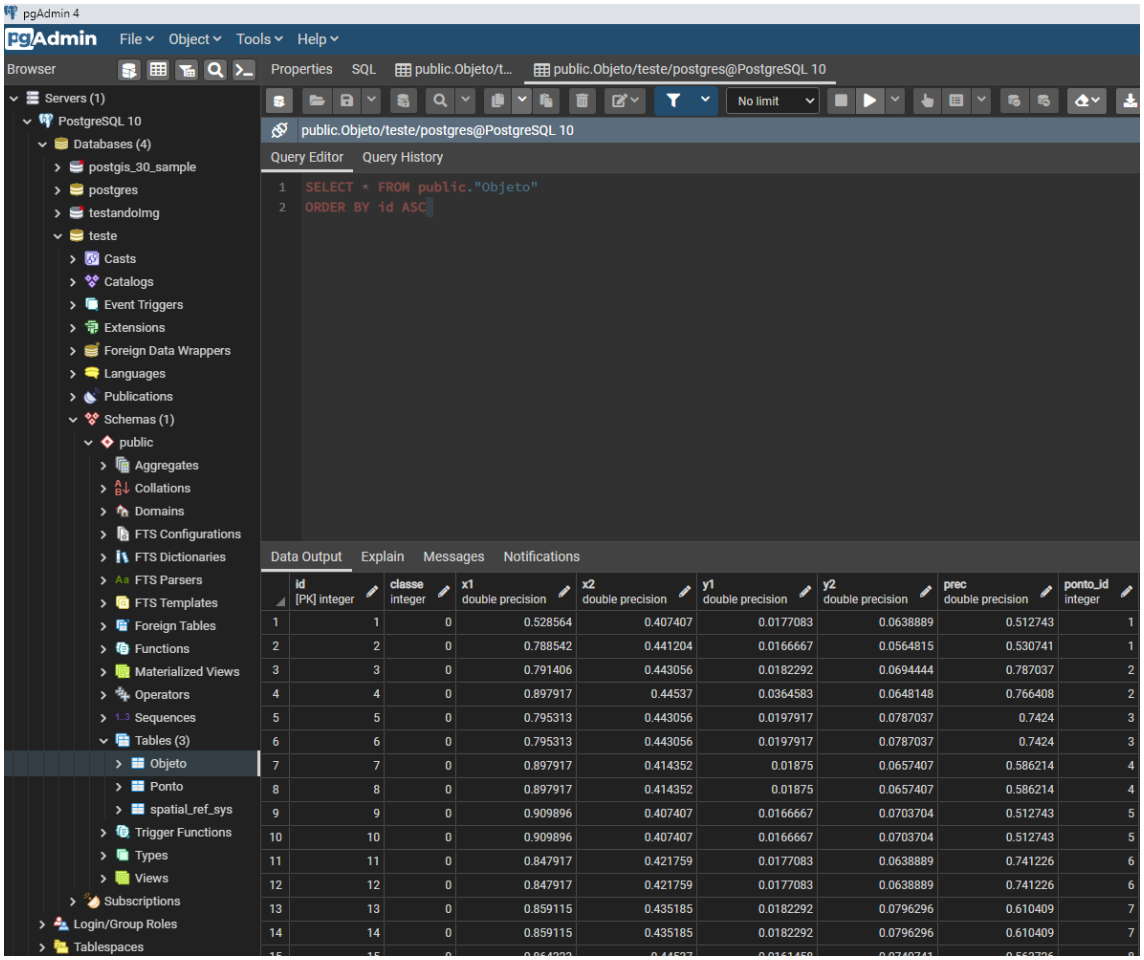
Como sistema gerenciador de banco de dados, foi utilizado o PostgreSQL[23], sua interface é observada na Figura 16, juntamente de alguns pontos inseridos, sua escolha foi devido a possibilidade de utilização de dados georreferenciados, e sua conexão com o QGIS[24], que é uma ferramenta utilizada para visualização dos dados cartográficos, na Figura 17 é possível observar pontos vermelhos onde há ocorrência de placas de trânsito, feita com base na detecção em um vídeo. Ao aproximar a imagem dos pontos, é possível visualizar cada um deles, para esses dados serem vinculados ao QGIS é feita a conexão com o banco utilizando por intermédio o PostGIS[25] que é uma extensão para dados espaciais.

Durante o desenvolvimento em *Python* foi encontrada a biblioteca SQLAlchemy[26], para utilização de ORM (*Object-Relational Mapping*), que consiste em tratar o banco de dados com programação orientada a objetos, para criação, inserção, exclusão, relacionamentos e consultas, facilitando sua implementação. Apesar de ser

[23]https://www.postgresql.org/
[24]https://qgis.org/
[25]https://postgis.net/
[26]https://www.sqlalchemy.org/

uma ferramenta completa, ela não possui compatibilidade com dados espaciais, fazendo necessário o uso de outra biblioteca, GeoAlchemy2[27] que fica responsável pela parte de georreferenciamento.

Figura 16 – Interface PostgreSQL

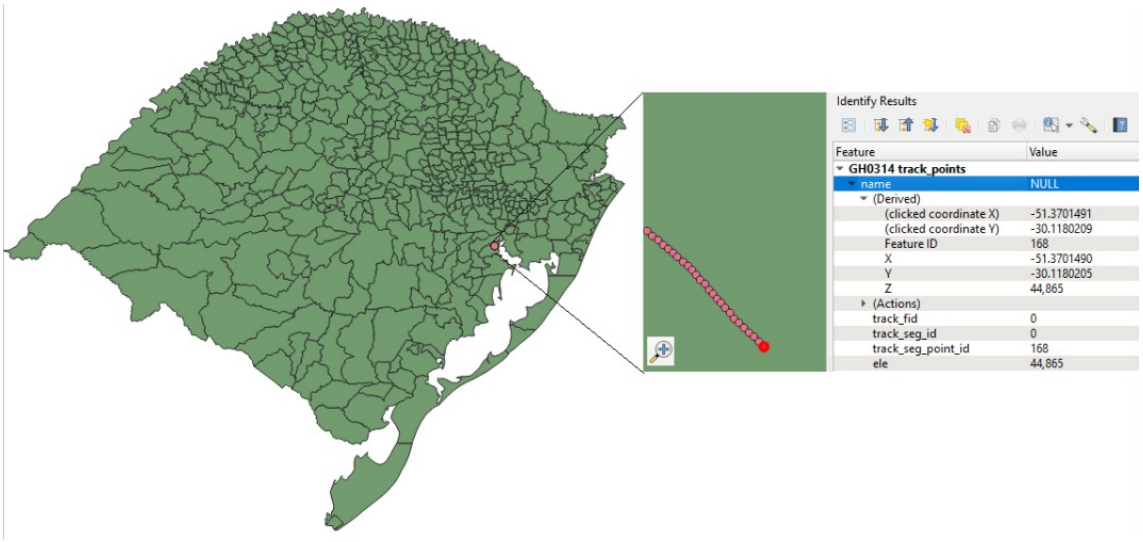


Fonte: Autor (2022)

Sua função é armazenar as imagens catalogadas pela YOLO a cada detecção feita por ela, tendo como objetivo realizar a retro alimentação para o *dataset*, fazendo com que se tenha uma base de dados vasta e variada para ser disponibilizada publicamente, fazendo assim que futuros pesquisadores possam usufruir, pois como dito anteriormente, não se localizou base de imagens catalogadas no padrão brasileiro com um tamanho significativo.

[27] https://geoalchemy-2.readthedocs.io/

Figura 17 – Visualização dos pontos no QGIS



Fonte: Autor (2022)

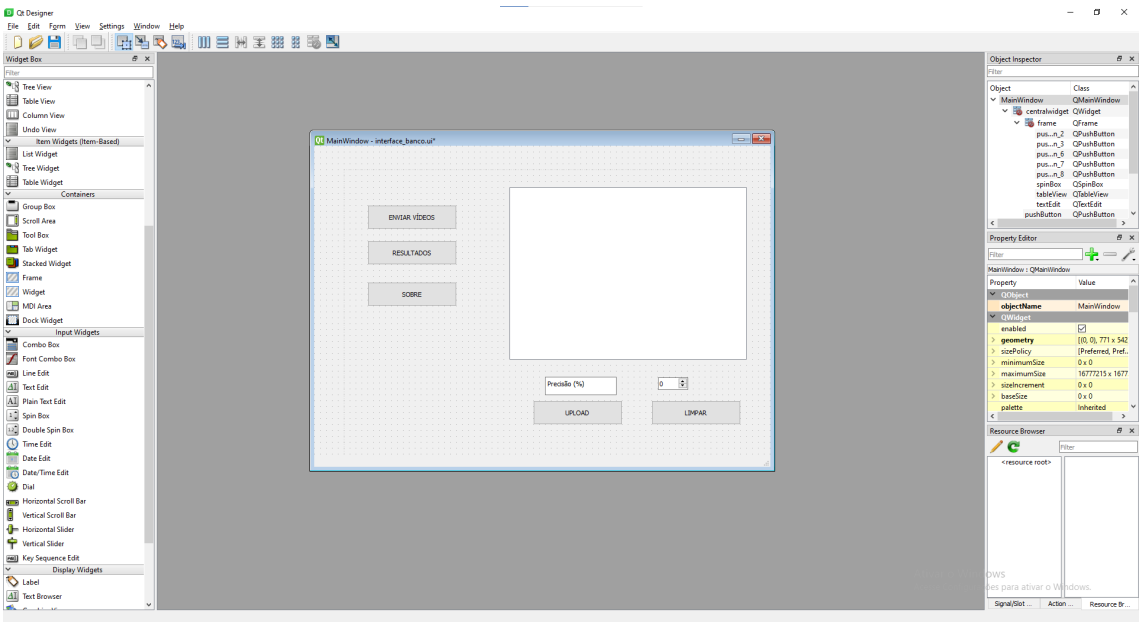
5.4 *Interface* gráfica

Para desenvolvimento da interface foi utilizada a ferramenta *QtDesigner*[28] que tem como princípio a construção de forma rápida interfaces gráficas de usuário, posicionados os botões, caixas para exibição dos dados e funcionalidades que estarão presentes para o usuário. A Figura 18 apresenta a ferramenta com o modelo de interface para detecção das imagens e inserção no banco sendo criado. Após modelada a interface, o *QtDesigner* retorna como saída um arquivo no formato UI (*User Interface*), convertido para *Python* possibilitando melhorar sua aparência, troca de cores, inserção de imagens, habilitar o uso dos botões.

Após os ajustes o resultado da interface é observado na Figura 19, as opções para o usuário são, enviar vídeo para detecção com a YOLOv5, passado como parâmetro o valor da precisão. Resultados, para salvar os resultados no banco de dados georreferenciado, para ser aberto em sistemas de informação geográfica. Após ser implementada a interface, durante os testes do seu funcionamento, foi detectado que durante a execução da rede para detecção ocorria de a interface ficar sem responder. Como solução foi necessário a implementação de *threads*, atribuída uma para executar a interface. A biblioteca escolhida para solução do problema foi PyQt5 que já estava sendo utilizada em demais partes do código, foram utilizadas as classes QThread[29], responsável pela execução da *thread* e a classe *pyqtSignal*, que tem função de sinalizar o término de sua execução.

[28]https://doc.qt.io/
[29]https://doc.qt.io/qtforpython-5/PySide2/QtCore/QThread.html

Figura 18 – Interface de detecção e inserção sendo criada com *QtDesigner*



Fonte: Autor (2022)

Após finalizada a interface para detecção e consulta, se deu início a modelagem e implementação da interface para consulta do banco, de forma que seja possível visualizar as imagens e conferir se não ocorreram falsos positivos e ser feita sua exclusão. Na Figura 20, é observado como foi realizada a modelagem dessa interface para o banco de dados. Assim como na interface de detecção, foi necessário realizar a conversão da interface da saída do *QtDesigner* para *Python*, e em sequência os ajustes de cores, funcionalidade dos botões, exibição dos dados, exclusão do dado no banco diretamente pela interface para o usuário. Na Figura 21 se observa o resultado da interface, devido haver possibilidade de ocorrência de mais de um objeto em um mesmo *frame*, foi necessário a criação de uma tabela para exibição os dados de forma dinâmica, pois a configuração de 1 ponto para n objetos permite isso.

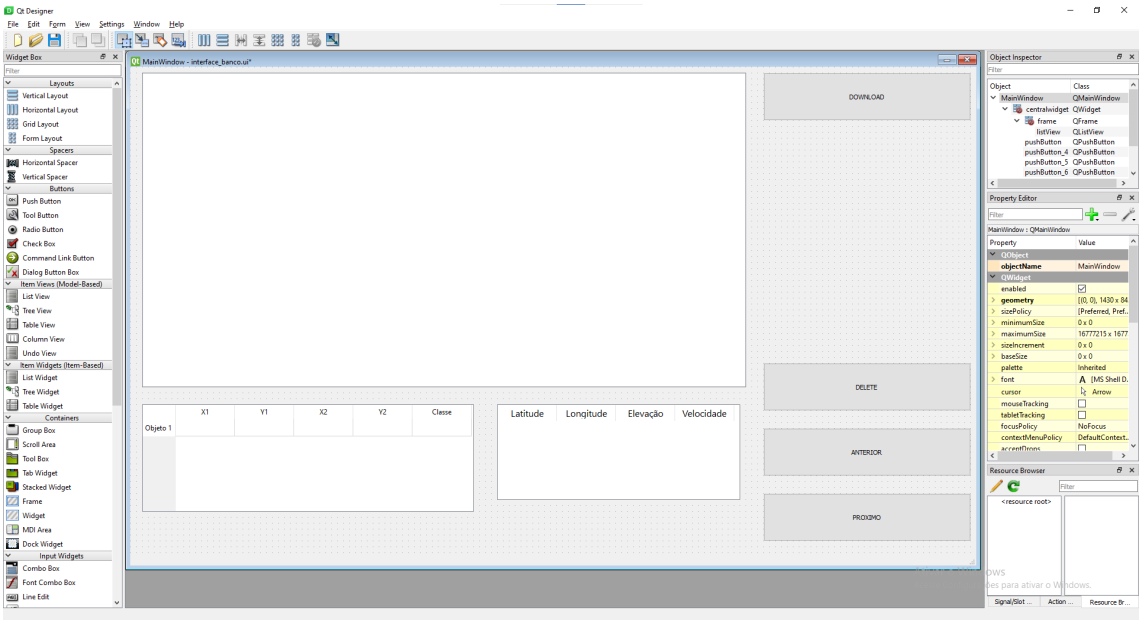
Figura 19 – Interface de detecção e inserção após refinamento

Fonte: Autor (2022)

Figura 21 – Interface de consulta finalizada

Fonte: Autor (2022)

Figura 20 – Interface de consulta sendo criada com *QtDesigner*



Fonte: Autor (2022)

5.5 Treinamento da rede

Com o *dataset* formado, os treinamentos da rede foram realizados utilizando o *Google Colab*, observado na Figura 22, a escolha foi feita devido os treinamentos demandarem uma alta capacidade de memória de vídeo. Foram realizados três treinamentos, fazendo alterações dos pesos dos parâmetros, sendo eles classificados em, *small, medium* e *extra large*.

Para realizar os treinamentos inicialmente é preciso configurar o *dataset* organizado em pastas de validação de treinamento, logo após é criado um arquivo com as informações de caminhos, número de classes e seus nomes com os comandos. Foi utilizado os comandos:

!echo "train: ./../dataset/images/train"> dataset.yaml

!echo "val: ./../dataset/images/val">> dataset.yaml

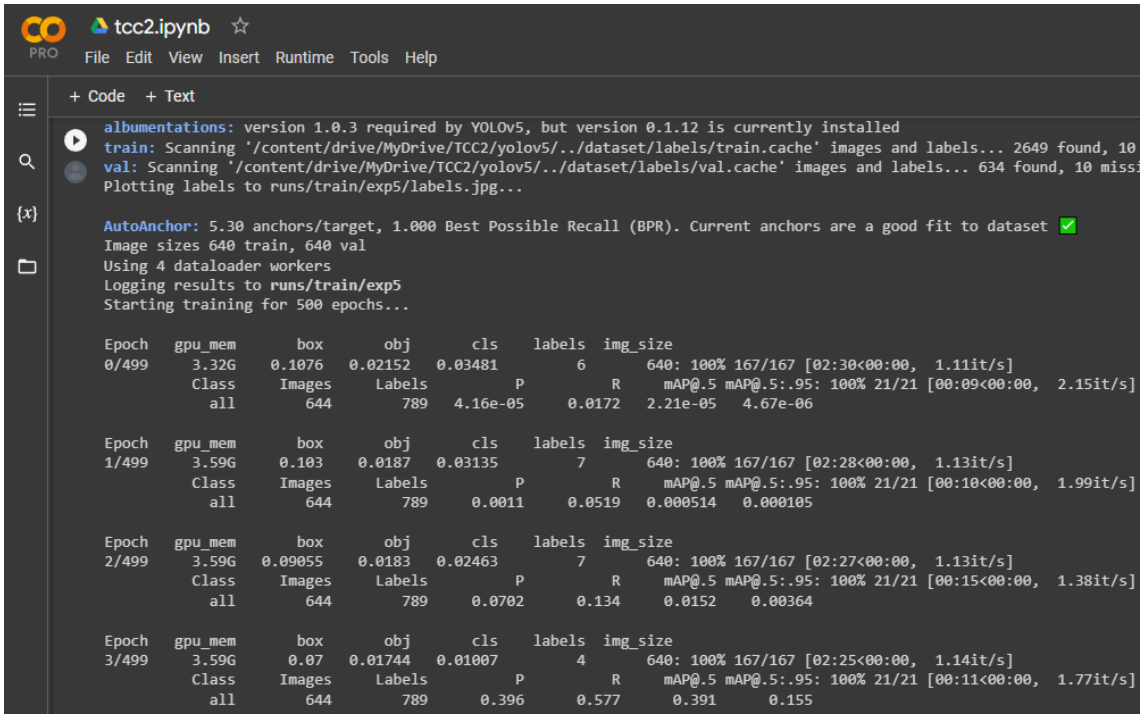
!echo "# number of classes">> dataset.yaml

!echo "nc: 3">> dataset.yaml

!echo "# class names">> dataset.yaml

!echo "names: [’nomes_classes’]">> dataset.yaml Após é rodado o comando "python train.py −−img [tamanho_imagem] −−batch [tamanho_lote] −−epochs [numero_epocas] −−data [caminho_dataset.yaml] −−cfg [caminho_pesos]"

Figura 22 – Interface *Google Colab*



Fonte: Autor (2022)

Onde foram utilizados os parâmetros, tamanho_imagem = 640, tamanho_lote = 16 e número de numero_epocas foi variado para cada peso devido durante o treinamento não ocorrer alterações nos resultados por 100 épocas ele é interrompido, e devido as redes *small* e *large* apresentarem alta recorrência de interrupção, suas épocas foram diminuídas, a GPU utilizada para os três treinamentos foi a Tesla P100-PCIE-16GB.

Tabela 4 – Tempo de treinamento

Peso	**Número de épocas**	**Tempo por época (min)**	**Tempo total (horas)**
Small	300	2.15	11.5
Large	300	2.38	12.55
Extra largee	500	5.05	41.7

Fonte: Autor (2022)

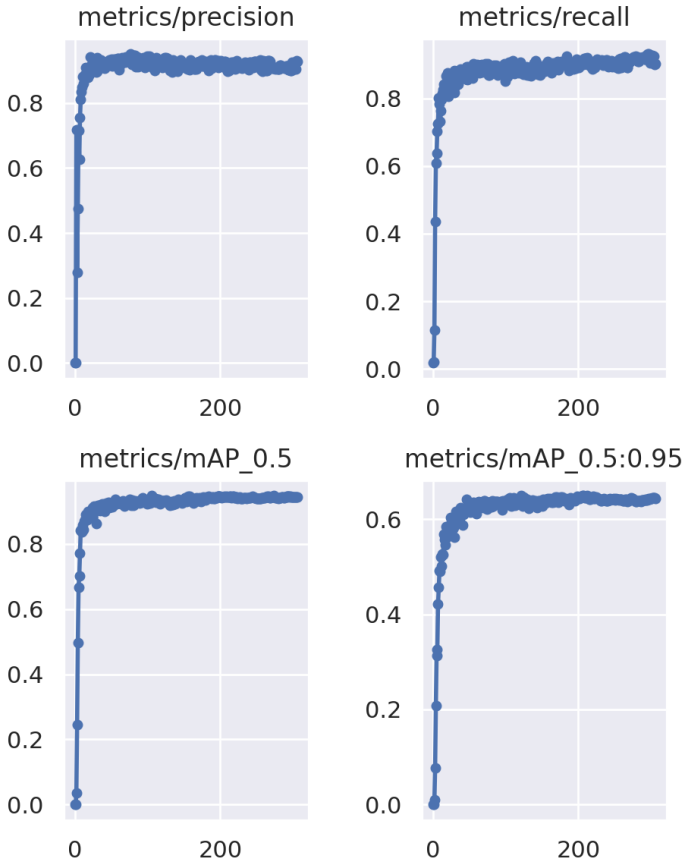
Os resultados obtidos com o treinamento das redes *small, large* e *extra large* são observados respectivamente nas Figuras 23, 24, 25. A precisão é uma métrica que apresenta a qualidade das suas detecções, ou seja, qual o percentual de suas detecções está correto. Quanto maior o valor, mais precisa a rede está. O *recall* analisa a proporção de objetos que são de fato placas, com todas as classificações realizadas e classificadas. Mede

o quão bom é encontrado todos os aspectos positivos. O mAP(*Mean Average Precision*) compara a caixa delimitadora de verdade com a caixa detectada e retorna uma pontuação. Quanto maior a pontuação, mais preciso é o modelo em suas detecções.

Todas as métricas descritas são incrementais, quanto maior o número de treinamentos e maior o *dataset*, melhor o resultado obtido, porém chega um momento onde não se é possível alcançar mais melhorias. Além disso, o método YOLO funciona de forma que a rede selecionada é a que possui os melhores resultados em uma determinada época, assim mesmo que ao final do treinamento ocorra uma piora, não é motivo de preocupação, pois a melhor será selecionada.

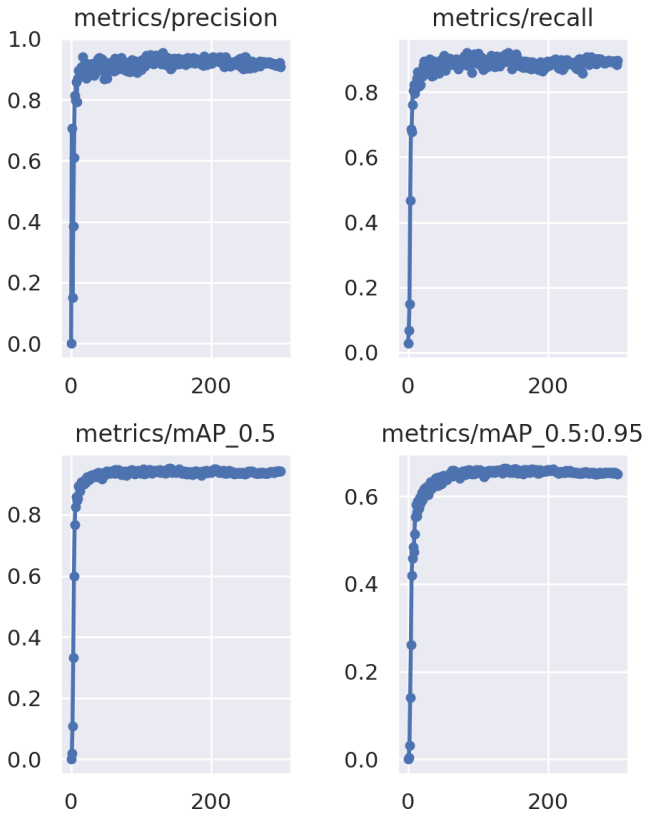
Com a melhor época de cada treinamento foi realizada a validação dos dados, utilizando a parte do *dataset* destinada para validação dos dados, com o comando "!python val.py −−img 640 −−weights [melhor_epoca_cada] −−data dataset.yaml". Os resultados da validação são vistos nas Tabelas 5, 6, 7.

Figura 23 – Resultados treinamento da rede *small*



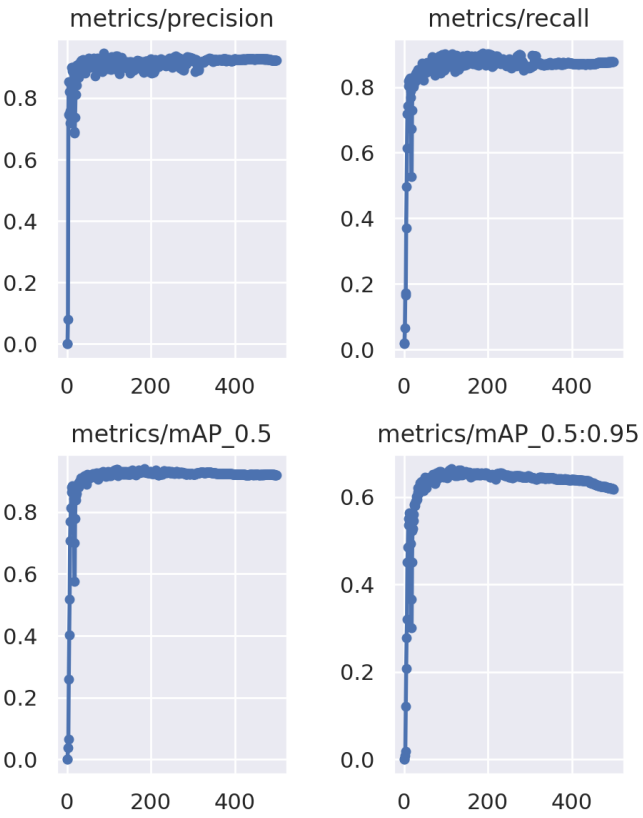
Fonte: Autor (2022)

Figura 24 – Resultados treinamento da rede *medium*



Fonte: Autor (2022)

Figura 25 – Resultados treinamento da rede *extra large*



Fonte: Autor (2022)

Tabela 5 – Validação rede *small*

Classe	*Precisão*	*Recall*	*mAP@.5*	*mAP@.5:.95: 100*
Todas	0.929	0.901	0.946	0.65
Regulamentação	0.966	0.935	0.971	0.733
Advertência	0.889	0.916	0.937	0.639
Indicação	0.931	0.852	0.93	0.579

Fonte: Autor (2022)

Tabela 6 – Validação rede *medium*

Classe	*Precisão*	*Recall*	*mAP@.5*	*mAP@.5:.95: 100*
Todas	0.926	0.912	0.952	0.664
Regulamentação	0.976	0.948	0.974	0.742
Advertência	0.891	0.92	0.935	0.679
Indicação	0.91	0.869	0.947	0.57

Fonte: Autor (2022)

Tabela 7 – Validação rede *extra large*

Classe	*Precisão*	*Recall*	*mAP@.5*	*mAP@.5:.95: 100*
Todas	0.92	0.872	0.937	0.665
Regulamentação	0.977	0.917	0.972	0.751
Advertência	0.864	0.904	0.938	0.684
Indicação	0.918	0.795	0.9	0.559

Fonte: Autor (2022)

Os resultados de precisão, ferramenta utilizada e país da placa, obtidos no levantamento dos trabalhos correlatos, são observados na Tabela 8, ordenados pela sua precisão, juntamente dos resultados obtidos utilizando as três arquiteturas propostas nessa pesquisa, a interpretação dos dados mostra que os resultados obtidos estão próximos dos trabalhos correlatos, fazendo com que o projeto utilizando placas brasileiras juntamente do YOLOv5 são válidos.

Tabela 8 – Resultados obtidos nos trabalhos correlatos

Trabalho	**Ferramenta utilizada**	**País da placas**	**Precisão (%)**
Wan *et al.* (2021)	YOLOv5	China	71.9
Dewi *et al.* (2021)	YOLOv3	Taiwan	84.9
Dewi *et al.* (2021)	YOLOv4	Taiwan	89.3
Trabalho proposto	**Extra large**	**Brasil**	**92.0**
Yang e Zhang (2020)	YOLOv3	China	92.1
Trabalho proposto	**Medium**	**Brasil**	**92.6**
Trabalho proposto	**Small**	**Brasil**	**92.9**
Laroca *et al.* (2018)	YOLOv2	-	93.5
Yang e Zhang (2020)	YOLOv4	China	94.4
Qin e Yan (2021)	YOLOv5	Nova Zelândia	98.0

Fonte: Autor (2022)

Ao utilizar o *software* em ambiente relevante, utilizando por completo, o tempo de detecção por *frame* com uma GPU GeForce GTX 1060 5Gb foram, 16ms para treinamento *small*, 24ms para *medium* e 70ms para *extra large*. Devido pequena diferença nas métricas adquiridas, a rede *small* acaba sendo vantajosa, pois além de seu treinamento ser mais rápido, sua detecção também é, fazendo assim a melhor escolha de uso durante a execução das detecções. Sendo possível sua utilização em detecções de tempo real, pois em 1 segundo é possível analisar 60 *frames*.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

No trabalho de conclusão apresentado, investigou-se o uso de redes neurais convolucionais utilizando o método YOLOv5 em placas de trânsito brasileiras, em que se buscou compreender o que se mostra sobre o fenômeno investigado, e a pergunta desta pesquisa, uma rede neural convolucional conseguiria identificar e catalogar as placas de trânsito? A presente pesquisa apresentou fortes contribuições do ponto de vista teórico sobre o tema, entregando estudos científicos sobre os métodos utilizados para identificação de objetos. E realizando a entrega de um modelo prático de resolução do problema com uso de Inteligência Artificial, com uso de redes neurais convolucionais, por um modelo básico e de simples implementação, que poderia ser utilizados em outros sistemas para identificação de placas de trânsito, além de ser entregue uma interface gráfica, facilitando sua utilização.

Fazendo com que para o presente trabalho tenha sido criado um amplo *dataset* das placas brasileiras, que se encontra disponível para demais pesquisadores via git[30], sendo o único com tamanha quantidade de dados catalogados para este objetivo, além de ser o primeiro utilizando a ferramenta YOLOv5 para este propósito se tornando um diferencial. Além de ser feito o armazenamento dos resultados em um banco de dados das placas detectadas para ser feita a ampliação do *dataset* em cada detecção e tendo conexão do banco de dados diretamente com sistema de informação geográfica, em questão o QGIS. Sendo o único projeto desenvolvido com essas ferramentas no Brasil até o momento.

Desta forma, foi possível verificar através dos estudos teóricos e técnicos computacionais que o modelo proposto através de Redes Neurais Convolucionais apresentou bons resultados, quando comparado as demais pesquisas para identificação de placas de trânsito de diferentes países. O projeto desenvolvido pode ser abordado em diferentes perspectivas, como sugestão para trabalhos futuros, a criação de uma versão dedicada para detecção de placas de trânsito brasileiras pode ser elaborada e desenvolvida de forma que consiga identificar qual é a placa, e não somente sua categoria, para isso é necessário apenas formular um *dataset* e refazer o treinamento, sendo possível aproveitar as demais partes desenvolvidas. Além do desenvolvimento de ferramentas de *tracking* que possibilite tratar o mesmo objeto como sendo único, acompanhado durante a movimentação da câmera.

[30]https://github.com/jean2612/Dataset-Placas-Transito

REFERÊNCIAS

ALPAYDIN, E. **Introduction to Machine Learning**. MIT Press, 2020. (Adaptive Computation and Machine Learning series). ISBN 9780262043793. Disponível em: https://books.google.com.br/books?id=tZnSDwAAQBAJ.

BARELLI, F. **Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV**. Casa do Código, 2018. ISBN 9788594188588. Disponível em: https://books.google.com.br/books?id=CA5ZDwAAQBAJ.

BARROS, J. G. R. D. Um estudo sobre redes neurais fractais. 2021.

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

BRAGA, A. de P.; CARVALHO, A. de L. F.; LUDERMIR, T. **Redes neurais artificiais: teoria e aplicações**. Livros Técnicos e Científicos, 2000. ISBN 9788521612186. Disponível em: https://books.google.com.br/books?id=cUgEaAEACAAJ.

BRASIL. **Manual de sinalização rodoviária**. 3. ed. Rio de Janeiro, 2010. Disponível em: <https://www.gov.br/dnit/pt-br/rodovias/operacoes-rodoviarias/faixa-de-dominio/regulamentacao-atual/743_manualsinalizacaorodoviaria-30-04-2021.pd.>

CIREŞAN, D. *et al.* A committee of neural networks for traffic sign classification. In: IEEE. **The 2011 international joint conference on neural networks**. [*S.l.*], 2011. p. 1918–1921.

DENG, J. *et al.* Imagenet: A large-scale hierarchical image database. In: IEEE. **2009 IEEE conference on computer vision and pattern recognition**. [*S.l.*], 2009. p. 248–255.

DEWI, C. *et al.* Yolo v4 for advanced traffic sign recognition with synthetic training data generated by various gan. **IEEE Access**, IEEE, v. 9, p. 97228–97242, 2021.

DU, K. *et al.* Sar atr based on displacement-and rotation-insensitive cnn. **Remote Sensing Letters**, Taylor & Francis, v. 7, n. 9, p. 895–904, 2016.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. O'Reilly Media, 2019. ISBN 9781492032618. Disponível em: https://books.google.com.br/books?id=HHetDwAAQBAJ.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. (Adaptive Computation and Machine Learning series). ISBN 9780262035613. Disponível em: https://books.google.com.br/books?id=Np9SDQAAQBAJ.

HAYKIN, S. **Neural Networks. A Comprehensive Foundation**. 2. ed. [*S.l.*]: Prentice Hall, 1998. ISBN 9780132733502,0132733501.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. 2. ed. Artmed, 2007. ISBN 9788577800865. Disponível em: https://books.google.com.br/books?id=bhMwDwAAQBAJ.

HOUBEN, S. *et al.* Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In: **The 2013 International Joint Conference on Neural Networks (IJCNN)**. [*S.l.: s.n.*], 2013. p. 1–8.

HUANG, R.; PEDOEEM, J.; CHEN, C. Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers. In: IEEE. **2018 IEEE International Conference on Big Data (Big Data)**. [*S.l.*], 2018. p. 2503–2510.

KELLEHER, J. **Deep Learning**. MIT Press, 2019. (MIT Press Essential Knowledge series). ISBN 9780262537551. Disponível em: https://books.google.com.br/books?id=b06qDwAAQBAJ.

LAROCA, R. *et al.* A robust real-time automatic license plate recognition based on the yolo detector. In: **2018 International Joint Conference on Neural Networks (IJCNN)**. [*S.l.: s.n.*], 2018. p. 1–10.

LECUN, Y.; BENGIO, Y. Convolutional networks for images, speech, and time series. **The handbook of brain theory and neural networks**, v. 3361, n. 10, p. 1995, 1995.

LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. Convolutional networks and applications in vision. In: IEEE. **Proceedings of 2010 IEEE international symposium on circuits and systems**. [*S.l.*], 2010. p. 253–256.

LUGER, G. **Inteligência Artificial**. PEARSON BRASIL, 2013. ISBN 9788581435503. Disponível em: https://books.google.com.br/books?id=UNEKvQEACAAJ.

MICHELUCCI, U. **Advanced applied deep learning: convolutional neural networks and object detection**. [*S.l.*]: Springer, 2019.

NEIVA, F. W.; SILVA, R. d. S. da. **Revisao sistemática da literatura em ciência da computaçao um guia prático**. [*S.l.*], 2016.

PRADONOV, C. C.; FREITAS, E. C. de. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico - 2ª Edição**. Editora Feevale, 2013. ISBN 9788577171583. Disponível em: https://books.google.com.br/books?id=zUDsAQAAQBAJ.

QIN, Z.; YAN, W. Q. Traffic-sign recognition using deep learning. In: SPRINGER. **Geometry and Vision: First International Symposium, ISGV 2021, Auckland, New Zealand, January 28-29, 2021, Revised Selected Papers 1**. [*S.l.*], 2021. p. 13–25.

REDMON, J. *et al.* You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S.l.: s.n.*], 2016.

REDMON, J.; FARHADI, A. Yolo9000: better, faster, stronger. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2017. p. 7263–7271.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. **arXiv preprint arXiv:1804.02767**, 2018.

SHARMA, P. *et al.* Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In: **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. [*S.l.: s.n.*], 2018. p. 2556–2565.

SOMMERVILLE, I. **Software Engineering**. Pearson Education, 2011. ISBN 9780133001495. Disponível em: https://books.google.com.br/books?id=fSYrAAAAQBAJ.

SZEGEDY, C.; TOSHEV, A.; ERHAN, D. Deep neural networks for object detection. 2013.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: SN. **Proceedings of the XXIX conference on graphics, patterns and images**. [*S.l.*], 2016. v. 1, n. 4.

WAN, H. *et al.* A novel neural network model for traffic sign detection and recognition under extreme conditions. **Journal of Sensors**, Hindawi, v. 2021, 2021.

WEN, C. *et al.* Spatial-related traffic sign inspection for inventory purposes using mobile laser scanning data. **IEEE Transactions on Intelligent Transportation Systems**, v. 17, n. 1, p. 27–37, 2016.

YANG, W.; ZHANG, W. Real-time traffic signs detection based on yolo network model. In: **2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)**. [*S.l.: s.n.*], 2020. p. 354–357.