UNIVERSIDADE REGIONAL INTEGRADA DO ALTO URUGUAI E DAS MISSÕES
CAMPUS DE ERECHIM
DEPARTAMENTO DE ENGENHARIAS E CIÊNCIA DA COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

CAROLINE PAULA KOLASSA

TREINAMENTO DE REDE NEURAL COM YOLOV8 PARA RECONHECIMENTO DE PLACAS DE TRÂNSITO

ERECHIM - RS
2023

CAROLINE PAULA KOLASSA

TREINAMENTO DE REDE NEURAL COM YOLOV8 PARA RECONHECIMENTO DE PLACAS DE TRÂNSITO

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel, Departamento de Engenharias e Ciência da Computação da Universidade Regional Integrada do Alto Uruguai e das Missões Campus de Erechim.

Orientador: Prof. MSc. Daniel Menin Tortelli

ERECHIM - RS
2023

AGRADECIMENTOS

Agradeço imensamente a minha mãe Dirce e ao meu irmão Walter que não mediram esforços para me manter estudando, sempre oferecendo apoio, incentivo e amor. Obrigada também ao meu namorado David Luis por estar comigo nesse momento desafiador e entender minha ausência. Agradeço aos meus amigos, Pedro e Maicon, por trazerem alegria às aulas, assim como aos demais colegas que tornaram o processo mais leve.

Um agradecimento especial aos meus amigos e colegas de trabalho, tanto do setor de suporte, quanto do setor de desenvolvimento e elétrica, que compartilharam conhecimento e mostraram-se prestativos durante toda a graduação. Um obrigada especial a minha vó Maria e a toda minha família que me quer bem e que torce pelo meu sucesso. Mesmo que alguns não estejam mais fisicamente presentes, vocês estão eternamente em meu coração.

Agradeço a meu orientador Daniel e a todos os professores que me ajudaram a formar o conhecimento e resistência que tenho hoje, vocês são pilares fundamentais na minha vida. Gratidão a Deus e a minha psicológa por me fornecerem tanta força e também agradeço a mim mesma por não ter desistido diante de tantos desafios.

Por fim, minha gratidão estende-se a todos que de alguma forma influenciaram e apoiaram minha jornada acadêmica. A finalização deste trabalho de conclusão é a realização de um sonho e finalizo esta etapa com muita alegria, entusiamo e conhecimento.

Se você está no caminho errado, voltar atrás significa progresso.

(C. S. Lewis)

RESUMO

Com o crescimento constante da frota de veículos em todo o mundo, a segurança no trânsito tem se tornado uma preocupação cada vez mais recorrente. Dentre as causas de acidentes nesse meio, destaca-se a falta de atenção dos motoristas às sinalizações nas vias públicas, que podem levar a graves acidentes. Neste contexto, o presente trabalho propõe-se a treinar uma rede neural com o algoritmo de detecção de objetos *YOLO* na versão 8 para identificação de 15 placas de regulamentação e 2 de advertência. Para a rotulagem das imagens foi utilizado o *framework Roboflow* e para o treinamento do modelo a linguagem de programação *Python*. O treinamento e estudo realizado acerca do tema contribuiu para o crescimento do conhecimento da área que é tão importante para o desenvolvimento de tecnologias que podem salvar vidas no meio viário. A precisão média da rede treinada alcançou 69,9%, o mAP atingiu 65,3%, e o Recall ficou em 58,8%, cumprindo o objetivo de identificação proposto.

Palavras-chave: Rede neural. YOLO. Segurança no trânsito. Inteligência Artificial. Visão Computacional.

ABSTRACT

With the constant growth of the vehicle fleet worldwide, traffic safety has become an increasingly recurring concern. Among the causes of accidents in this field, the lack of attention of drivers to the signs on public roads stands out, which can lead to serious accidents. In this context, this work proposes to train a neural network with the object detection algorithm YOLO in version 8 to identify 15 regulatory and 2 warning signs. The Roboflow framework was used for image labeling and the Python programming language for model training. The training and study carried out on the subject contributes to the growth of knowledge in an area that is so important for the development of technologies that can save lives on the road. The average result of the trained network reached 69.9%, the mAP reached 65.3%, and the Recall was 58.8%, fulfilling the proposed identification objective.

Keywords: Neural network. YOLO. Road safety. Artificial intelligence. Computer vision.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

YOLO	*You Only Look Once*
IBGE	*Instituto Brasileiro de Geografia e Estatística*
IA	*Inteligência Artificial*
RNA	*Rede neural artificial*
MLP	*Multi Layer Perceptron*
CNN	*Rede neural convolucional*
RNCs	*Redes Neurais Convolucionais*
COCO	*Common Objects in Context*
DPM	*Deformable Part Models*
R-CNN	*Regions with Convolutional Neural Network*
CTB	*Código de Trânsito Brasileiro*
CONTRAN	*Conselho Nacional de Trânsito*
DARPA	*Defense Advanced Research Projects Agency*
USA	*United States of America*
SAE	*Society of Automotive Engineers*
TCC	*Trabalho de Conclusão de Curso*
IDE	*Integrated Development Environment*
OpenCV	*Open Source Computer Vision Library*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

No Brasil, uma pesquisa do Ministério dos Transportes, Portos e Aviação, apontou que mais de 50% dos acidentes de trânsito são causados por falhas humanas em rodovias federais. Dessa contagem, 30,3% é fruto do desrespeito às leis de tráfego e 23,4% falta de atenção do condutor. (TRANSPORTES, 2018). Além disso, segundo dados do IBGE, a frota de veículos continua crescendo sendo em sua maioria os automóveis, portanto, a segurança no trânsito precisa ser pautada e discutida, já que atinge direta ou indiretamente a vida de todos os cidadãos. (IBGE, 2022).

Diante desse cenário, a tecnologia de Visão Computacional tem sido amplamente explorada como uma ferramenta de apoio à segurança viária. Aliado a isso, as Redes Neurais Convolucionais também agregam a esta solução, pois automatizam todas as fases da detecção (seleção de região, extração de características e classificação) e de reconhecimento de objetos. (SÁ, 2021).

O *YOLO* é um método de detecção de objetos focado em processamento em tempo real que, diferente dos sistemas de detecção tradicionais, emprega uma rede neural convolucional na imagem completa. Essa rede profunda divide a imagem em regiões e estima seus limites e probabilidades para cada região, as quais são avaliadas e suas probabilidades calculadas. Por isso, consegue identificar múltiplos objetos e manter a performance, ideal para ser empregado em um ambiente que demanda identificação rápida e precisa. (ZHAO, 2019).

No entanto, quando trata-se de automatizar veículos para auxiliar na segurança viária, muitas são as questões que permeiam o assunto, desde a legislação, questões éticas e o desempenho da própria tecnologia utilizada. Baseado nisso, o presente trabalho propõem um estudo e treinamento de uma rede neural com o algoritmo de detecção de objeto *YOLO* em sua oitava versão para reconhecimento de algumas placas de trânsito visando documentar e validar se a rede treinada seria capaz de desempenhar uma boa performance em um cenário simulado.

O restante do trabalho está organizado da seguinte forma: capítulo 2 apresenta a Revisão Bibliográfica sobre assuntos relevantes para o tema. No capítulo 3 é realizado um estudo sobre trabalhos relacionados com o tema aqui proposto. O capítulo 4 apresenta todas as ferramentas utilizadas no desenvolvimento e o capítulo 5 descreve o *dataset* utilizado, bem como o processo de rotulagem. O capítulo 6 descreve os treinamentos da rede neural e o capítulo 7 exibe os resultados obtidos. Por fim, o capítulo 8 apresenta as conclusões e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo é dedicado à escrita da fundamentação teórica, onde são descritos conceitos importantes sobre Visão Computacional, Inteligência Artificial e Redes Neurais. Além disso, são apresentadas informações sobre o algoritmo *YOLO* e placas de trânsito brasileiras, bem como uma breve explicação sobre veículos autônomos.

2.1 Visão Computacional

A Visão Computacional teve sua origem concreta nos anos 1970, inicialmente concebida como uma maneira de emular a inteligência humana e conferir comportamento inteligente aos robôs. Entretanto, à medida que o tempo avançou, ficou claro que o processo não era tão simplista como sugerido na época e o pensamento de "conectar uma câmera a um computador e fazer com que o computador descreva o que vê", não era fácil. Em vez disso, a Visão Computacional exigia uma compreensão profunda da visão humana e da percepção, juntamente com o desenvolvimento de algoritmos capazes de extrair significado a partir de imagens. (BODEN, 1988).

> A Visão Computacional é a construção de descrições explícitas e significativas de objetos físicos a partir de imagens. A compreensão da imagem é muito diferente do Processamento da Imagens, que estuda transformações de imagem a imagem e não a descrição explícita da construção. As descrições são um pré-requisito para o reconhecimento, manipulação e o pensamento sobre os objetos. (BALLARD; BROWN, 1982)

Um dos desafios centrais da Visão Computacional reside na natureza inversa do problema, no qual se busca recuperar incógnitas a partir de informações insuficientes para especificar completamente a solução. Consequentemente, a disciplina recorre a modelos fundamentados em princípios físicos e probabilísticos, bem como ao aprendizado de máquina com base em conjuntos extensos de exemplos, a fim de discernir entre soluções potenciais. No entanto, modelar o mundo visual em toda a sua complexidade e riqueza se revela consideravelmente desafiador. (SZELISKI, 2021).

Por vezes, os seres humanos percebem a estrutura tridimensional do mundo com certa facilidade e naturalidade, no entanto, é essencial reconhecer o quanto as imagens são ricas em termos de dados e como sua manipulação pode conduzir à geração de soluções e à simplificação de processos em computação e robótica. (SÁ, 2021).

Atualmente, a Visão Computacional está sendo amplamente utilizada em uma variedade de aplicações do mundo real, que incluem:

a) Reconhecimento óptico de caracteres (OCR): pode ser utilizado na leitura de códigos postais manuscritos e no reconhecimento automático de placas de veículos;

b) Inspeção de máquinas: inspeção rápida de peças para garantia de qualidade;

c) Logística de armazém: para entrega autônoma de pacotes e "drives"que transportam paletes e coleta de peças por manipuladores robóticos;

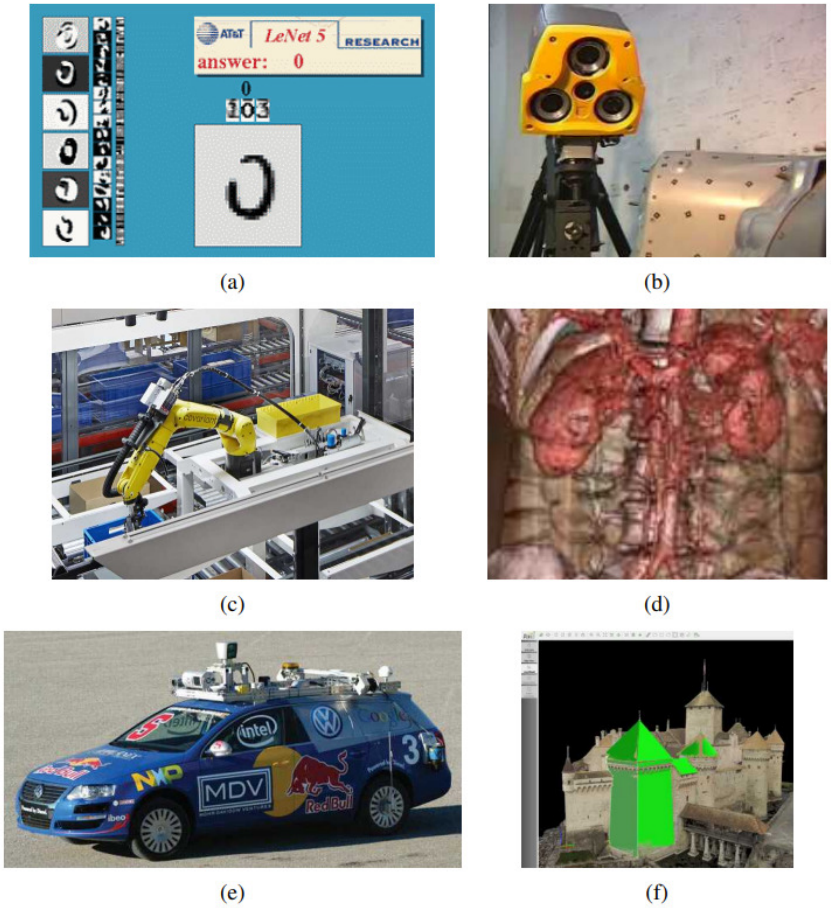
d) Imagens médicas: é possível automatizar o controle de qualidade de uma produção e em áreas mais específicas pode ser útil na caracterização e classificação de minérios;

e) Veículos autônomos: capazes de dirigir de um ponto a outro entre cidades, bem como voo autônomo;

f) Construção de modelos 3D: construção totalmente automatizada de modelos 3D a partir de fotografias aéreas e de drones.

A Figura 1 exibe os exemplos citados para melhor entendimento:

Figura 1 – Exemplos de utilização de Visão Computacional.



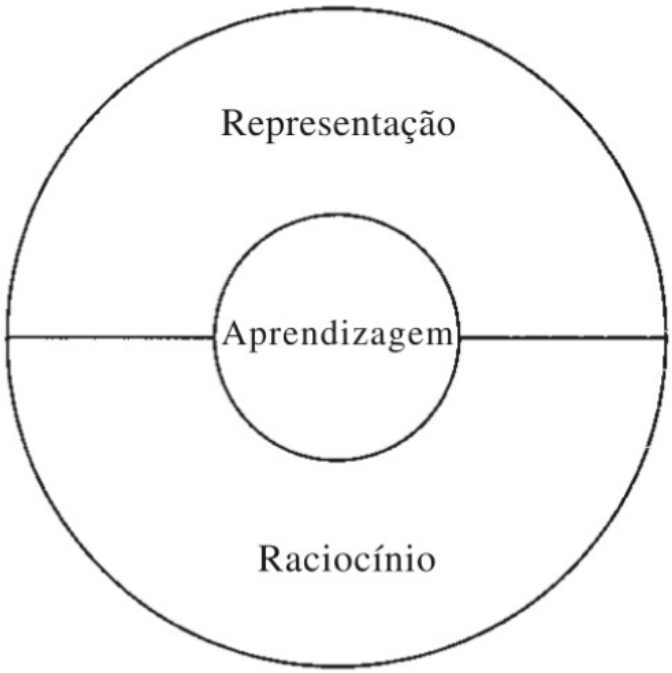
Fonte: Szeliski (2021)

2.2 Inteligência Artificial

O objetivo da Inteligência Artificial é o desenvolvimento de paradigmas ou algoritmos que requeiram máquinas para realizar tarefas cognitivas, para as quais os humanos são atualmente melhores. Ela deve ser capacitada para realizar três tarefas: armazenar conhecimento, utilizar conhecimento armazenado para solucionar problemas e adquirir novos conhecimentos

através da experiência. Sendo assim, representação, raciocínio e aprendizagem, como apresentado na Figura 2 (SAGE, 1990).

Figura 2 – Ilustração dos três componentes principais de um sistema de IA



Fonte: SAGE (1990)

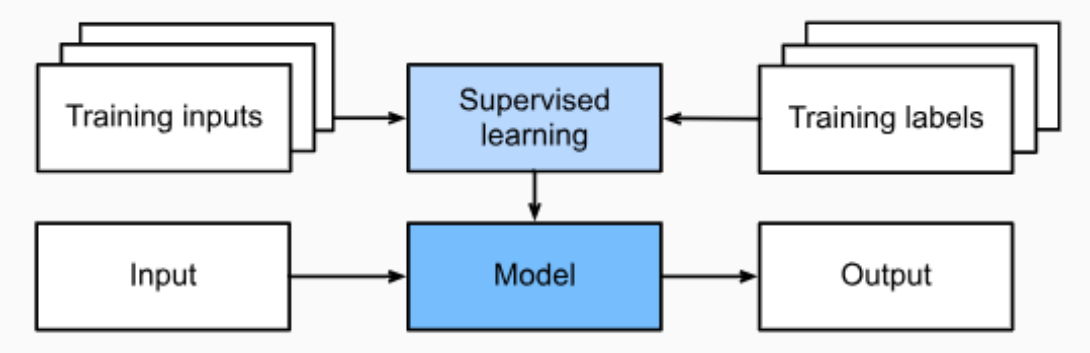
2.3 Machine Learning

Em 1959, Arthur Samuel, cientista da computação, definiu aprendizado de máquina como: "O campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados". (AGGARWAL, 2019). As tarefas de aprendizado de máquina são majoritariamente classificadas em três categorias amplas:

1. **Aprendizado supervisionado:** O aprendizado supervisionado aborda a tarefa de prever rótulos com recursos de entrada. Esse processo envolve a coleta de dados, o treinamento de um modelo e o uso dele para fazer absorção com entradas não vistas. São apresentadas ao computador exemplos de entradas e saídas desejadas e o objetivo é produzir um modelo que mapeia qualquer entrada para uma previsão de *label*. Este tipo de aprendizagem pode ser aplicado em vários ramos, como a área da saúde, onde é possível, por exemplo, prever um ataque cardíaco em um paciente com base em sua frequência cardíaca e pressão arterial. (ZHANG et al., 2020). A Figura 3 descreve o fluxo do processo.
   Outros exemplos relacionados são os sistemas de recomendação e classificação onde o objetivo é exibir itens relevantes para os usuários, mas com ênfase na personalização. A

busca, por outro lado, lida com a ordenação de resultados relevantes para os usuários, como em pesquisa da internet, onde a ordem importa. Cada um desses campos utiliza algoritmos e técnicas específicas de aprendizagem supervisionada para resolver problemas diferentes. (ZHANG et al., 2020).

Figura 3 – Fluxo do Aprendizado supervisionado



Fonte: Zhang et al. (2020)

2. **Aprendizado não supervisionado:** No Aprendizado não supervisionado nenhum tipo de etiqueta é entregue ao algoritmo de aprendizado, deixando-o sozinho para encontrar estrutura nas entradas fornecidas. O aprendizado não supervisionado descobre novos padrões nos dados ou um meio para atingir um fim. Esse modelo é utilizados para três tarefas principais, sendo elas armazenamento em *cluster*, associação e redução de dimensionalidade. (IBM, 2023b).

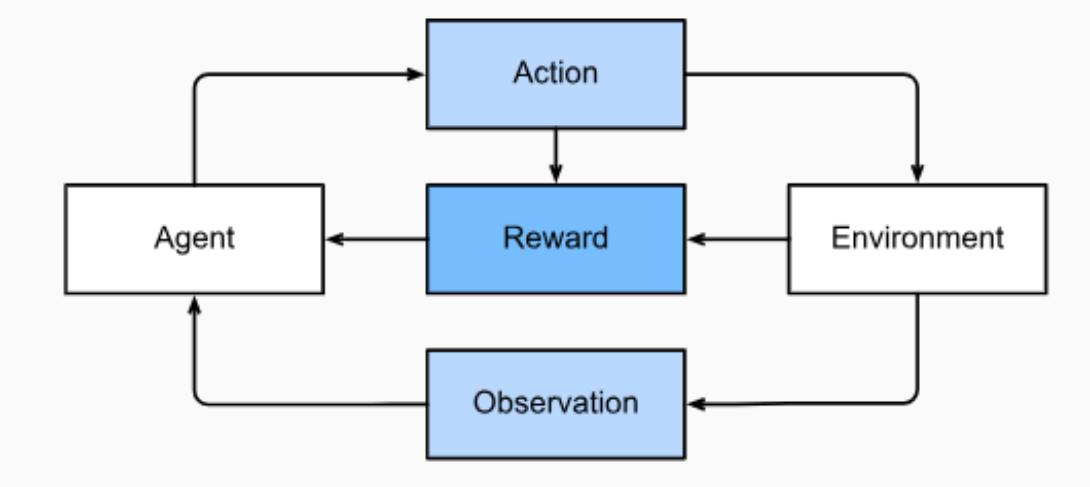
   O armazenamento em *cluster* é uma abordagem da mineração de dados que visa agrupar dados não rotulados com base em suas semelhanças ou diferenças. Algoritmos de clusterização são empregados para organizar objetos de dados não classificados e em sua forma bruta em grupos que refletem estruturas ou padrões nas informações. Esses algoritmos podem ser classificados em diferentes tipos, incluindo exclusivos, sobrepostos, hierárquicos e probabilísticos. (IBM, 2023b).

   Uma regra de associação é uma abordagem baseada em regras para identificar conexões entre variáveis em um conjunto de dados específico. Esses métodos são frequentemente empregados na análise de cestas de compras, o que permite às empresas obter um entendimento mais profundo das relações entre diversos produtos. Embora existam diferentes algoritmos para gerar regras de associação, como , *Eclat* e *FP-Growth*, o algoritmo *Apriori* é amplamente reconhecido como o mais utilizado. (IBM, 2023b).

   A redução de dimensionalidade é uma técnica empregada quando o número de características ou dimensões em um conjunto de dados é significativamente elevado. Seu propósito é reduzir o número de entradas de dados para um nível gerenciável, ao mesmo tempo em que tenta preservar a integridade do conjunto de dados ao máximo possível. (IBM, 2023b).

3. **Aprendizado por reforço:** A Aprendizagem por Reforço é uma técnica de aprendizado de máquina que combina o aprendizado profundo com a aprendizagem por reforço. Ela tem sido usada em jogos como o xadrez e o Go, onde a rede *Q-Network* foi capaz de derrotar jogadores humanos usando apenas a entrada visual. Outros exemplo é a *deep Q-netfowk* que derrotou os humanos nos jogos da Atari usando apenas a entrada visual. Esse aprendizado fornece uma declaração geral de um problema em que um agente interage com um ambiente ao longo de uma série de etapas. Em cada etapa o agente recebe alguma observação do ambiente e deve escolher uma ação que é posteriormente transmitida de volta para o ambiente por meio de algum mecanismo. Por fim, o agente recebe uma recompensa do meio ambiente. Este processo é ilustrado na Figura 4. (ZHANG et al., 2020).

Figura 4 – Aprendizado por reforço



Fonte: Zhang et al. (2020)

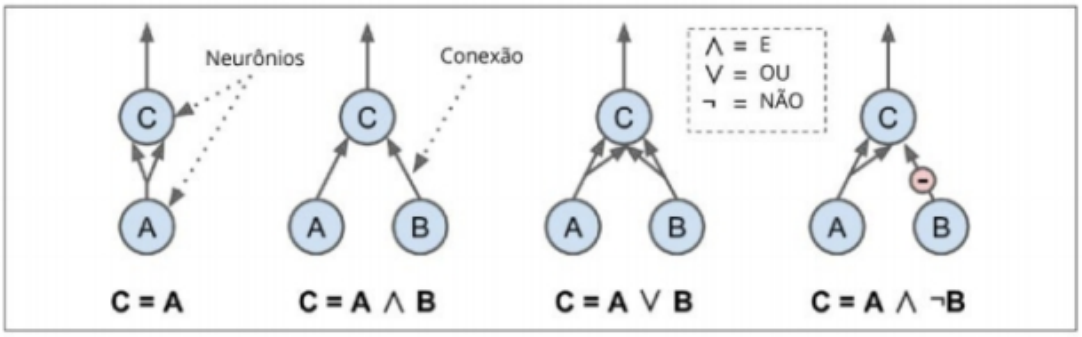
2.4 Redes Neurais Artificiais

O primeiro modelo neural foi proposto por dois pesquisadores, o neurocientista Warren McCulloch e o profissional de lógica computacional Walter Pitts no ano de 1943. Juntos eles escreveram um artigo que tentava modelar matematicamente como um neurônio funcionaria. (B., 2000). Esse modelo tinha uma ou mais entradas binárias e uma saída também binária e ativava sua saída quando mais de um certo número de suas entradas estavam ativas. Assim, os pesquisadores mostraram que mesmo com um modelo bastante simples, é possível construir uma rede de neurônios artificiais que implementa qualquer tipo de operação lógica. (FERREIRA, 2021).

Na Figura 5 é possível identificar as operações pois na primeira rede à esquerda é realizada a operação de identidade: se o neurônio A for ativado, o neurônio C também será ativado (uma vez que recebe dois sinais de entrada do neurônio A); mas se o neurônio A está desativado, o neurônio C também será desativado. Na segunda rede é realiza a operação lógica

E: o neurônio C é ativado apenas quando os neurônios A e B são ativados (um único sinal de entrada não é suficiente para ativar o neurônio C). Na terceira rede realiza a operação lógica OU: o neurônio C é ativado se o neurônio A ou o neurônio B for ativado. Por fim, se é suposto que uma conexão de entrada pode inibir a atividade do neurônio, a quarta rede realiza uma operação lógica de não lógico: o neurônio C é ativado apenas se o neurônio A estiver ativo e o neurônio B estiver desativado.

Figura 5 – Redes neurais executando diferentes operações lógicas

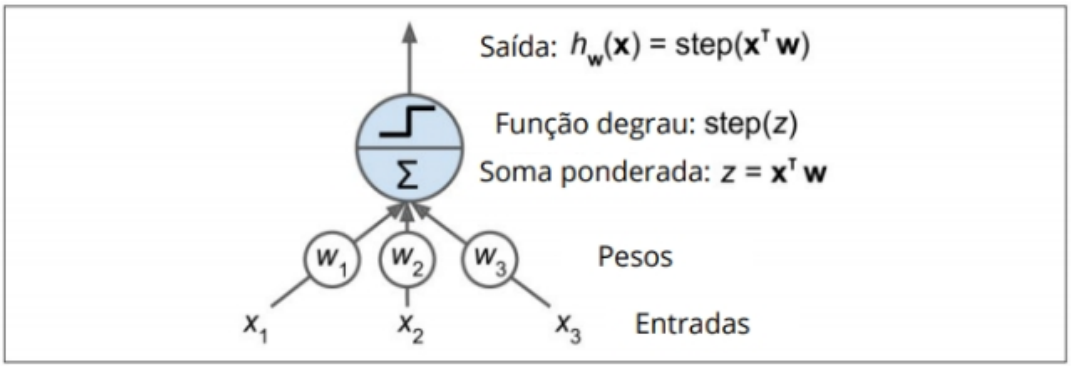


Fonte: Geron (2019)

2.4.1 Perceptron

O Perceptron é uma das arquiteturas de rede neural artificial (RNA) mais simples que existe. Ela foi inventada em 1957 por Frank Rosenblatt, e foi baseada em um neurônio artificial diferente do proposto pelo McCulloch e Pitts pois as entradas e saídas são números (em vez de valores binários) e cada conexão de entrada está associada a um peso. A Figura 6 ilustra o processo. (FERREIRA, 2021).

Figura 6 – Neurônio Artificial aplicanda a função degrau



Fonte: Geron (2019)

No entanto, uma arquitetura de redes neurais com poucos neurônios tem baixo potencial de resolução de problemas. Para realizar tarefas complexas, é necessário associar diversos

neurônios em uma arquitetura baseada em camadas.

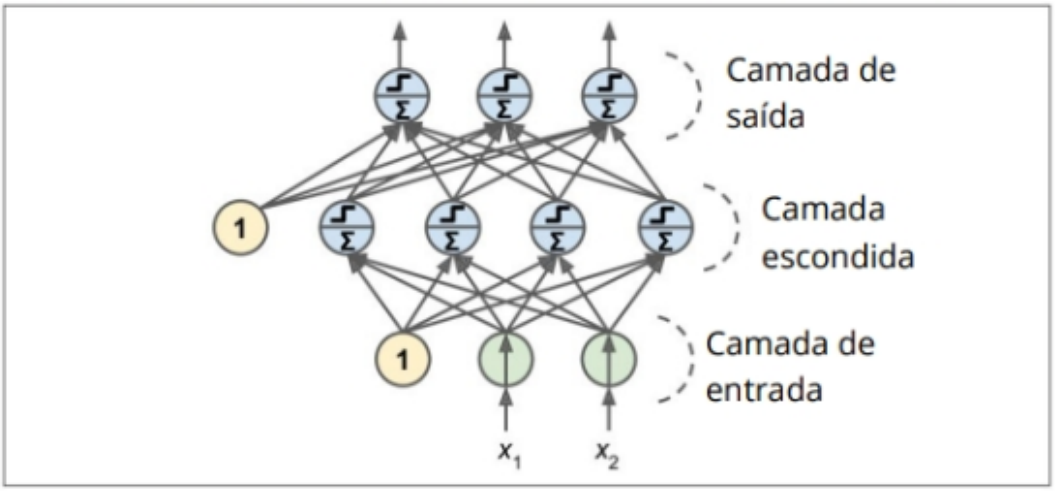
2.4.2 Algoritmo Backpropagation

Por muitos anos, os pesquisadores da área de IA trabalharam para encontrar uma maneira eficiente de treinar o perceptron multicamadas (MLP), mas não obtinham sucesso. No entanto, em 1986, David Rumelhart, Geoffrey Hinton e Ronald Williams publicaram um artigo que introduziu o algoritmo de treinamento chamado backpropagation (retro-propagação), que continua sendo usado até os dias atuais. (FERREIRA, 2021).

Para cada conjunto de dados de treinamento, a primeira etapa do algoritmo consiste em realizar uma previsão e calcular o erro associado, o que é conhecido como *feed-forward*. Em seguida, a atenção se volta para a camada imediatamente anterior, onde é calculada a contribuição de cada conexão para o erro, processo denominado *backpropagation*. Por fim, os pesos das instalações são ajustados de maneira a reduzir o valor do erro, empregando a técnica de descida do gradiente. Para que o algoritmo funcione corretamente, seus autores fizeram uma mudança fundamental na arquitetura do MLP: eles substituíram a função degrau pela função logística.

Isso foi essencial porque a função degrau contém apenas segmentos planos, assim, não há um gradiente para processar, porque a técnica da descida do gradiente não pode se mover em uma superfície plana. Por sua vez, a função logística tem uma derivada diferente de zero bem definida, permitindo que a descida do gradiente faça progressos ao longo das etapas. A Figura 7 exemplifica o processo.

Figura 7 – Arquitetura de um MLP com três camadas



Fonte: Geron (2019)

As redes neurais contam com dados de treinamento para aprender e melhorar sua precisão tornando-se ferramentas poderosas. Algumas tarefas de reconhecimento podem levar minutos em vez de horas quando comparadas com a identificação manual feita por especialistas

humanos. Uma das redes neurais mais conhecidas é o algoritmo de procura do *Google*. (IBM, 2023a).

2.4.3 Redes Neurais Convolucionais

As Redes Neurais Convolucionais, conhecidas como *Convolutional Neural Networks* (CNN), tiveram sua origem a partir do estudo do córtex visual do cérebro e têm sido empregadas no reconhecimento de imagens desde a década de 1980. Nos últimos anos, devido ao aumento significativo do poder de processamento computacional, à disponibilidade de grandes volumes de dados e ao desenvolvimento de métodos para abordar desafios relacionados ao treinamento de redes neurais profundas, as CNN alcançaram um desempenho sobre-humano em algumas tarefas visuais complexas. (FERREIRA, 2021).

Em 1959, David Hubel e Torsten Wiesel realizaram uma série de experimentos com gatos que forneceram *insights* importantes sobre a estrutura do córtex visual. Em particular, eles mostraram que muitos neurônios no córtex visual têm um pequeno campo receptor local, o que significa que eles reagem apenas a estímulos visuais localizados em uma região limitada do campo visual. Esses estudos inspiraram uma rede neural chamada *neocognitron*, introduzida em 1980, e que gradualmente evoluiu as atuais redes neurais convolucionais. Tais redes apresentam dois novos tipos de camadas: as convolucionais e as camadas *pooling* (FERREIRA, 2021).

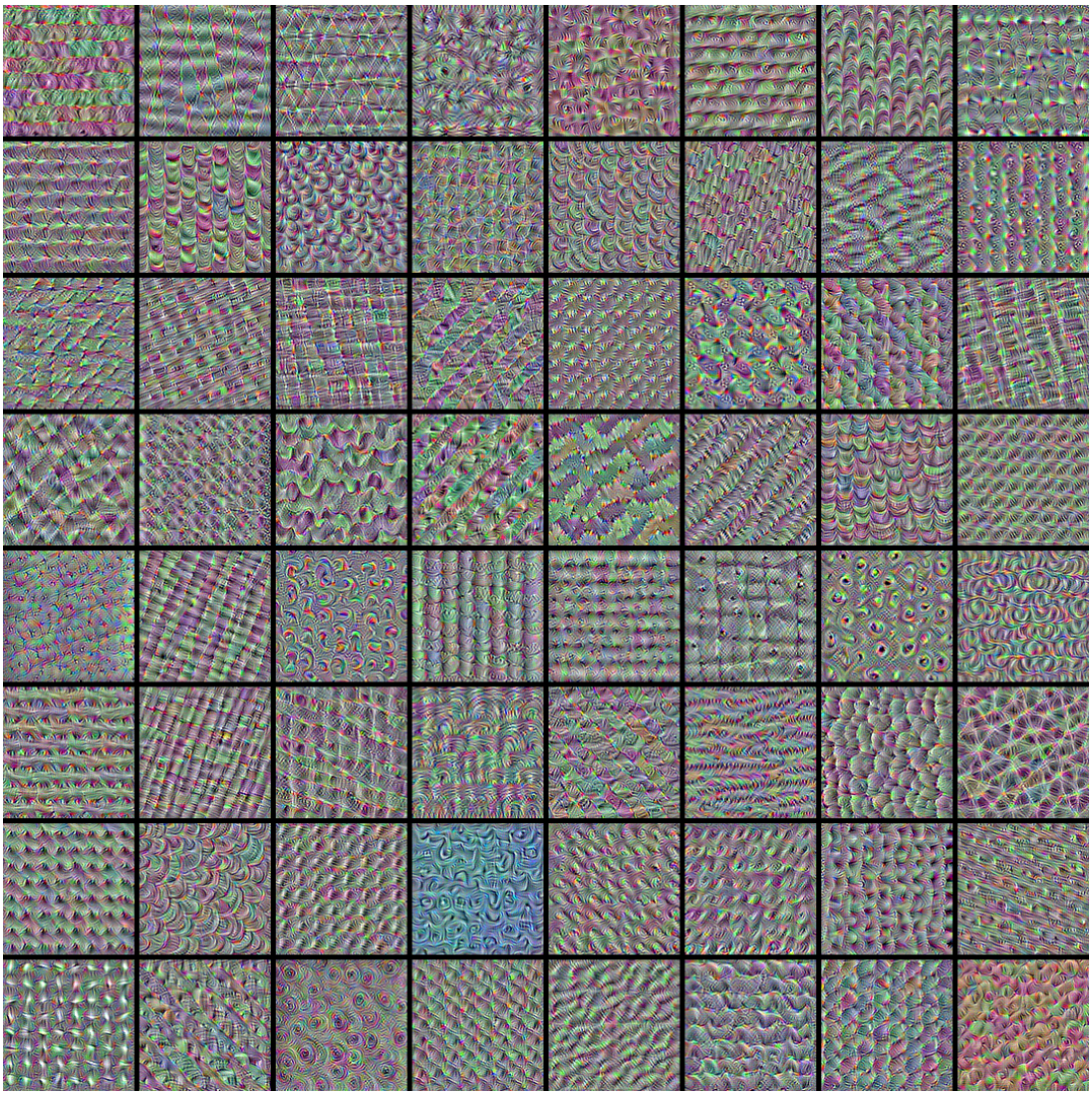
A camada de convolução é responsável por extrair, bem como mapear o conteúdo da imagem, transformando-o em dados. Esse processo ocorre por meio da aplicação de pequenos blocos, conhecidos como filtros, que permitem a obtenção das informações de sub-blocos da imagem. Já na camada de *pooling* os blocos que contém as informações extraídas na etapa de convolução são recebidos e as informações simplificadas, resumindo os dados do sub-bloco da imagem em um único valor e os repassa para uma camada totalmente conectada; há mais três elementos fundamentais na composição de uma Rede Neural Convolucional (PASSOS, 2021).

1. **Camada totalmente conectada:** Aqui é onde é iniciado o processo para classificar as informações extraídas pelas camadas anteriores. A camada totalmente conectada achata o sub-bloco contendo os dados extraídos, ou seja, o bloco é transformado em uma única linha que contém todas as informações extraídas. (PASSOS, 2021).
2. **Camada de dropout:** Possui a responsabilidade de reduzir o *overfitting* - sobreajuste - da rede. Ele ocorre quando a rede decora os dados utilizados no aprendizado e não é capaz de aplicar as relações aprendidas em dados novos. É comum dizermos que neste cenário o modelo gerado a partir da rede treinada não tem capacidade de generalização. Portanto, não apresenta bom desempenho quando colocado em produção. (PASSOS, 2021).
3. **Função de ativação:** A função de ativação, por sua vez, é responsável pelo aprendizado da rede, bem como pela relação entre as variáveis, decidindo quais neurônios serão ativados. (PASSOS, 2021).

Os filtros exibidos na Figura 8 são matrizes que permitem que as redes neurais convolucionais (RNC) identifiquem padrões na imagem, como bordas, formas, texturas, curvas, linhas

horizontais e verticais, cantos, cores e partes de um determinado objeto presente na imagem. Quanto mais profunda for a camada da rede, mais sofisticado é o filtro aplicado. (PASSOS, 2021).

Figura 8 – Filtros das camadas convolucionais

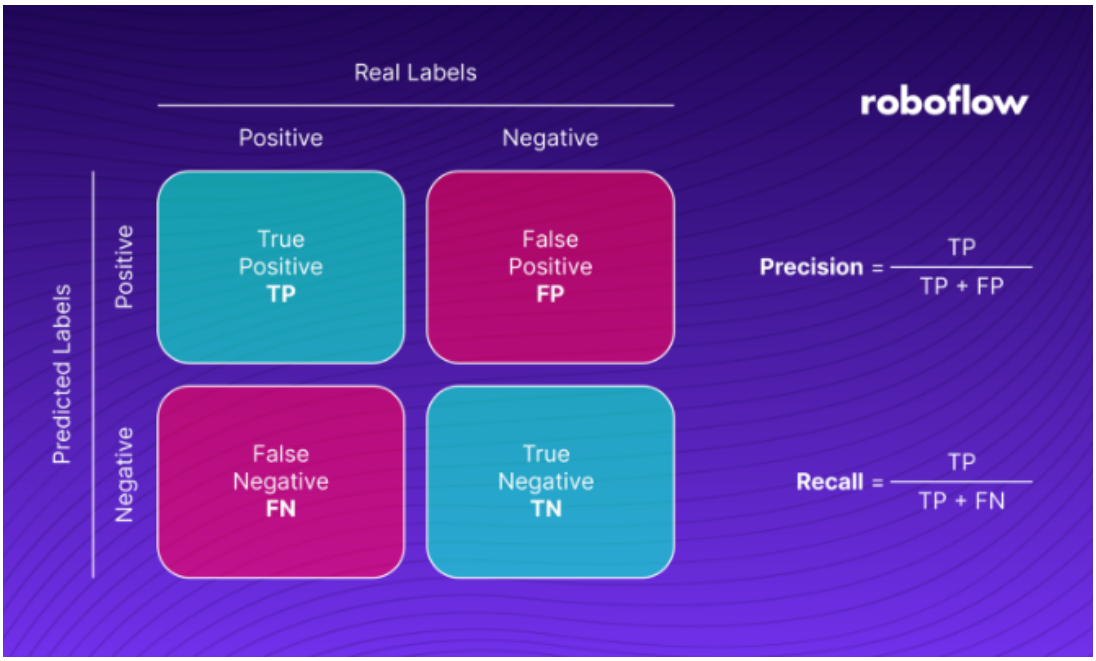


Fonte: Keras (2020)

2.4.4 Matriz de Confusão

Para avaliar a qualidade das detecções do modelo, geralmente o modelo é dividido em quatro grupos, como exibido na Figura 9. Um caso em que o modelo detecta corretamente um objeto é chamado de Verdadeiro Positivo (TP). Já quando um objeto que não está realmente na imagem é encontrado, ele é um Falso Positivo (FP). Por outro lado, quando um objeto na verdade não é detectado, ele é Falso Negativo (FN). O último grupo é formado por Verdadeiros Negativos (TN), Porém, no caso de detecção de objetos, isso não é levado em consideração. Os quatro grupos formam a chamada matriz de confusão. (SOLAWETS, 2020)

Figura 9 – Matriz de Confusão



Fonte: Solawets (2020)

2.4.5 YOLO

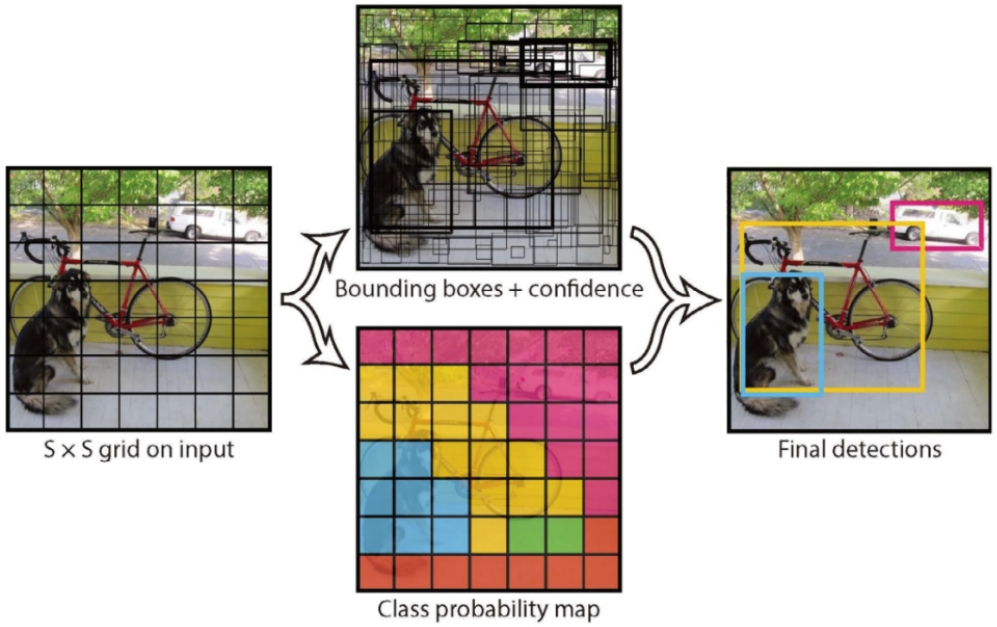
You Only Look Once ou *YOLO*, é um algoritmo apresentado por Joseph Redmon, Santosh Divvala, Ross Girshick e Ali Farhadi em 2016 durante seus estudos de doutorado. Esse algoritmo tem como objetivo apresentar uma nova abordagem para detecção de objetos, trazendo 3 principais benefícios em relação aos outros modelos conhecidos até então, sendo eles: (REDMON, 2016)

1. **Rapidez:** Em seu artigo, os autores afirmam que o *YOLO* atinge mais de duas vezes a precisão média;
2. **Detecção mais livre de erros:** Ao contrário das técnicas baseadas em proposta de janela deslizante e região, o *YOLO* vê a imagem inteira durante o treinamento e o tempo de teste, de modo que codifica implicitamente as informações contextuais sobre as classes, bem como sua aparência. Os pesquisadores ainda acrescentam que O Fast R-CNN, um dos principais métodos de detecção, confunde manchas de fundo em uma imagem com objetos porque não consegue ver o contexto maior, como o *YOLO*;
3. **Altamente generalizável:** O *YOLO* aprende representações generalizáveis de objetos. Quando treinado em imagens naturais e testado em obras de arte, o *YOLO* supera os principais métodos de detecção, como DPM e R-CNN, por uma ampla margem.

Em contraste com sistemas de detecção tradicionais, onde classificadores e localizadores eram utilizados sequencialmente localizando objetos e os classificando individualmente, em etapas separadas, o modelo *YOLO* aplica uma rede neural convolucional na imagem inteira como exibido na Figura 10. Essa rede profunda divide a imagem em regiões e deduz seus limites

e probabilidades para cada região, em que eles são avaliados e suas probabilidades calculadas (ZHAO, 2019)

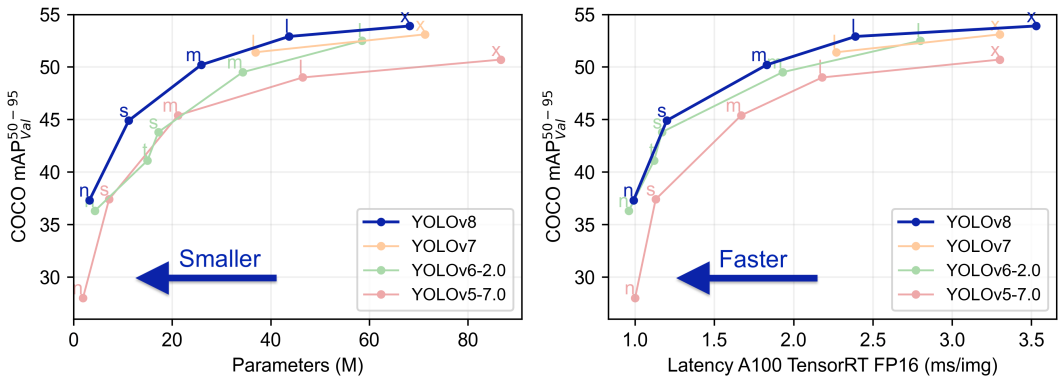
Figura 10 – Funcionamento do *YOLO*



Fonte: Redmon (2016)

Atualmente, o algoritmo encontra-se em sua oitava versão e tem seus desempenhos comparados às versões anteriores na Figura 11.

Figura 11 – Comparação com outros *YOLOs*



Fonte: Ultralytics (2022)

Os modelos *YOLOv8 Detect*, *Segment* e *Pose* são pré-treinados no conjunto de dados COCO (Common Objects in Context), que incluem 80 classes pré-treinadas. Esse conjunto de dados contém 330 mil imagens, com 200 mil imagens com anotações para tarefas de detecção,

segmentação e legenda de objetos e é projetado para incentivar a pesquisa em uma ampla variedade de categorias de objetos e é comumente usado para *benchmarking* de modelos de Visão Computacional. (ULTRALYTICS, 2023).

Na Figura 12 é possível visualizar os tipos de modelo do *YOLO*:

Figura 12 – Tipos de modelos do *YOLO*

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	18.4	142.4	1.21	3.5	10.5
YOLOv8s	640	27.7	183.1	1.40	11.4	29.7
YOLOv8m	640	33.6	408.5	2.26	26.2	80.6
YOLOv8l	640	34.9	596.9	2.43	44.1	167.4
YOLOv8x	640	36.3	860.6	3.56	68.7	260.6

Fonte: Ultralytics (2022)

Para definição do modelo escolhido, a rede foi aplicada a uma imagem do próprio autor para comparar duas versões do algoritmo. O resultado é visto nas Figuras 13 e 14.

Figura 13 – Algoritmo aplicado em foto do autor na versão *nano*

Fonte: Autor

Figura 14 – Algoritmo aplicado em foto do autor na versão *large*

Fonte: Autor

Percebe-se que a precisão na versão *large* é levemente maior, no entanto, ainda não é suficiente para justificar a utilização, visto que leva mais tempo de processamento. Outro teste realizado, desta vez para validação do tema proposto, foi a confirmação de que a única classe referente a placas de trânsito presente no conjunto de imagens base do *YOLO* é a placa de Pare ou "stop sign"como demonstrado na Figura 15.

Figura 15 – Conferência de classe *"Stop sign"* existente

Fonte: Autor

2.5 Sinalização de trânsito no Brasil

O trânsito brasileiro é regulamentado pela Lei nº 9.503, de 23 de setembro de 1997: o Código de Trânsito Brasileiro (CTB) que define atribuições das diversas autoridades e órgãos ligados ao trânsito, fornece diretrizes para a Engenharia de Tráfego e estabelece normas de conduta, infrações e penalidades para diversos usuários deste complexo sistema. (DETRAN.SP, 2022).

Segundo o Art. 80 do CTB, sempre que necessário, será colocada ao longo da via, sinalização prevista neste Código e em legislação complementar, destinada a condutores e pedestres, vedada a utilização de qualquer outra. Essa sinalização será colocada em posição e condições que a tornem perfeitamente visível e legível durante o dia e a noite, em distância compatível com a segurança do trânsito, conforme normas e especificações do Conselho Nacional de Trânsito. (CONTRAN, 2022).

2.5.1 Sinalização Vertical

Os sinais viários, normalmente placas, estão fixados na posição vertical, ao lado da via ou suspensos sobre ela. Transmitem mensagens através de legendas ou símbolos pré-estabelecidos e servem para aumentar a segurança, ordenar os fluxos de tráfego e orientar os usuários das vias. A sinalização vertical, de acordo com sua função, pode ser: sinalização de regulamentação, sinalização de advertência e sinalização de indicação. (CONTRAN, 2022).

2.5.2 Sinalização de Regulamentação

São sinais que informam aos usuários as proibições, obrigações e restrições impostas no ponto ou trecho sinalizado. Desobedecer aos sinais de regulamentação significa infringir as normas de trânsito e, portanto, estar sujeito a penalidades e medidas administrativas. Os símbolos são em preto, o fundo é branco e a cor vermelha indica obrigação, proibição ou restrição. Algumas placas apresentam tarja vermelha na diagonal, que significa proibição. Os sinais R-1 e R-2 são os únicos sinais de regulamentação que diferem do formato circular dos demais, e podem ser reconhecidos à distância. Existem ainda, informações complementares à sinalização de regulamentação podendo utilizada uma placa adicional ou incorporada à placa principal, formando um só conjunto. (CONTRAN, 2022).

2.5.3 Sinalização de Advertência

São sinais que servem para alertar os usuários, com antecedência, sobre situações de perigo na via, para que possam reagir de forma adequada. Os símbolos são em preto, e a cor de fundo é amarela que indica atenção e perigo. O formato das placas geralmente é quadrado. As recomendações básicas em qualquer situação de risco são: reduzir a velocidade e redobrar a atenção. Além disso, o condutor deve interpretar corretamente o sinal de advertência e tomar os cuidados necessários para cada situação. Desrespeitar a sinalização de advertência significa imprudência e negligência. Também existem sinalizações especiais e informações complementares à sinalização de advertência, para faixas ou pistas exclusivas de ônibus e para pedestres. Elas podem estar incorporada à placa principal ou em uma placa adicional abaixo da principal. (CONTRAN, 2022).

2.5.4 Sinalização de Indicação

Tem caráter informativo ou educativo. Servem para indicar vias, locais de interesse, distâncias e orientar condutores sobre percurso, destino e serviços auxiliares. (CONTRAN, 2022).

2.5.5 Outras sinalizações

- **Sinalização Horizontal:** Refere-se aos sinais que apresentam-se ao condutor pintados ou desenhados sobre o piso, na posição horizontal, na forma de faixas símbolos ou inscri-

ções;

- **Dispositivos Auxiliares:** Aumentam a visibilidade dos sinais e chamam a atenção para obstáculos no local. São confeccionados em material refletivo ou luminoso;
- **Sinalização Semafórica:** Sinais luminosos controlados eletricamente que servem para controlar o fluxo de veículos e pedestres;
- **Sinalização de Obras:** Muito semelhante as placas de advertência com a diferença do fundo laranjado e caráter temporário;
- **Gestos de agentes de autoridade de trânsito:** Sempre que a sinalização for efetuada pelo agente, esta tem prioridade sobre as demais.
- **Gesto de condutores:** São sinais auxiliares, indicativos de manobras;
- **Sinais sonoros:** São os apitos do policial ou agente de trânsito.

2.6 Veículos Autônomos

Os primeiros veículos motorizados amplamente utilizados foram os barcos a vapor e os trens. Os trilhos guia dos trens foram adotados principalmente para suportar seu enorme peso, mas também serviam para controle direcional. Cerca de uma década após sua invenção, o avião ganhou seu primeiro piloto automático. (WEBER, 2014).

Não é por acaso que a autonomia chegou primeiro aos veículos diferentes dos terrestres já que, mesmo sendo difíceis, são ambientes relativamente tolerantes para veículos autoguiados. Um dos principais desafios para veículos autônomos terrestres são a quantidade de questões que enfrentam, principalmente ao tratar-se de vidas de pessoas inocentes. Algumas ideias como transformar as pistas em trincheiras que manteriam os carros separados em suas próprias "trilhas" eram pautadas. No entanto, alguns anos depois, algumas das ideias se alteraram, continuando com o objetivos principais da segurança, velocidade e acesso, mas também carros partilhando as estradas comuns. (WEBER, 2014).

Durante as décadas de 1980 e 1990, a área da Inteligência Artificial e a capacidade dos computadores avançaram significativamente, possibilitando importantes realizações no campo dos carros autônomos. Desde então, várias iniciativas surgiram, incluindo alguns desafios, como o realizado em 2004 pela Administração de Projetos de Pesquisa Avançada de Defesa dos EUA (DARPA), que ofereceu um prêmio de 1 milhão de dólares para quem trabalhasse em veículos autônomos para competir. Atualmente, existem várias iniciativas em andamento para o desenvolvimento de carros autônomos, com testes principalmente na Alemanha e nos EUA. (WEBER, 2014).

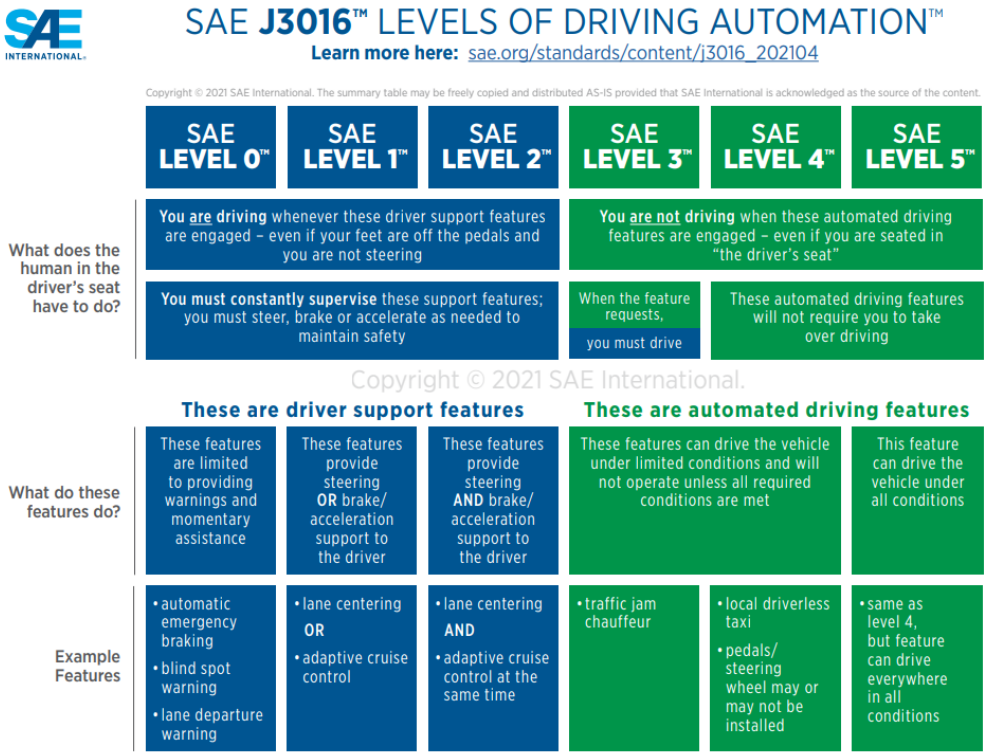
> Um carro autônomo é um veículo que, em última instância, pode funcionar sem a intervenção humana. Para que isso seja possível, algumas tecnologias devem estar embarcadas no automóvel como controlador de velocidade, GPS de alta resolução, sensores como radares e câmeras que interpretam o entorno do veículo, computadores e algoritmos de Inteligência Artificial adequados. (NETWORKS, 2023).

A Sociedade dos Engenheiros Automotivos (SAE) desenvolveu uma classificação amplamente aceita para descrever os níveis de autonomia dos veículos. Essa classificação é conhecida como os "Níveis SAE de Autonomia"e ajuda a entender a capacidade e o grau de autonomia dos veículos. Os níveis são:

- **Nível 0:** Sem Automação de Condução: O veículo não possui recursos de automação e todas as funções de direção são realizadas pelo motorista.
- **Nível 1:** Assistência ao Motorista: O veículo possui recursos que auxiliam o motorista em tarefas específicas, como controle de velocidade de cruzeiro adaptativo ou assistência de direção em faixa. No entanto, o motorista deve estar sempre atento e manter o controle total do veículo.
- **Nível 2:** Automação de Direção Parcial: O veículo é capaz de assumir o controle de certas funções de direção, como acelerar, frear e fazer curvas em determinadas condições. No entanto, o motorista ainda deve estar preparado para retomar o controle do veículo a qualquer momento e deve permanecer atento.
- **Nível 3:** Automação Condicional: O veículo pode assumir o controle em certas situações e ambientes específicos, permitindo que o motorista se concentre em outras tarefas. No entanto, o motorista deve estar preparado para retomar o controle quando solicitado pelo sistema.
- **Nível 4:** Alta Automação de Condução: O veículo é capaz de realizar a maioria das tarefas de direção sem intervenção do motorista em determinadas condições e ambientes predefinidos. Embora um motorista possa ter a opção de assumir o controle, o veículo é capaz de operar de forma autônoma na maioria das situações.
- **Nível 5:** Automação de Direção Completa: O veículo é totalmente autônomo, capaz de realizar todas as tarefas de direção em qualquer condição ou ambiente. Não requer intervenção ou supervisão humana.

Essa classificação ajuda a entender os diferentes níveis de autonomia dos veículos, fornecendo uma base comum para a discussão e o desenvolvimento contínuo da tecnologia de direção autônoma. Mais informações são exibidas na Figura 16.

Figura 16 – Níveis de carros autônomos conforme SAE J3016



Fonte: SAE (2018)

3 TRABALHOS CORRELACIONADOS

Durante as pesquisas, alguns trabalhos foram selecionados para agregar a este trabalho final, pois todos tratam de um assunto em comun: Treinamento para identificação de placas de trânsito. Neste subcapítulo, eles serão abordados.

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

O trabalho do aluno Jean Lucas da Silva Cimirro foi realizado no ano de 2022. Seu TCC defende que a expansão contínua das cidades cria uma demanda crescente para a instalação de novas placas e manutenção das existentes. Por isso, é necessário registrar com precisão os dados dos sinais, como o número de sinais, tipo, condição e localização geográfica. Um conjunto de dados dividido em três classes: regulamentação, advertência e indicação foi criado, possuindo um total de 3283 imagens, posteriormente divididas entre treinamento e validação. O objetivo foi reconhecer sinais de trânsito brasileiros utilizando o método *YOLO* em sua quinta versão. Foram realizados testes com três arquiteturas do *YOLO*: *small*, *large* e *extra large*, obtendo precisões de 92,9%, 92,6% e 92,0%, respectivamente. Além disso, foi criado um *dataset* das placas brasileiras e uma interface para o usuário com o utilitário *QtDesigner*. (CIMIRRO, 2022) [1]

3.2 Rotulagem e treinamento do *YOLO* para reconhecimento de placas de trânsito brasileira

O TCC do aluno Bruno Oliveira Dantas foi realizado do ano de 2021. A pergunta de sua pesquisa é: "Como detectar e classificar placas de trânsito brasileiras utilizando o *YOLOv3*, e ainda documentar o processo de rotulagem e treinamento?"Bruno desenvolve seu próprio *dataset* com 12 placas de trânsito mais comumente encontradas em vias públicas urbanas, além de aplicar seu modelo em vários vídeos mostrando diversas situações climáticas. Ao fim, conclui que o modelo gerado teve como resultado geral, uma acurácia de 86% e uma precisão de 100%. (DANTAS, 2021a) [2]

3.3 Criação e validação de uma base de dados com elementos do trânsito brasileiro para veículos autônomos

O artigo foi publicado na revista *Brazilian Applied Science Review* e descreve a importância de um bom *dataset*, já que o Brasil possuia na época, muitas universidades trabalhando

[1] https://dspace.unipampa.edu.br/handle/riu/7527
[2] https://bdm.unb.br/handle/10483/30375

em protótipos de carros autônomos, tendo em foco o desenvolvimento do veículo em si e não em uma base de dados. O *dataset* foi dividido em 47 classes com cones, pedestres, bicicletas, ciclistas, veículos, motocicletas, motociclistas, semáforos, placas de regulamentação, advertência, serviços auxiliares, identificação e indicação. Além disso, o *software Blender3D* foi utilizado para aumentar o número de imagens, adicionando filtros para que as mesmas tivessem variações para enriquecer o repertório como variação de brilho, contraste e temperatura da cor dentre outros, totalizando 79627 imagens disponíveis para posterior processamento. Após, uma rede neural foi treinada com *YOLOv3* e o resultado obtido foi uma detecção de 21,5% de objetos, o qual 83% destes foram classificados corretamente no cenário virtual; e uma detecção de 61,8% de objetos, o qual 88% destes foram classificados corretamente no cenário real. (CAMPOS ELDER DE OLIVEIRA RODRIGUES, 2020) [3]

3.4 Classificação de Placas de Trânsito com Redes Neurais para Automação de Veículos

O estudo conduzido por Gustavo Bulka Bonafé Bademian e Celso Ap.de França aborda o emprego de redes neurais convolucionais na categorização de sinais de trânsito. O principal objetivo deste trabalho é o desenvolvimento de uma ferramenta destinada a veículos independentes, com o intuito de aprimorar a tomada de decisões durante a condução. Isso pode incluir o controle da velocidade do veículo, identificação de mudanças possíveis no trajeto ou mesmo a apresentação de informações ao passageiro por meio de um computador de bordo. A pesquisa se vale do conjunto de dados *German Traffic Sign 1*, composto por 39.209 imagens de treinamento e 12.630 imagens de teste, distribuídas em 43 categorias distintas, cada uma representando um tipo específico de sinal de trânsito. Após o treinamento, o resultado de acurácia chegou a ser 99,2% no processo de identificação e classificação da base de dados. (FRANçA, 2022) [4]

3.5 Detecção e Classificação de sinais de tráfego em tempo real com base no algoritmo *YOLO* Versão 3

O artigo em questão trata da detecção e classificação de sinais de trânsito em tempo real com base no algoritmo *YOLOv3*. O estudo foi realizado por V. Sichkar e S. Kolyubin, pesquisadores da Universidade ITMO, na Rússia. O artigo descreve o processo de detecção e classificação de sinais de trânsito em duas etapas. Na primeira etapa, os sinais de trânsito são agrupados em quatro categorias com base em suas formas. Na segunda etapa, é realizada uma classificação precisa dos sinais de trânsito localizados em uma das quarenta e três categorias pré-definidas. O modelo para detecção de sinais de trânsito foi treinado no *German Traffic Sign Detection Benchmark* (GTSDB) com 630 e 111 imagens RGB para treinamento e validação, respectivamente. O modelo de classificação foi treinado com a mesma base de dados e com 66.000 imagens RGB em uma biblioteca *numpy* com dimensão 19×19 de filtros de camada

[3] https://ojs.brazilianjournals.com.br/ojs/index.php/BASR/article/view/10783/9009
[4] https://repositorio.ufscar.br/handle/ufscar/15692

convolucional e atingiu precisão de 0,868 no conjunto de dados de teste. Os resultados experimentais ilustraram que o treinamento do primeiro modelo de rede profunda com apenas quatro categorias para localização de sinais de trânsito produziu alta precisão de mAP (média de precisão média), atingindo 97,22% . A camada convolucional adicional do segundo modelo aplicado para classificação final cria um sistema inteiro eficiente. Experimentos de processamento de arquivos de vídeo demonstraram quadros por segundo (FMS) entre 36 e 61, o que torna o sistema viável para aplicações em tempo real. (V.N., 2020)

3.6 Considerações sobre trabalhos correlacionados

Ao finalizar a análise dos trabalhos correlacionados é visível a importância do tema e da diversidade de pesquisas na área, pois com o número de veículos aumentando nas vias, a preocupação com a segurança é crescente e legítima. O presente trabalho expande o conhecimento existente ao utilizar a versão 8 da *YOLO*, que é mais recente e apresenta melhoras significativas em termos de precisão e velocidade, além de apresentar um aumento de 4 a 9 pontos percentuais em mAP em relação a versão 5 para um tempo de execução semelhante.

Além disso, este trabalho tem como objetivo preencher algumas lacunas no conhecimento existente, como melhorar o desempenho do reconhecimento de placas de trânsito brasileiras, utilizar técnicas de rotulagem, testar a rede neural treinada, tanto em tempo real quanto em vídeos previamente gravados e identificar alguns desafios que a tecnologia dos veículos autônomos enfrenta, como placas desgastadas ou ocultas. Agrego ainda que este trabalho apresenta um sólido conteúdo teórico sobre conceitos importantes para entendimento do assunto, como redes neurais artificiais, visão computacional e placas de trânsito brasileiras.

4 FERRAMENTAS UTILIZADAS

Além do embasamento teórico é necessário conhecer os instrumentos utilizados no desenvolvimento deste trabalho. Este capítulo detalha as ferramentas utilizadas, sendo o *Roboflow*, o software de rotulagem de objetos o *Google Colab* para treinamento da rede neural, a biblioteca *Ultralytics* e a linguagem de programação *Python*.

4.1 Roboflow

O *Roboflow* é uma plataforma online que capacita os desenvolvedores a criarem seus próprios aplicativos de Visão Computacional, independentemente de seu conjunto de habilidades ou experiência, além de oferecer a hospedagem de um banco de dados de mais de 200.000 conjuntos de dados. (ROBOFLOW, 2023) No trabalho, foi utilizado para realizar a rotulagem das imagens e a exportação desse material. Uma alternativa para esse software é o *label image*, amplamente utilizado pelos desenvolvedores.

4.2 Overleaf

O overleaf é um editor da linguagem *LaTeX* utilizado para escrever e editar trabalhos e arquivos (OVERLEAF, 2023). No trabalho, foi utilizado para estruturar todo o documento, além de referenciar e ajustar as imagens selecionadas. Algumas alternativas para a ferramenta são: TeXShop, Kile e LyX, que também suportam o *LaTeX*.

4.3 Google Colab

O *Google Colab*, ou *Google Collaboratory*, é um serviço de armazenamento em nuvem de notebooks (ou células de códigos) voltados à criação e execução de códigos em *Python*, diretamente em um navegador. Com ele é possível ler, desenvolver e rodar códigos e *rich texts* em documentos interativos. (GOOGLE, 2023a). Sua utilização foi imprescindível pois mesmo na versão gratuita, dispõem de diversos recursos para treinamento de rede neural, como unidade de processamento gráfico (GPU), unidades de processamento de tensores (TPU), memória, entre outros.

4.4 Google Drive

O *Google Drive* é uma plataforma de armazenamento em nuvem criado pela empresa *Google*. Com ele é possível armazenar, compartilhar e colaborar em arquivos e pastas usando qualquer dispositivo móvel, *tablet* ou computador. (GOOGLE, 2023b). Sem ele, seria necessá-

rio realizar o *upload* das imagens no Google Colab em cada execução de código, pois o Google Colab possui uma memória volátil.

4.5 Linguagem de programação *Python*

Python é uma linguagem de programação criada pelo matemático e programador Guido van Rossum em 1991 e, embora seja considerada uma linguagem mais antiga, vem mantendo sua posição como uma das mais utilizadas. Uma das principais características da linguagem é ser de fácil aprendizagem pois seu objetivo inicial era permitir um código enxuto e menos verboso. Além disso, destaca-se por ser multiplataforma, possuir coletor de lixo para gerenciar automaticamente o uso de memória, modo interativo entre outros. (PAIVA, 2020).

4.5.1 PyCharm Community Edition

A IDE PyCharm Community Edition pertence a empresa tcheca Jetbrains. Ela fornece algumas *features* como complementação de código inteligente, inspeções de código, realce dinâmico de erros e correções rápidas e recursos de navegação avançados. Neste trabalho, a IDE é utilizada para execução de dois códigos: um para executar a rede em tempo real via *webcam* e outra para executar em um vídeo previamente gravado. (S.R.O, 2023).

4.5.2 Ultralytics

A *Ultralytics* é uma biblioteca de aprendizado de máquina de código aberto que oferece soluções de ponta para uma ampla gama de tarefas de Inteligência Artificial, incluindo detecção, segmentação, classificação, rastreamento e estimativa de pose. O *Ultralytics YOLOv8* é a versão mais recente do modelo de detecção de objetos e segmentação de imagem em tempo real, construído com base em avanços de ponta em aprendizado profundo e Visão Computacional, oferecendo desempenho incomparável em termos de velocidade e precisão (JOCHER, 2023).

4.5.3 OpenCV

OpenCV é uma biblioteca de código aberto projetada para fornecer uma infraestrutura comum para aplicações de Visão Computacional e e aprendizado de máquina. Ela oferece mais de 2500 algoritmos otimizados que possibilitam o processamento e aprimoramento de imagens, classificação de ação humana em vídeos, reconhecimento facial, entre outros. Alguns usos implantados do *OpenCV* abrangem desde unir imagens de *StreetView*, detectar invasões em vídeos de vigilância, e inspecionar rótulos de produtos em fábricas em todo o mundo. O código-fonte é escrito nativamente em C++, mas também possui suporte em diversas outras linguagens, como MATLAB e Java. Além disso, é compatível com os Sistemas Operacionais *Windows, Linux, Android e Mac OS* (OPENCV, 2023).

4.6 Câmera

A *webcam* é um periférico de computador que permite aos usuários gravar ou fotografar o que for necessário. A câmera utilizada para execução do algoritmo em tempo real foi a Webcam Kross, Full HD 1080P com foco manual. Possui uma resolução de 1920x1080px, possui interface USB e de conceito *plug and play*. A Figura 17 exibe o equipamento.

Figura 17 – Câmera utilizada

Fonte: Autor

5 *Dataset* e rotulagem

Datasets são componentes fundamentais em um projeto de *Machine Learning*. Eles são a base utilizada para que os algoritmos consigam aprender, evoluir e exibir seus resultados. No trabalho aqui apresentado, é necessário um conjunto de dados com boa visibilidade e com vários exemplos para o sucesso da rede treinada.

5.1 *Dataset utilizado*

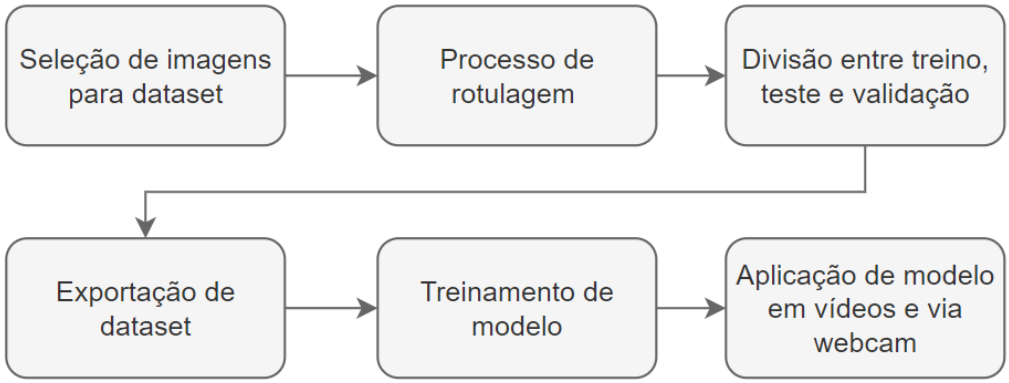
Para o primeiro treinamento da rede, o *dataset* utilizado foi desenvolvido por alunos do Instituto Federal de Santa Catarina [1] (MEDEIROS, 2018), com quatro objetivos específicos: 1) Aprender sobre a criação de bases de imagens; 2) Capturar imagens de vias que contenham placas de sinalização; 3) Aplicar técnicas de Processamento de Imagens para adequar as imagens capturadas; 4) Disponibilizar o banco de imagens publicamente através de uma licença *open source*.

Após a finalização da rotulagem e treinamento, notou-se que seriam necessárias mais imagens para o sucesso do treinamento da rede neural e por isso, outros conjuntos de imagens foram utilizados. Através do *Roboflow Universe*, um conjunto de dados foi encontrado [2] (JUNIOR, 2021) auxiliando no treinamento, além de algumas imagens disponibilizadas pelo graduando Bruno Oliveira Dantas com seu trabalho: "Rotulagem e treinamento do Yolo para reconhecimento de placas de trânsito brasileiras"[3] (NASCIMENTO, 2021), e pelo graduando Guilherme Lopes do Nascimento com seu trabalho: "Classificação de placas de trânsito utilizando redes neurais convolucionais"[4]. (DANTAS, 2021b).

Na Figura 18 é possível visualizar as etapas do desenvolvimento:

[1] https://wiki.sj.ifsc.edu.br/images/d/df/DiegoMedeiros-Projeto-13-2018-CSJ-Placas.pdf
[2] https://universe.roboflow.com/edward-junior/placas-de-transito-7po0k/dataset/1
[3] https://bdm.unb.br/handle/10483/30375
[4] https://repositorio.pucgoias.edu.br/jspui/bitstream/123456789/3542/1/TCC-Guilherme-2021.pdf

Figura 18 – Fluxograma de treinamento do modelo

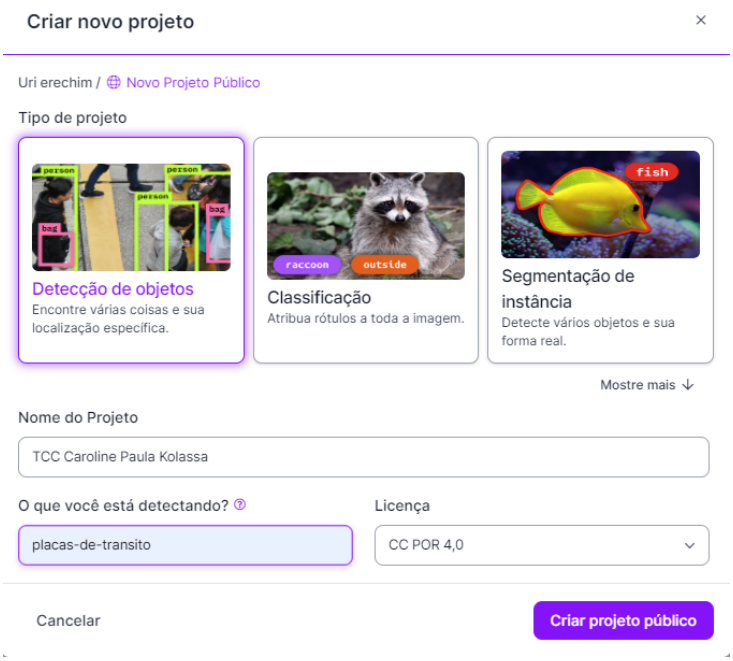


Fonte: Autor

5.1.1 Criação do projeto para rotulagem

Para criação do projeto, o primeiro passo foi a criação da conta no *Roboflow* e posterior a isso, a criação do projeto das placas, como explicado na Figura 19. Após, todas as imagens foram selecionadas e carregadas a plataforma.

Figura 19 – Fluxograma de treinamento do modelo



Fonte: Autor

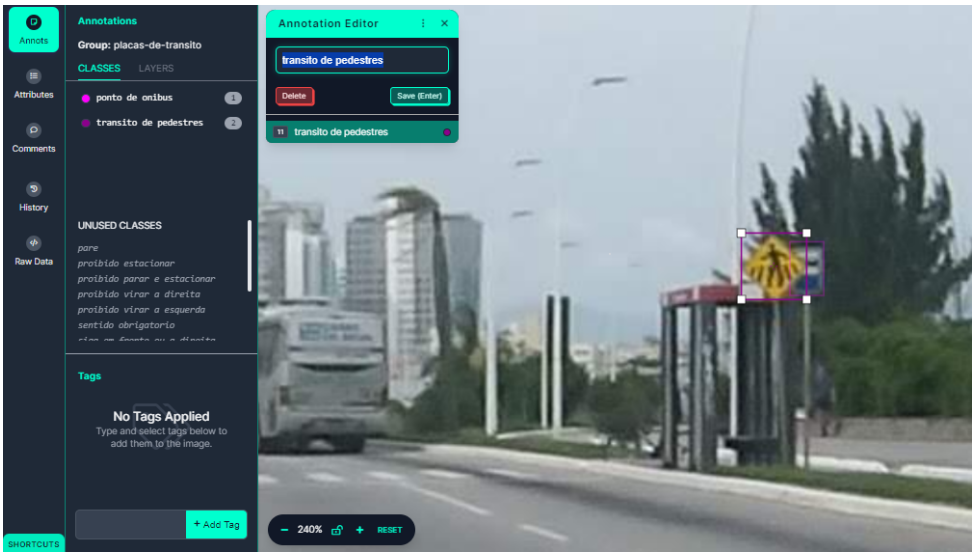
5.1.2 Técnicas Utilizadas

Para a rotulagem, as seguintes técnicas foram utilizadas, disponibilizadas pelo *Roboflow*:

- Criar um conjunto de dados específico para a tarefa que deseja-se realizar;
- Criar caixas delimitadoras precisas e ajustadas ao objeto;
- Ao anotar objetos, certificar-se de que as caixas delimitadoras incluam todos os detalhes importantes;
- Ao lidar com oclusão, desenhar uma caixa delimitadora para cada objeto visível;
- Não cortar parte do objeto ao desenhar a caixa delimitadora;
- Usar uma terminologia clara e concisa para descrever os objetos;
- Usar ferramentas de rotulagem que sejam fáceis de usar e que forneçam *feedback* preciso.

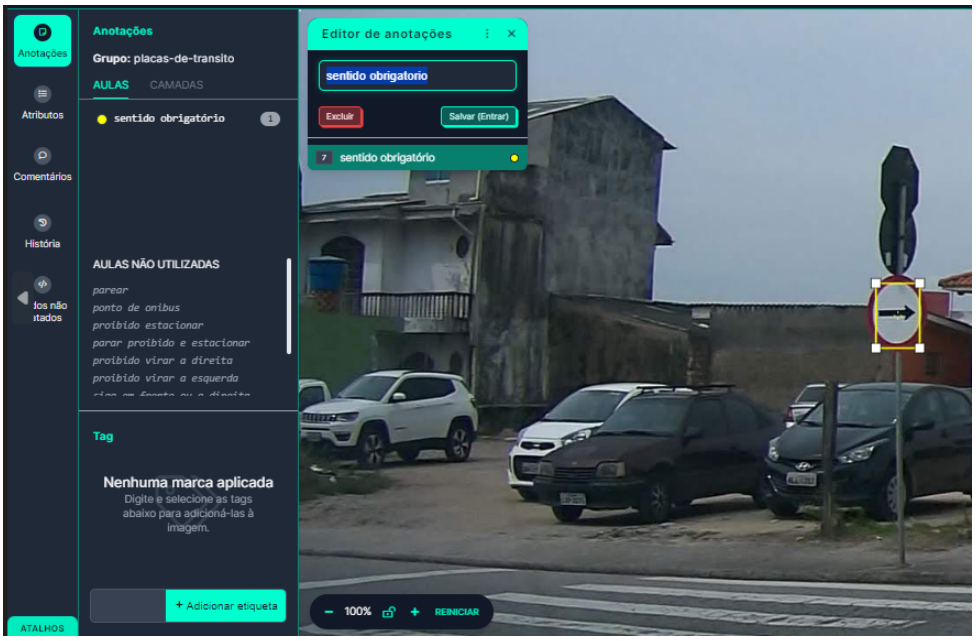
Alguns exemplos da aplicação dessas técnicas encontram-se nas Figuras 20, 21 e 22.

Figura 20 – Exemplo de oclusão entre placas



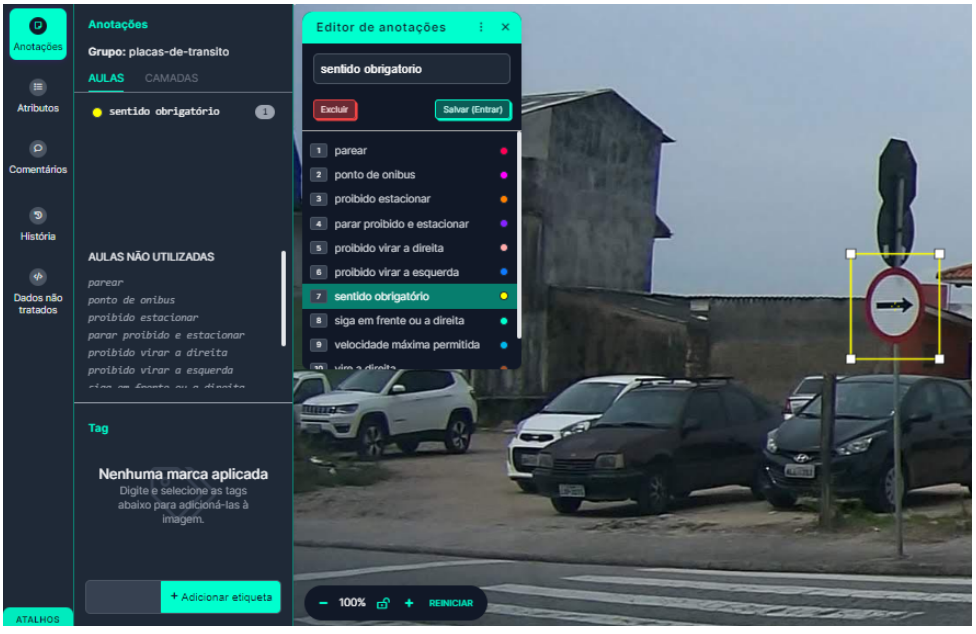
Fonte: Autor

Figura 21 – Exemplo de caixa delimitadora muito justa



Fonte: Autor

Figura 22 – Exemplo de caixa delimitadora muito espaçada



Fonte: Autor

5.1.3 Processo de Rotulagem

Após a importação das imagens no *framework Roboflow*, cada imagem é rotulada com uma caixa delimitadora utilizando o nome da classe ao qual a placa pertence. Após o processo ser realizado em todas as imagens, é separado a porcentagem de treino, validação e teste, sendo

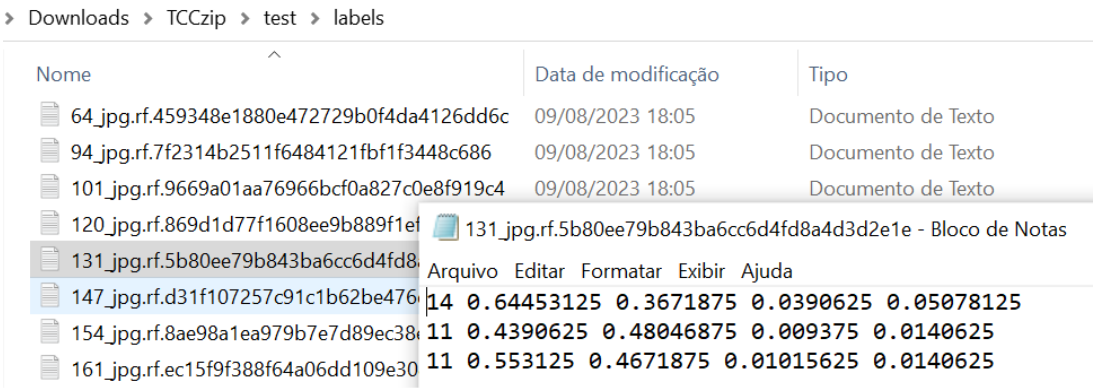
recomendadas as seguintes porcentagens, respectivamente: 70%, 20% e 10%. A exportação é realizada e o resultado é um arquivo .zip com o conteúdo de 3 diretórios e um arquivo com extensão .yaml, como demostrado na Figura 23.

Figura 23 – Dataset exportado

Fonte: Autor

Ao acessar os diretórios, há mais duas pastas separadas por *"Image"* e *"Labels"*. Na pasta *Image* estão contidas todas as imagens e na pasta *Labels* estão todos os arquivos de rotulação em formato *YOLO*, onde a primeira informação é a classe da placa e na sequência, as cordenadas (x,y) e (w,h), representando a largura e altura do *bounding box* como demonstrado na Figura 24.

Figura 24 – Exemplo da notação *YOLO*



Fonte: Autor

Os arquivos *README* são arquivos com informações sobre a exportação, licença e URL do *dataset* [5] e não são necessários para o treinamento da rede. O arquivo data.yaml apresentado na Figura 25 apresenta as informações necessárias para o treinamento, como o nome das classes e caminho dos arquivos.

[5] https://universe.roboflow.com/uri-erechim/tcc-caroline-paula-kolassa-final

Figura 25 – Arquivo data.YAML

data.yaml

```yaml

license: CC BY 4.0
project: tcc-caroline-paula-kolassa-final
url: https://universe.roboflow.com/uri-erechim/tcc-caroline-paula-kolassa-final/dataset/1
version: 1
workspace: uri-erechim
names: ['curva a esquerda', 'curva acentuada a direita', 'curva acentuada a esquerda',
'de a preferencia', 'pare', 'permitido estacionar', 'ponto de onibus', 'proibido estacionar',
'proibido parar e estacionar', 'proibido virar a direita', 'proibido virar a esquerda',
'saliencia ou lombada', 'sentido obrigatorio', 'sentido proibido',
'siga em frente ou a direita', 'siga em frente ou a esquerda', 'transito de escolares',
'transito de pedestres', 'velocidade maxima permitida', 'vire a direita', 'vire a esquerda']
nc: 21
roboflow:
test: ../test/images
train: ../train/images
val: ../valid/images

```

Fonte: Autor

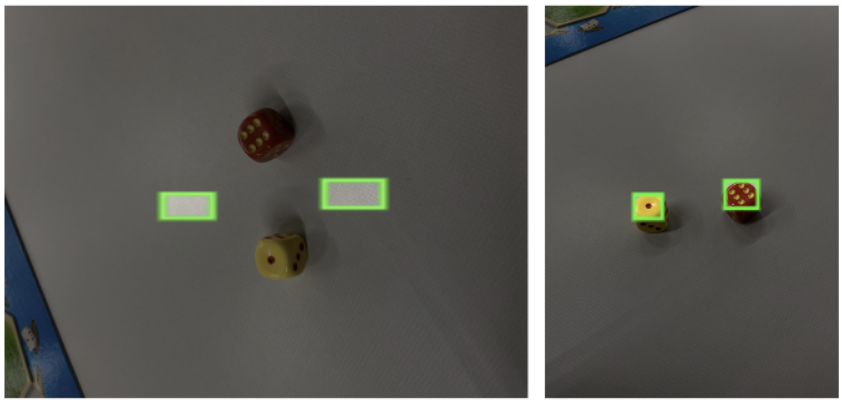
6 TREINAMENTO DA REDE NEURAL

Foram realizados vários treinamentos e utilizados vários conjunto de dados, mas neste trabalho, o foco foi somente no primeiro e no último treinamento. Ambos os treinamentos foram realizados pela plataforma do *Google Colab*, no entanto, o segundo treinamento também foi realizado na plataforma *Roboflow* para testes. Nota-se uma grande diferença entre o primeiro e o segundo treinamento, onde no segundo, o dataset foi melhor selecionado e rotulado, pois algumas imagens estavam prejudicando o modelo devido a sua baixa visibilidade. O número de classes também diminuiu, sendo 17 classes no primeiro treinamento e 11 classes no segundo treinamento.

6.1 Primeiro treinamento

No primeiro treinamento realizado, as imagens tiveram um pré-processamento para redimensionamento de diversos tamanhos para 640x640 e foram orientadas automaticamente, como visto na Figura 26. Essa última função impede um dos problemas mais comuns encontrados em projetos de Visão Computacional: perda dos metadados que determinam a orientação pela qual ela deve ser exibida em relação à forma como os *pixels* estão organizados no disco. (DWYER, 2020)

Figura 26 – Incompatibilidade de orientação EXIF entre anotação e imagem



Fonte: Brad Dwyer

Também fez-se uso da geração de imagens a partir das já carregadas para melhorar o desempenho do modelo. Os aumentos realizados foram: Cisalhamento de ±15° Horizontal e ±15° Vertical, exposição entre -25% e +25% e caixa delimitadora com desfoque de até 2,5px. Assim, o *dataset* inicial de 615 imagens ficou com 1486. Nas Figuras 27 e 28 é possível verificar como ficaram as configurações.

Figura 27 – Cisalhamento

Fonte: Autor

Figura 28 – Exposição

Fonte: Autor

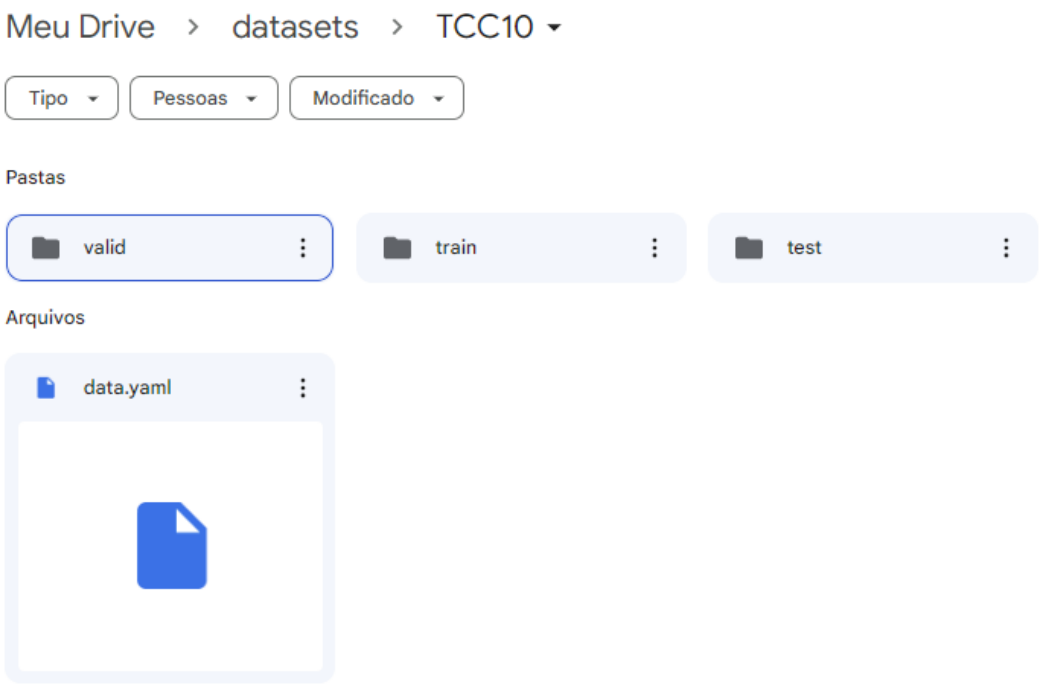
A porcentagem de treino, teste e validação, tornou-se respectivamente: 88%, 8% e 4%, e a Figura 29 exibe as placas rotuladas. Após a exportação das imagens, é necessário disponibi-

lizar os arquivos em uma pasta no *Google Drive*, como exibido na Figura 30. [1]

Figura 29 – Placas rotuladas no primeiro treinamento

Fonte: Autor

Figura 30 – Imagens Google Drive

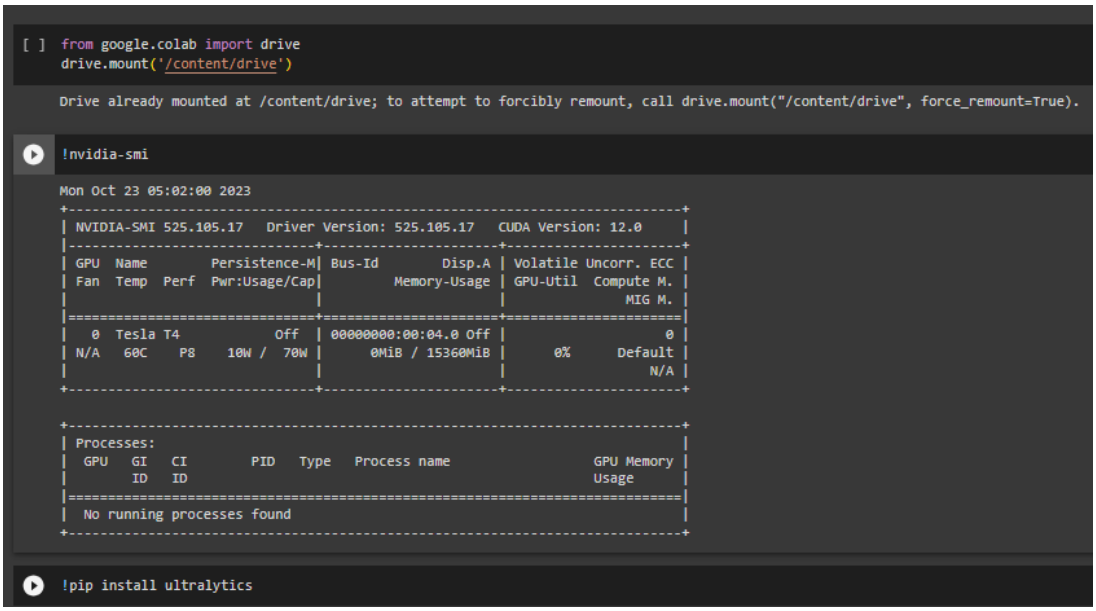


Fonte: Autor

Após, a ferramenta *Google Colab* foi iniciada, onde o projeto: Treinamento Rede do TCC.ipynb foi criado. Nesta etapa é necessário configurar o acelerador de Hardware para: T4 GPU e autorizar o acesso ao *Google Drive*, como exibido na Figura 31.

[1] https://app.roboflow.com/uri-erechim/tcc-caroline-paula-kolassa-final/2

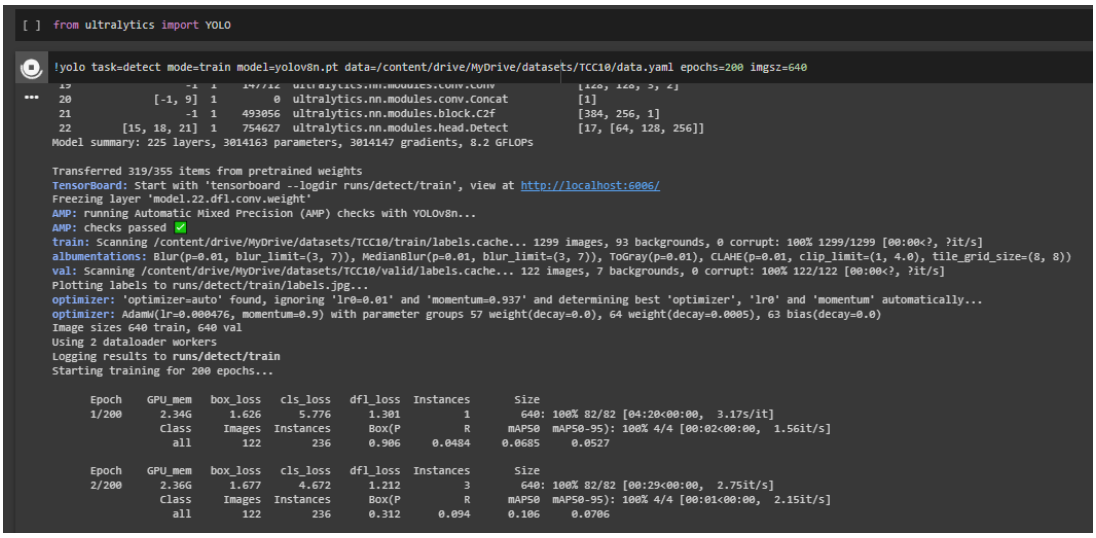
Figura 31 – Configurações para treinamento



Fonte: Autor

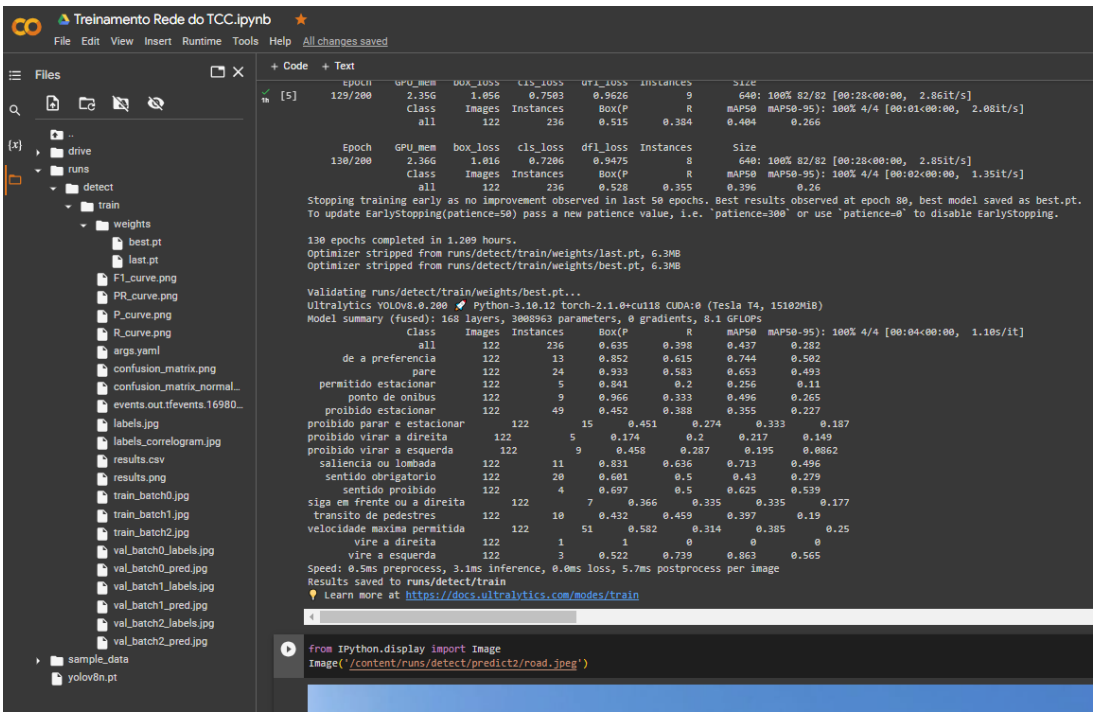
A instalação do pacote *ultralytics* é necessária, por meio do comando !pip, conjunto de pacotes do *python*. Após, a tarefa de detecção com o modelo *nano* do *YOLO* é escolhida, o caminho parametrizado, além do número de épocas e a dimensão da imagem. Quanto maior o número de épocas, maior será o tempo do treinamento. Problemas mais simples podem convergir rapidamente enquanto problemas mais complexos podem exigir mais épocas de treinamento. No entanto, um número excessivo de épocas pode levar ao *overfitting*, onde a rede neural se ajusta demasiadamente aos dados de treinamento e não generaliza bem para dados não vistos. (BROWNLEE, 2018). A Figura 32 mostra o início do treinamento e a Figura 33 a conclusão. O treinamento foi concluído após 72 minutos.

Figura 32 – Inicio do treinamento



Fonte: Autor

Figura 33 – Fim do treinamento



Fonte: Autor

Como mostra a Figura 33, várias métricas são geradas e o melhor desempenho do modelo fica gravado em: "/content/runs/detect/train/weights/best.pt". Os gráfico do primeiro treinamento está apresentado na Figura 34 e a matriz de confusão na Figura 35.

O eixo Y representa um valor percentual em formato decimal. O valor X, o número da época definida (TRAORé, 2022). Algumas definições para melhor entendimento dos gráficos:

- **box loss:** Métrica de perda que mede o quão justas as caixas delimitadoras previstas estão em relação aos objetos rotulados;
- **cls loss:** Também é uma métrica de perda, baseada em uma função de perda específica, que mede a exatidão da classificação de todas as caixas delimitadoras previstas;
- **mAP:** A Precisão Média (mAP) é usada para medir o desempenho de modelos de Visão Computacional. mAP é igual à média da métrica Precisão em todas as classes de um modelo. O mAP é medido entre 0 e 1. As outras métricas relacionadas a esta são definindo a intersecção sobre a união (IoU) de 0,50 ou de 0,50 a 0,95.
- **precision:** Métrica de quão preciso é um modelo no momento da previsão. Os verdadeiros positivos são divididos por todos os positivos que foram adivinhados;
- **recall:** Uma medida de desempenho de um sistema de previsão. *Recall* é usado para avaliar se um sistema de previsão está adivinhando o suficiente. Os verdadeiros positivos são divididos por todos os verdadeiros positivos possíveis.
- **train:** Métrica do conjunto de treinamento;
- **val:** Métrica do conjunto de validação.

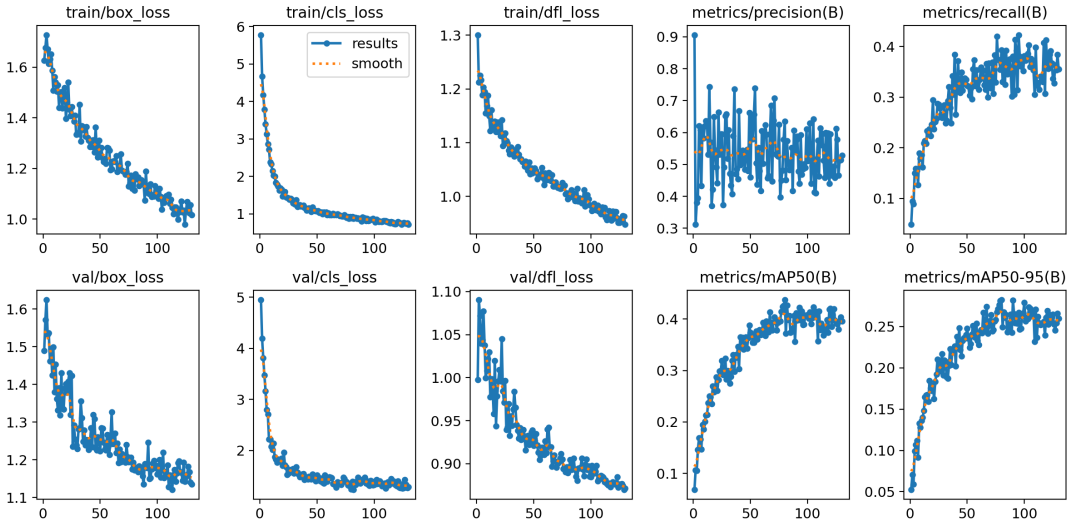
Ao analisar os gráficos da Figura 34 é possível identificar dois tipos de informações: o quanto o modelo treinado está errando (*loss*) e o quanto está acertando (*precision* e MAP). O gráfico *"train/box loss"* mostra que a perda da caixa delimitadora diminuiu de cerca de 1.6 para 1, o que significa que ao passar das épocas o modelo errou menos a projeção as *bounding boxes* nas imagens. O gráfico *"train/cls loss"* também manteu-se decrescente, o que significa que conforme o treinamento avançava, o modelo errava menos a classe proposta. Outro gráfico importante na análise é o *"train/dfl loss"* que mede a perda focal de distribuição. Essa métrica representa o quanto o modelo consegue fazer previsões menos desequilibradas e melhorar o seu desempenho, especialmente em conjuntos de dados com graves desequilíbrios de classe pois ele consegue considerar às classes raras como as mais importantes.

O gráfico *"metrics/precision(B)"* mostra que a precisão da rede variou muito durante o treinamento e que o modelo não conseguiu melhorar significamente até a última época. Isso pode ter ocorrido por uma junção de motivos como: rotulamento incorreto, imagens muito escuras e/ou não tratadas, parâmetros desajustados. Além disso a linha pontilhada que representa a validação é constante, não seguindo a mesma tendência da linha azul o que pode indicar um problema de *overfitting*, que é quando o modelo se ajustou demais aos dados de treinamento e não generaliza bem para os dados de validação. Para evitar o *overfitting*, é possível aplicar técnicas de regularização, aumento de dados, redução da complexidade do modelo ou parada antecipada.

O gráfico *"metrics/recall(B)"* mostra que o recall aumentou de 0 para cerca de 0,4, o que é um bom, pois significa que a capacidade de encontrar todos os casos relevantes dentro de um conjunto de dados foi aumentada. Por fim, o gráfico *"metrics/mAP50(B)"* mostra que o mAP50 aumentou de 0,1 para cerca de 0,4, o que significa que o modelo tem uma boa média de precisão em todos os limiares de confiança, no entanto, ou seja, uma boa medida geral de

desempenho.

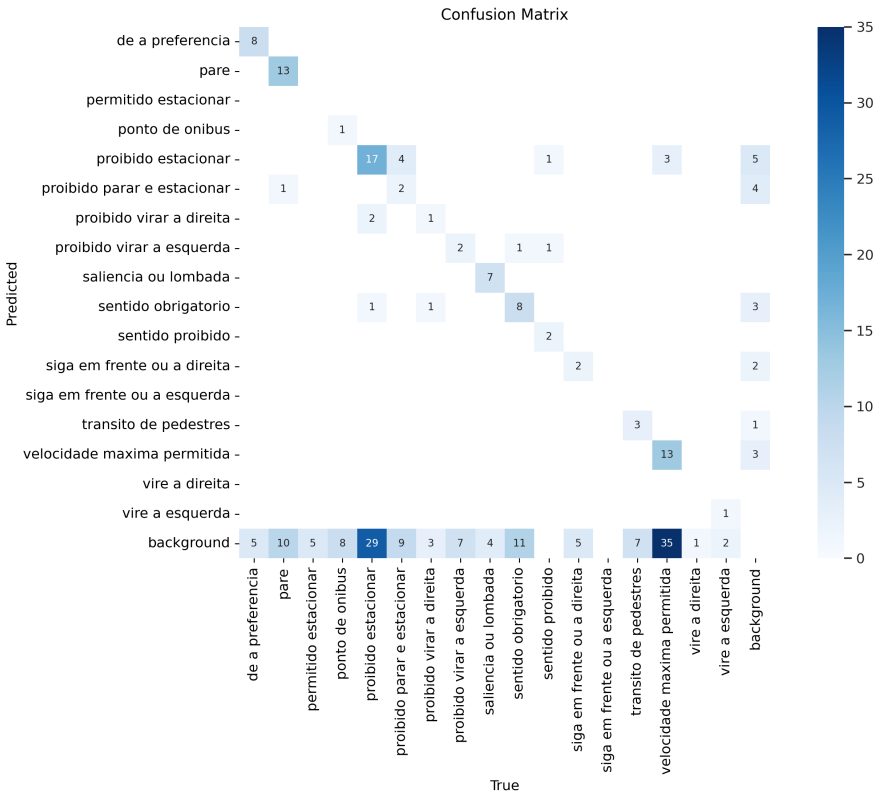
Figura 34 – Métricas e resultados



Fonte: Autor

A Matriz confusão da Figura 35 demonstra que houveram classes que não foram identificadas corretamente em nenhum momento, como a classe de "permitido estacionar"e de "siga em frente ou a esquerda". Devido a isso, um novo treinamento foi realizado, desta vez, eliminando as classes que possuíam poucas imagens e/ou baixo desempenho.

Figura 35 – Matriz de confusão do primeiro treinamento



Fonte: Autor

6.2 Segundo treinamento

Para o segundo treinamento, uma cópia do projeto anterior e algumas classes com desempenho inferior foram retiradas. Além disso, duas novas classes foram criadas: proibido ultrapassar e proibido retornar à esquerda. A Figura 36 exibe as placas rotuladas:

Figura 36 – Placas rotuladas no segundo treinamento

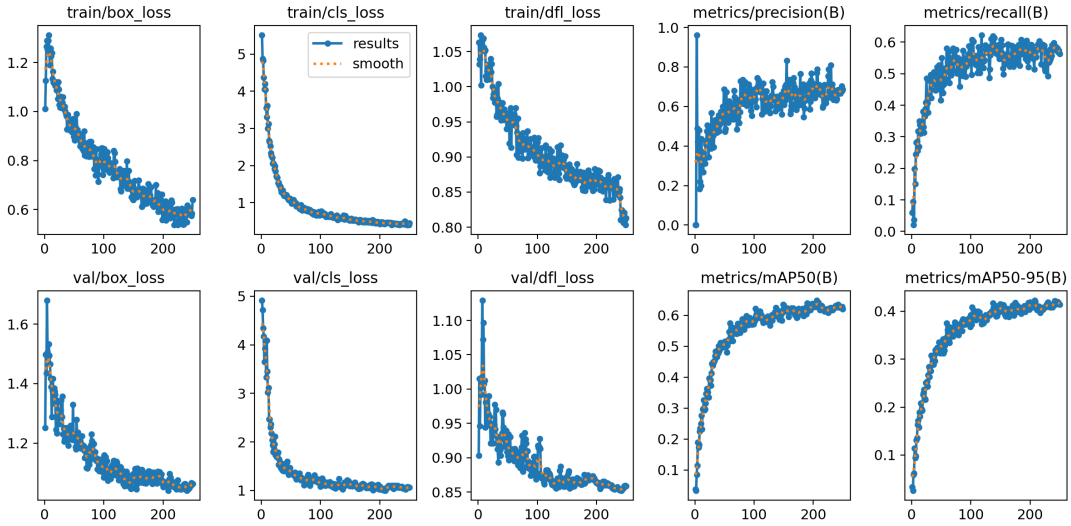
Fonte: Autor

Foram utilizadas 606 imagens, sem nenhum tipo de acréscimo, permitindo então a separação do conjunto de dados entre: 70%, 20% e 10%. [2] O treinamento levou 55 minutos para concluir e as métricas são vistas nas Figuras 37 e 38.

Em comparação as métricas da Figura 34, percebe-se que a precisão do modelo (*precision*) da Figura 37 é o maior diferencial, pois ele conseguiu manter-se crescente. Na prática, o modelo também saiu-se melhor, com uma identificação mais rápida e correta.

[2] https://app.roboflow.com/uri-erechim/tcc-caroline-paula-kolassa-fina2

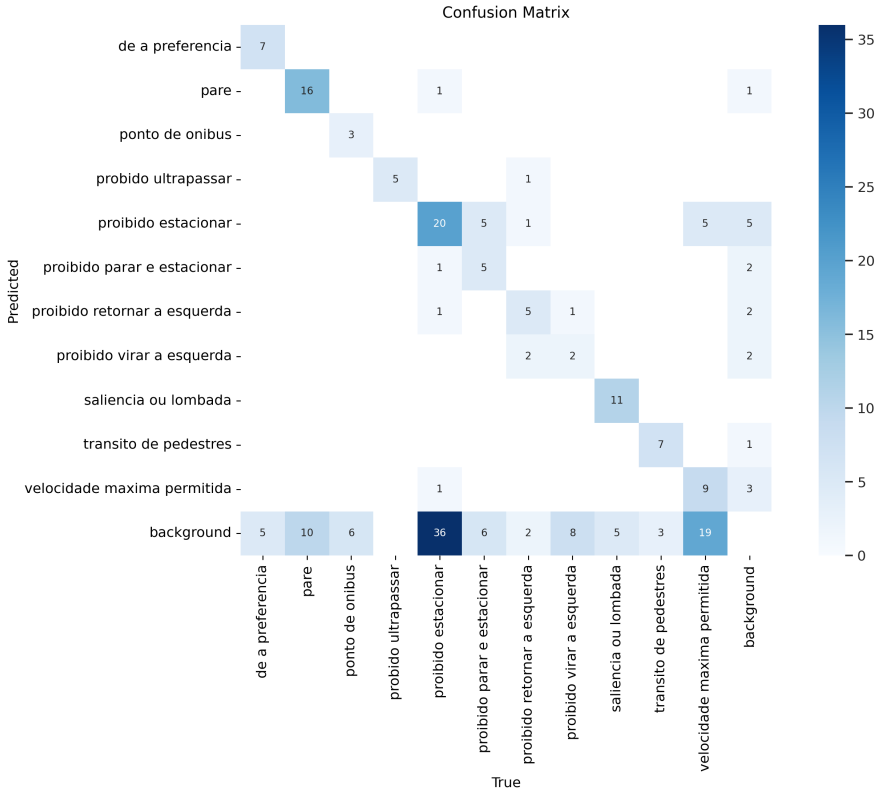
Figura 37 – Métricas e resultados



Fonte: Autor

Em relação a matriz confusão, é possível verificar que todas as placas foram identificadas corretamente por pelo menos 3 vezes. O destaque fica com a classe "proibido estacionar"que acertou 20 das 23 placas apresentadas ao modelo.

Figura 38 – Matriz de confusão do segundo treinamento



Fonte: Autor

É importante ressaltar que existem ferramentas próprias para acompanhamento de pipeline de aprendizado de máquina, desde os conjuntos de dados até os modelos de produção. Também é possível comparar experimentos, otimizar hiperparâmetros, gerenciar o ciclo de vida, monitorar o desempenho, entre outros. São exemplos desse ramo de *software*: WandB, TensorBoard, MLflow e HuggingFace.

6.3 Algoritmos para execução

Para validação da rede neural treinada 7 vídeos foram gravados, incluindo cenários climáticos e cidades diferentes e disponibilizados na plataforma do Google Drive [3].

Além disso, devido a um demo disponibilizado pelo *Roboflow*, é possível implementar a rede neural treinada via API ou executá-la diretamente em um dispositivo móvel [4]. Nas subseções a seguir, os algoritmos para execução em vídeos e em tempo real serão apresentados.

6.3.1 Execução via *webcam*

Para realizar a execução via *webcam* o código da Figura 39 deve ser executado. O resultado é visto na Figura 40. É importante salientar que os requisitos informados na documentação do YOLO devem ser instalados [5]. A execuçao é realizada através do comando: python main.py.

Figura 39 – Código da execução em tempo real

main.py 1 ×

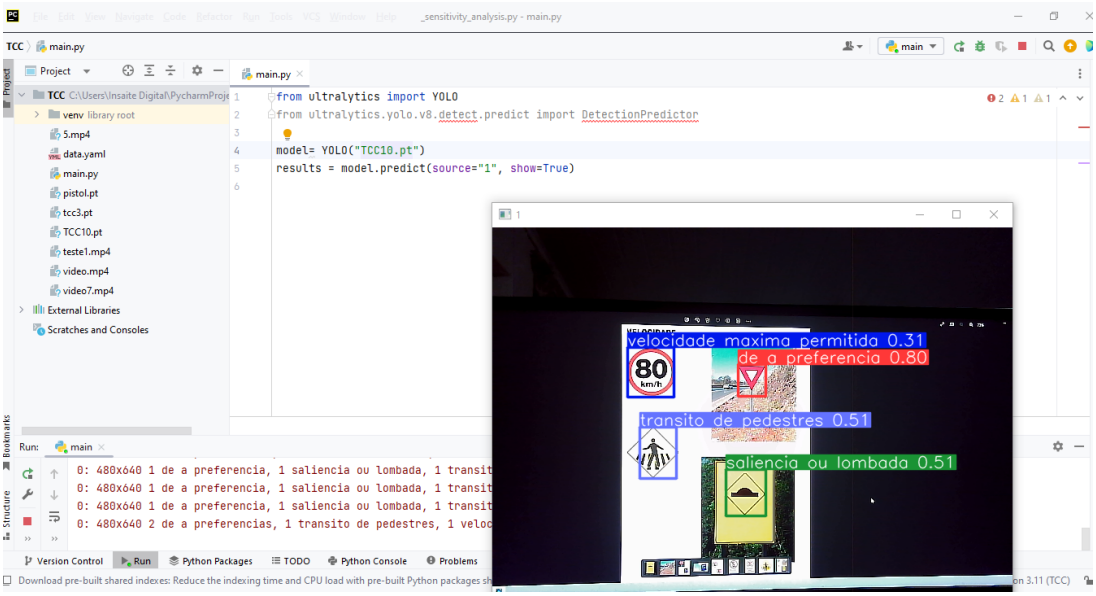
main.py > ...

```
1   from ultralytics import YOLO
2   from ultralytics.yolo.v8.detect.predict import DetectionPredictor
3
4   model= YOLO("best.pt")
5   results = model.predict(source="1", show=True)
```

Fonte: Autor

[3] https://drive.google.com/drive/u/1/folders/1O-eOYppbLEl9LimEW0WPYJuHy5AOyqHi
[4] https://app.roboflow.com/uri-erechim/tcc-caroline-paula-kolassa-final-2/1
[5] https://github.com/ultralytics/ultralytics/blob/main/requirements.txt

Figura 40 – Execução em tempo real



Fonte: Autor

6.3.2 Execução de vídeos em tempo real

Para executar os vídeos em tempo real a biblioteca cvzone deve ser instalada pelo comando: pip install cvzone. Para realizar a execução é necessário executar o código como exibido na Figura 41 e o resultado é visto através da Figura 42. A execução é realizada através do comando: python main.py. O código está disponível na plataforma GitHub. [6].

[6]https://github.com/Carolkolassa/identifica-o-das-placas

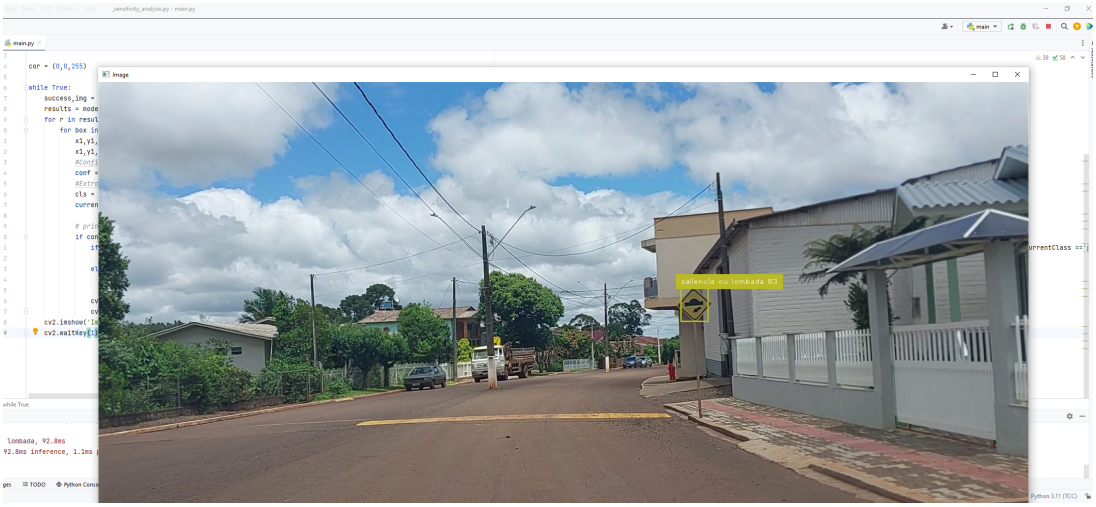
Figura 41 – Código da execução em vídeo em tempo real

main.py

```python
import cv2
import cvzone
from ultralytics import YOLO

cap = cv2.VideoCapture('video7.mp4')
modelo = YOLO('TCC10.pt')

classNames = ['de a preferencia', 'pare', 'permitido estacionar', 'ponto de onibus', 'proibido estacionar',
              'proibido parar e estacionar', 'proibido virar a direita', 'proibido virar a esquerda',
              'saliencia ou lombada', 'sentido obrigatorio', 'sentido proibido', 'siga em frente ou a direita',
              'siga em frente ou a esquerda', 'transito de pedestres', 'velocidade maxima permitida',
              'vire a direita', 'vire a esquerda']

cor = (0,0,255)

while True:
    success,img = cap.read()
    results = modelo(img,stream=True)
    for r in results:
        for box in r.boxes:
            x1,y1,x2,y2 = box.xyxy[0]
            x1,y1,x2,y2 = int(x1),int(y1),int(x2),int(y2)
            #Confiança
            conf = int(box.conf[0] * 100)
            #Extrair Classe
            cls = int(box.cls[0])
            currentClass = classNames[cls]

            # print(x,y,w,h,cls,conf)
            if conf >=30:
                if currentClass == 'de a preferencia' or currentClass =='pare' or currentClass =='permitido estacionar' or currentClass == 'proibido estacionar' or currentClass ==
                    cor = (0,0,255)
                elif currentClass == 'ponto de onibus' or 'saliencia ou lombada' or 'transito de pedestres':
                    cor = (39, 186, 186)

                cvzone.putTextRect(img,f'{classNames[cls]} {conf}',(x1,y1-5),scale=1,thickness=1,colorB=cor, colorT=(255,255,255), colorR=cor)
                cv2.rectangle(img, (x1, y1), (x2, y2), cor,2)
    cv2.imshow('Image',img)
    cv2.waitKey(1)
```

Fonte: Autor

Figura 42 – Resultado do código



Fonte: Autor

7 RESULTADOS OBTIDOS

No contexto do trabalho aqui apresentado, alguns obstáculos foram encontrados no reconhecimento, como obstrução de placas devido a vegetação e placas com baixa visibilidade em decorrência do desgaste e influências climáticas, como exibidos nas Figuras: 43 e 44.

Figura 43 – Baixa visibilidade da placa devido a plantação

Fonte: Autor

Figura 44 – Placa desgastada

Fonte: Autor

Conforme apresentado nos gráficos, a precisão média das redes ficou em torno de 69.9%, o mAP 65.3% e o Recall 58.8%. Embora considerado baixo, o desempenho em tempo real mostrou-se bastante satisfatório. Houveram alguns casos em que devido ao conjunto de dados possuir mais placas distantes, o algoritmo identifica melhor placas afastadas em comparação às próximas, como exibido na Figura 45.

Figura 45 – Exemplo de detecção de placa distante

Fonte: Autor

Além disso, alguns falsos positivos foram observados, como na Figura 46, onde a letra "o"do "Hotel"foi identificada como uma placa de Pare.

Figura 46 – Exemplo de falso positivo

Fonte: Autor

Na Figura 47 é possível visualizar uma tabela com os resultados da pesquisa de campo realizada. A Tabela utiliza os seguinte dados:

- **Verdadeiro positivo:** o modelo previu corretamente a classe positiva;
- **Falso positivo:** o modelo previu incorretamente a classe positiva quando a classe real era negativa;
- **Falso negativo:** o modelo previu incorretamente a classe negativa quando a classe real era positiva.

Figura 47 – Análise de placas em campo

	Ensolarado ☀️	Chuvoso 🌧️	Nublado ☁️	À noite 🌙
Quantidade de placas existentes no percurso:	43	30	42	35
Quantidade de placas que foram treinadas e estavam no percurso:	36	20	35	25
Falso positivo:	5	3	7	3
Falso negativo:	3	10	3	7
Verdadeiro positivo:	28	7	25	15

Fonte: Autor

O verdadeiro negativo não será utilizado, pois durante o trajeto há muitos objetos que o algoritmo não reconhece como uma placa de trânsito, tornando-o correto.

Ao analisar a tabela da Figura 47, percebe-se que nos dias chuvosos e à noite a rede treinada deixou de reconhecer várias placas, principalmente por baixa visibilidade, e isso resultou em vários falsos negativos. Já nos demais testes o falso positivo não se fez tão presente, uma vez que ao visualizar algo semelhante ao formato e cor de uma placa de trânsito o algoritmo já iniciava o reconhecimento para descobrir o tipo da placa. A placa de "dê a preferencia", por exemplo, por possuir características únicas, com ênfase no formato triangular, foi reconhecida todas as vezes. Um dos parâmetros utilizados no treinamento foi o "conf=0.50", pois assim a rede identificava apenas placas que tinham no mínimo, 50% de confiabilidade na classe.

8 CONCLUSÃO E TRABALHOS FUTUROS

O presente trabalho teve como principal objetivo o treinamento de uma rede neural capaz de identificar algumas placas de trânsito brasileiras. Devido a isso, a primeira etapa realizada foi um estudo teórico para aprendizado de conceitos importantes da área de Visão Computacional, aprendizado de máquina e redes neurais. Além disso, a pesquisa sobre trabalhos relacionados também foi realizada, buscando um diferencial: a utilização da versão 8 do *YOLO*.

O trabalho também conseguiu apresentar uma boa fundamentação teórica para entendimento dos processos, além de unificar vários *datasets* em um só, criando um próprio. A disponibilização de todos os materiais foi realizada, cumprindo assim, mais um dos objetivos do trabalho. Outro fator importante foi a pesquisa realizada sobre placas de trânsito, visando o treinamento das mais comuns na região.

Embora a precisão dos dois modelos tenha sido baixa, eles apresentaram um bom desempenho em condições favoráveis, justificando o custo elevado dos veículos autônomos atuais, que precisam de bom desempenho em todas as situações. Por esses e outros motivos, a segurança viária ainda precisa de muito desenvolvimento e tecnologia alcançável, pois embora o poder computacional tenha aumentado, a aplicação prática ainda é muito custosa.

Para trabalhos futuros, é importante aumentar o número de imagens no *dataset*, podendo expandir o número de classes, tanto com novas placas de trânsito, tanto com outros itens, como faixas de pedestre, sinaleiras entre outros, aproximando-se do cenário real. Ademais, é necessário utilizar um *software* para refinamento e acompanhamento das versões do modelo treinado, como o WandB.

Além disso, é importante buscar formas de automatizar o processo de rotulagem ou então buscar um meio de não se fazer necessário o destacamento das placas na figura, pois esse processo demanda muito tempo e torna-se cansativo. Outra questão que pode ser abordada é a melhoria da qualidade dos vídeos gravados e da *webcam* testada em tempo real. Para isso, um hardware com maior desempenho precisa ser adquirido. Com uma infraestrutura mais robusta, ainda é possível utilizar a versão *Extra Large* do algoritmo, pois por ser mais preciso, demanda mais capacidade de processamento.

A jornada de criação deste trabalho enriqueceu meu entendimento sobre o campo da Visão Computacional e reforçou o quão importante ela é para auxiliar, não só a segurança no trânsito, mas diversas outras áreas da sociedade. Portanto, a busca contínua por conhecimento e pesquisa nesta área é fundamental, considerando a presença de projetos com a capacidade de transformar e avançar a tecnologia.

REFERÊNCIAS

AGGARWAL, C. C. **Neural Networks and Deep Learning: A Textbook**. [S.l.]: Springer, 2019. Citado na página 15.

B., L. T. **Redes neurais artificiais: teoria e aplicações**. [S.l.]: Livros Técnicos e Científicos Editora, 2000. Citado na página 17.

BALLARD, D. H.; BROWN, C. M. **Computer Vision**. [S.l.]: Prentice Hall, 1982. v. 1. Citado na página 13.

BODEN, M. A. **Mind as Machine: A History of Cognitive Science**. [S.l.]: Cambridge University Press., 1988. Citado na página 13.

BROWNLEE, J. **Better Deep Learning Train Faster, Reduce Overfitting, and Make Better Predictions**. [S.l.]: Machine Learning Mastery, 2018. Citado na página 46.

CAMPOS ELDER DE OLIVEIRA RODRIGUES, E. C. C. D. H. C. **Criação e validação de uma base de dados com elementos do Trânsito Brasileiro para Veículos autônomos**. 2020. Disponível em: <https://ojs.brazilianjournals.com.br/ojs/index.php/BASR/article/view/10783/9009>. Acesso em: 22 de outubro de 2023. Citado na página 32.

CIMIRRO, J. L. D. S. **Rotulagem e treinamento do Yolo para reconhecimento de placas de trânsito brasileira**. 2022. Disponível em: <https://dspace.unipampa.edu.br/bitstream/riu/7527/1/TCC_II_Jean_Cimirro_2022_Final.pdf>. Acesso em: 22 de outubro de 2023. Citado na página 31.

CONTRAN. **Manuais Brasileiros de Sinalização de Transito**. 2022. Disponível em: <https://www.gov.br/transportes/pt-br/assuntos/transito/senatran/manuais-brasileiros-de-sinalizacao-de-transito>. Acesso em: 08 de novembro de 2023. Citado 2 vezes nas páginas 26 e 27.

DANTAS, B. O. **Rotulagem e treinamento do Yolo para reconhecimento de placas de trânsito brasileira**. 2021. Disponível em: <https://bdm.unb.br/handle/10483/30375>. Acesso em: 22 de outubro de 2023. Citado na página 31.

DANTAS, B. O. **Rotulagem e treinamento do Yolo para reconhecimento de placas de trânsito brasileiras**. 2021. Disponível em: <https://bdm.unb.br/handle/10483/30375>. Acesso em: 10 de novembro de 2023. Citado na página 37.

DETRAN.SP. **CÓDIGO DE TRÂNSITO BRASILEIRO - CTB**. 2022. Disponível em: <https://www.detran.sp.gov.br/wps/portal/portaldetran/detran/legislacao/sa-codigotransitobrasileiro/36f3521d-ea6d-4f36-a989-11ab4a1ff07a#:~:text=O%20tr%C3%A2nsito%20brasileiro%20%C3%A9%20regulamentado,de%20Tr%C3%A2nsito%20Brasileiro%20(CTB).> Acesso em: 14 de setembro de 2023. Citado na página 26.

DWYER, B. **Quando devo orientar automaticamente minhas imagens?** 2020. Disponível em: <https://blog.roboflow.com/exif-auto-orientation/>. Acesso em: 15 de outubro de 2023. Citado na página 43.

FERREIRA, R. **Deep learning**. [S.l.]: Editora Saraiva, 2021. Citado 4 vezes nas páginas 17, 18, 19 e 20.

FRANçA, G. B. B. B. e C. A. **Classificação de placas de trânsito com redes neurais para automação de veículos**. 2022. Disponível em: <https://repositorio.ufscar.br/handle/ufscar/15692>. Acesso em: 22 de outubro de 2023. Citado na página 32.

GERON, A. **Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. [S.l.]: O'Reilly Media, 2019. Citado 2 vezes nas páginas 18 e 19.

GOOGLE. **Google Colab**. 2023. Disponível em: <https://colab.research.google.com/>. Acesso em: 19 de outubro de 2023. Citado na página 34.

GOOGLE. **Google Drive**. 2023. Disponível em: <https://www.google.com/intl/pt-BR/drive/>. Acesso em: 19 de outubro de 2023. Citado na página 34.

IBGE. **Frota de veículos**. 2022. Disponível em: <https://cidades.ibge.gov.br/brasil/pesquisa/22/28120?tipo=grafico&indicador=28122>. Acesso em: 06 de novembro de 2023. Citado na página 12.

IBM. **O que são redes neurais?** 2023. Disponível em: <https://www.ibm.com/br-pt/topics/neural-networks>. Acesso em: 11 de setembro de 2023. Citado na página 20.

IBM. **O que é Aprendizado não Supervisionado?** 2023. Disponível em: <https://www.ibm.com/br-pt/topics/unsupervised-learning>. Acesso em: 22 de outubro de 2023. Citado na página 16.

JOCHER, G. **Ultralytics**. 2023. Disponível em: <https://docs.ultralytics.com/>. Acesso em: 07 de novembro de 2023. Citado na página 35.

JUNIOR, E. **Projeto de Visão Computacional de Placas de Trânsito**. 2021. Disponível em: <https://universe.roboflow.com/edward-junior/placas-de-transito-7po0k/dataset/1>. Acesso em: 14 de novembro de 2023. Citado na página 37.

KERAS. **Documentação Oficial**. 2020. Disponível em: <https://keras.io/examples/vision/visualizing_what_convnets_learn/#visualize-the-first-64-filters-in-the-target-layer>. Acesso em: 14 de setembro de 2023. Citado na página 21.

MEDEIROS, D. da Silva de. **Desenvolvimento de uma base de imagens de placas de sinalização**. 2018. Disponível em: <https://wiki.sj.ifsc.edu.br/images/d/df/DiegoMedeiros-Projeto-13-2018-CSJ-Placas.pdf>. Acesso em: 08 de novembro de 2023. Citado na página 37.

NASCIMENTO, G. L. do. **CLASSIFICAÇÃO DE PLACAS DE TRÂNSITO UTILIZANDO REDES NEURAIS CONVOLUCIONAIS**. 2021. Disponível em: <https://repositorio.pucgoias.edu.br/jspui/bitstream/123456789/3542/1/TCC-Guilherme-2021.pdf>. Acesso em: 10 de novembro de 2023. Citado na página 37.

NETWORKS, T. **Carros autônomos: serão eles o novo "padrão" da indústria?** 2023. Disponível em: <https://www.sae.org/standards/content/j3016_201806/>. Acesso em: 09 de setembro de 2023. Citado na página 28.

OPENCV. **Documentação Oficial**. 2023. Disponível em: <https://opencv.org/about/>. Acesso em: 15 de outubro de 2023. Citado na página 35.

OVERLEAF. **Overleaf**. 2023. Disponível em: <https://www.overleaf.com/>. Acesso em: 19 de setembro de 2023. Citado na página 34.

PAIVA, F. A. P. D. **Introdução a Python com aplicações de Sistemas Operacionais**. [S.l.]: IFRN, 2020. Citado na página 35.

PASSOS, B. T. **O mundo do ponto de vista das Redes Neurais Convolucionais (RNCs)**. 2021. Disponível em: <https://ateliware.com/blog/redes-neurais-convolucionais>. Acesso em: 14 de setembro de 2023. Citado 2 vezes nas páginas 20 e 21.

REDMON, J. **You Only Look Once: Unified, Real-Time Object Detection**. 2016. Disponível em: <https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf>. Acesso em: 07 de novembro de 2023. Citado 2 vezes nas páginas 22 e 23.

ROBOFLOW. **Documentação Oficial**. 2023. Disponível em: <https://roboflow.com/>. Acesso em: 15 de outubro de 2023. Citado na página 34.

SAE. **Documentação Oficial**. 2018. Disponível em: <https://www.sae.org/standards/content/j3016_201806/>. Acesso em: 09 de setembro de 2023. Citado na página 30.

SAGE, A. P. E. **Concise Encyclopedia of Information Processing in Systems and Organizations**. [S.l.]: New York: Pergamon, 1990. Citado na página 15.

SOLAWETS, J. **What is Mean Average Precision (mAP) in Object Detection?** 2020. Disponível em: <https://blog.roboflow.com/mean-average-precision/>. Acesso em: 08 de novembro de 2023. Citado 2 vezes nas páginas 21 e 22.

S.R.O, J. **PyCharm**. 2023. Disponível em: <https://www.jetbrains.com/pt-br/pycharm/>. Acesso em: 19 de outubro de 2023. Citado na página 35.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. [S.l.]: Editora Springer, 2021. Citado 2 vezes nas páginas 13 e 14.

SÁ, Y. V. d. A. **Desenvolvimento de aplicações IA: robótica, imagem e visão computacional**. [S.l.]: Editora Saraiva, 2021. Citado 2 vezes nas páginas 12 e 13.

TRANSPORTES, M. dos. **Estudo aponta que mais de 50trânsito são causados por falhas humanas**. 2018. Disponível em: <https://www.gov.br/transportes/pt-br/pt-br/assuntos/noticias/ultimas-noticias/estudo-aponta-que-mais-de-50-dos-acidentes-de-transito-sao-causados-por-falhas-humanas765>. Acesso em: 06 de novembro de 2023. Citado na página 12.

TRAORé, M. **Roboflow Train: Understanding Training Graphs**. 2022. Disponível em: <https://help.roboflow.com/faqs/roboflow-train-understanding-training-graphs?from_search=131044134>. Acesso em: 08 de novembro de 2023. Citado na página 47.

ULTRALYTICS. **Documentação Oficial**. 2022. Disponível em: <https://github.com/ultralytics/ultralytics>. Acesso em: 14 de setembro de 2023. Citado 2 vezes nas páginas 23 e 24.

ULTRALYTICS. **Documentação Oficial**. 2023. Disponível em: <https://docs.ultralytics.com/datasets/detect/coco/>. Acesso em: 29 de setembro de 2023. Citado na página 24.

V.N., K. S. S. Detecção e classificação em tempo real de sinais de trânsito com base no algoritmo yolo versão 3. **ITMO University**, v. 20, n. 3, p. 418–424, 2020. Citado na página 33.

WEBER, M. **Where to? A history of autonomous vehicles. Computer History Museum**. 2014. Disponível em: <http://www.computerhistory.org/atchm/where-to-a-history-ofautonomous-vehicles/>. Acesso em: 06 de novembro de 2023. Citado na página 28.

ZHANG, A. et al. **Dive into Deep Learning**. [S.l.: s.n.], 2020. Https://d2l.ai. Citado 3 vezes nas páginas 15, 16 e 17.

ZHAO, Z.-Q. **Object Detection with Deep Learning: A Review**. 2019. Disponível em: <https://arxiv.org/abs/1807.05511>. Acesso em: 08 de novembro de 2023. Citado 2 vezes nas páginas 12 e 23.