Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Rotulagem e treinamento do Yolo para reconhecimento de placas de trânsito brasileiras

Autor: Bruno Oliveira Dantas
Orientador: Prof. Giovanni Almeida Santos

Brasília, DF
2021

Bruno Oliveira Dantas

Rotulagem e treinamento do Yolo para reconhecimento de placas de trânsito brasileiras

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Giovanni Almeida Santos

Brasília, DF

2021

Bruno Oliveira Dantas

Rotulagem e treinamento do Yolo para reconhecimento de placas de trânsito brasileiras/ Bruno Oliveira Dantas. – Brasília, DF, 2021-

104 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Giovanni Almeida Santos

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2021.

1. Yolo v3. 2. Veículo autônomo. I. Prof. Giovanni Almeida Santos. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Rotulagem e treinamento do Yolo para reconhecimento de placas de trânsito brasileiras

CDU 02:141:005.6

Bruno Oliveira Dantas

Rotulagem e treinamento do Yolo para reconhecimento de placas de trânsito brasileiras

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 13 de Maio de 2021:

Prof. Giovanni Almeida Santos
Orientador

Prof. Dr. Maurício Serrano
Examinador

Profa. Dra. Milene Serrano
Examinadora

Brasília, DF
2021

Este trabalho é dedicado a todos que buscam seus sonhos incessantemente e independentemente dos desafios e dificuldades que encontram durante o caminho.

Agradecimentos

Agradeço primeiramente a Deus por estar sempre me dando forças para nunca desistir. Em seguida, aos meus pais, Ivan e Vanessa, que sempre prezaram pela família, e ao apoio incondicional que proporcionam. Por falar em amor incondicional, eu agradeço à minha avó Lurdes, que é uma das pessoas mais importantes neste mundo para mim.

Agradeço muito a todos os meus amigos, em especial aos que tive a honra de conhecer durante a minha jornada na Faculdade. Não irei citar nomes. Caso contrário, os agradecimentos teriam várias páginas. Mas todos são especiais para mim, e tornam meus dias mais felizes. Irei levar grande parte deles para o resto da vida. Agradeço demais ao meu irmão, Rodrigo, e ao meu grande amigo, Lucas, que me acompanharam na caçada às placas de trânsito. Foi uma bela aventura. Agradeço à minha namorada, Natália, por ouvir minhas lamentações e deixar a minha vida infinitamente mais feliz.

Agradeço imensamente ao meu orientador, Giovanni Almeida, por me dar a oportunidade de ser seu orientando e por apontar os melhores direcionamentos. Deixo um agradecimento especial, para todos os professores, que de alguma forma enriqueceram a minha bagagem de conhecimento. E também obrigado a todos os funcionários da Faculdade, que trabalham arduamente todos os dias para proporcionar o melhor local possível para os alunos poderem estudar.

Hoje, sou quem sou por conta de todas as pessoas que passaram por minha vida. Todos contribuíram de alguma forma para que eu possa ter chegado até aqui. Agradeço de coração a todos.

Resumo

Os veículos autônomos estão cada vez mais presentes na área de estudos de inteligência artificial, porque a temática ainda possui muitos desafios a serem solucionados. Em particular, há muita complexidade na implementação de automóveis autônomos para que possam ser utilizados de forma popular nas ruas e avenidas. Existem diversas problemáticas encontradas, por exemplo: a legislação de trânsito, a identificação de sinalização de trânsito, o alto volume de veículos e pessoas. Essas são algumas das variáveis imprevisíveis que podem ocorrer. Pensando nisso, este trabalho tem o intuito de contribuir no tema de veículos autônomos, principalmente na realidade brasileira. O objetivo, então, tem sido construir um processo replicável de rotulagem e treinamento do YOLO para detecção de placas de trânsito brasileiras. Neste trabalho, foi treinado um modelo com 12 placas de trânsito que são mais comumente encontradas em vias públicas urbanas. O modelo gerado teve como resultado geral, uma "acurácia = 86%"e uma "precisão = 100%". E ainda, o modelo foi submetido a diversos cenários, que envolve a iluminação, o clima e o ambiente em que as placas de trânsito estão inseridas.

Palavras-chave: Yolov3. Rotulagem e treinamento. Detecção de placas de trânsito. Placas de trânsito brasileiras. Veículo autônomo.

Abstract

Autonomous vehicles are increasingly present in the field of artificial intelligence studies, because the subject still has many challenges to be solved. In particular, there is a lot of complexity in the implementation of autonomous cars so that it can be used popularly in the streets and avenues. There are several problems encountered, for example: traffic legislation, the identification of traffic signs, the high volume of vehicles and people. These are some unpredictable variables that can occur. With this in mind, this work aims to contribute to the theme of autonomous vehicles, mainly in the Brazilian reality. The objective, then, is to build a replicable YOLO labeling and training process for detecting Brazilian traffic signs. In this work, a model was trained with 12 traffic signs that are most commonly found on urban public roads. The generated model had as a general result, an " accuracy = 86 % " and an " accuracy = 100 % ". Furthermore, the model was subjected to several scenarios, which involve lighting, the climate and the environment in which the traffic signs are inserted.

Key-words: Yolov3. labeling and training. Detection of traffic signs. Brazilian traffic signs. Autonomous vehicle.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CPU	*Central Process Unit*, ou Unidade Central de Processamento
GPU	*Graphics Processing Unit*, ou a Unidade de Processamento Gráfico
POC	Prova de Conceito
TCC	Trabalho de Conclusão de Curso
YOLO	*You Only look Once*
TP	*True positive* (Verdadeiro Positivo)
TN	*True negative* (Verdadeiro Negativo)
FP	*False positive* (Falso Positivo)
FN	*False negative* (Falso Negativo)

Sumário

Parte I

Introdução

1 Introdução

Os veículos autônomos têm atraído bastante atenção nos últimos anos devido às possibilidades que esse tema pode promover para a sociedade, tais como, utilizar veículos que atuam sem a direção humana. Em razão disto, o olhar de vários pesquisadores voltou-se para esse assunto, pela infinidade de desafios que a temática de veículos autônomos gera, principalmente na área de Inteligência Artificial (OSóRIO; HEINEN; FORTES, 2001).

O sistema de automação já existe em veículos autônomos aéreos, aquáticos, ferroviários e são funcionais. Entretanto, adentrando o contexto de automação em automóveis convencionais, as dificuldades dessa implementação são maiores que a de outros veículos; visto que há complexidade em vários pontos, por exemplo: nas regras de trânsito, na quantidade de veículos em circulação que devem ser coordenados harmoniosamente, tal como a variedade de vias, avenidas e possíveis rotas (SANTOS, 2017).

A questão, ainda dita pelo autor, é que existem várias situações adversas e imprevisíveis que podem ocorrer, e riscos elevados à segurança dos pedestres e passageiros. Dessa forma, existem diversos estudos que buscam viabilizar a utilização de veículos autônomos no cotidiano das pessoas. No Brasil, tais estudos ainda são escassos sobre o tema. Visando isto, esse trabalho tem como intenção contribuir para que a realidade de pesquisa brasileira sobre o contexto de veículos autônomos avance cada vez mais.

Especificamente neste trabalho, optou-se, devido à ampla gama de possibilidades, que será abordada uma pequena parte desse tema. Dessa forma, o presente trabalho tem como foco o mapeamento do processo de identificação e interpretação de placas de trânsito brasileiras para que, assim, seja possível classificar placas de trânsito. O resultado é possibilitar a sua utilização em um veículo autônomo para a detecção dessas placas.

Para tornar isso viável, dentre todos os modelos e sistemas que têm capacidade de detecção de objetos no *frame* de uma imagem, foi selecionado para este trabalho o YOLOv3. Sendo assim, é a partir dele que será desenvolvido o processo de rotulagem e treinamento de um *dataset* de placas de trânsito brasileiras. A questão da pesquisa relatada neste trabalho é:

Como detectar e classificar placas de trânsito brasileiras utilizando o YOLOv3, e ainda documentar o processo de rotulagem e treinamento?

1.1 Objetivos do trabalho

O objetivo geral deste trabalho é definir e documentar um processo de rotulagem e treinamento utilizando o sistema de identificação de objetos, YOLOv3, para realizar o

reconhecimento de placas de trânsito brasileiras. E também tem como objetivos específicos:

1. Definir um processo de rotulagem e treinamento do YOLOv3 para que outros pesquisadores sejam capazes de recriar as etapas da pesquisa e, assim, conseguir detectar e classificar placas de trânsito brasileiras.
2. Determinar um conjunto de métricas que poderão indicar a qualidade de detecção e classificação das placas de trânsito brasileiras.
3. Executar um conjunto de simulações em vídeo para identificar a capacidade de detecção e classificação das placas de trânsito brasileiras em diferentes contextos.
4. Verificar a distância máxima que o modelo de treinamento detecta, identifica e classifica as placas de trânsito brasileiras.

1.2 Estrutura do trabalho

Este trabalho é constituído por cinco capítulos, sendo eles:

- **Capítulo 1 - Introdução:** o presente capítulo tem como principal objetivo contextualizar o tema da pesquisa, a justificativa para o projeto, a questão central e objetivos gerais e específicos;
- **Capítulo 2 - Fundamentos Teóricos:** este capítulo apresenta conceitos importantes que foram necessários para ter um bom entendimento sobre o assunto definido ao decorrer da pesquisa. O YOLOv3, a matriz de confusão e placas de trânsito são abordados nesse capítulo.
- **Capítulo 3 - Metodologia:** é a estratégia estruturada que orientou a pesquisa para que, dessa forma, os objetivos sejam alcançados.
- **Capítulo 4 - Resultados:** apresenta os resultados e como eles foram obtidos através de uma análise do processo de desenvolvimento da pesquisa.
- **Capítulo 5 - Conclusão:** evidencia a conclusão referente aos resultados obtidos, assim como as considerações para trabalhos futuros.

2 Fundamentos Teóricos

Neste capítulo, são apresentados os conceitos que fundamentam este trabalho. Dessa forma, será descrita brevemente a teoria relacionada ao funcionamento do YOLOv3, da Matriz de Confusão e placas de trânsito, visando, assim, facilitar o entendimento desses termos que serão utilizados ao longo deste trabalho.

2.1 YOLOv3

O YOLOv3 (You Only Look Once; pt: Você somente olha uma vez) é um algoritmo desenhado e proposto por Redmon et al. (2016), que tem como objetivo e finalidade classificar e detectar objetos, sendo possível, assim, obter a posição e a categoria atribuída ao objeto encontrado na imagem em que foi feita a predição.

Mas o que seria a detecção de objetos? De acordo com Mantripragada (2020), a detecção de objetos é uma tecnologia que engloba duas tarefas: a de classificação e a de localização de objetos. E isso pode ser usado em imagens estáticas ou mesmo em tempo real em vídeos.

O YOLOv3 é uma melhoria em relação às redes de detecção YOLO anteriores. Comparando-o com suas versões anteriores, ele apresenta algumas melhorias, como: detecção em várias escalas, rede de extração de recursos mais forte e algumas alterações na função de perda (LI, 2019). Além disso, ele é mais robusto, mas também é mais lento do que as versões anteriores (REDMON; FARHADI, 2017). Ainda assim, o YOLOv3 é considerado um dos mais poderosos algoritmos de detecção de objetos em tempo real (MANTRIPRAGADA, 2020).

Dito isso, a escolha do Yolov3 para o presente TCC deve-se às diversas qualidades que ele oferece para a a detecção de objetos. Inclusive, ele se mostra superior a outros sistemas de detecção. De acordo com a pesquisa realizada pelo Redmon et al. (2016), ela mostra um comparativo entre o YOLO e outros sistemas de detecção, por exemplo, “RetinaNet-50” e “RetinaNet-101”. Como resultado, o YOLO é executado significativamente mais rápido do que os outros modelos de detecção, que possuem um desempenho comparável.

O algoritmo de detecção do YOLOv3 utiliza *bounding boxes*, que são marcações nas imagens para fazer as análises da detecção e classificação de objetos. Um exemplo de uma aplicação da *bounding box* pode ser visto na Figura 1. Nessa figura, por exemplo, há 3 objetos (uma pessoa e dois animais) para cada *frame*. O YOLO tentará ajustar e localizar a posição e o reconhecimento do objeto.

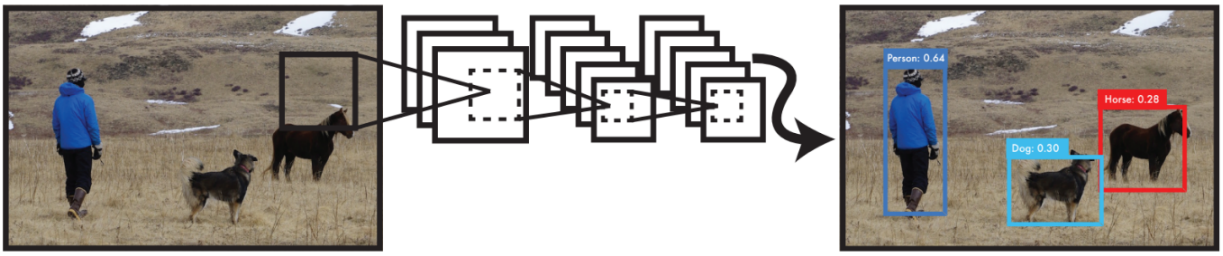


Figura 1 – Exemplo da aplicação de *bounding boxes* em uma imagem.

Fonte: Adaptado do Redmon et al. (2016)

Adentrando um pouco mais sobre como ocorre o funcionamento do YOLOv3, o autor Li (2019) criou um diagrama, pode ser visto na Figura 2, que simplifica a arquitetura do YOLOv3. Basicamente, o autor explica que a arquitetura pode ser dividida em dois componentes principais: **Extrator de Recurso e Detector de Recurso**. Quando uma nova imagem chega, ela passa primeiro pelo extrator de recursos para que possa obter informações de escalas das imagens. Em seguida, as informações são direcionadas para o detector, é nesse momento em que são geradas as *bounding boxes* e a identificação da classe.

Figura 2 – Arquitetura de rede do YOLOv3.

Fonte: Adaptado do Li (2019)

O **extrator de recursos** utilizado pelo YOLOv3 é chamado de *Darknet*[1], que também foi desenvolvido por Redmon (2013–2016). O *Darknet* é uma estrutura de rede neural de código aberto[2] que foi construída a partir das linguagens de programação "C" e "CUDA". Essa rede neural é rápida e de fácil instalação, e oferece suporte para computação de CPU[3] e de GPU[4].

[1] <https://pjreddie.com/darknet/>.Acessado em: 03 de Maio de 2021

[2] Significa que o código fonte é disponibilizado para que qualquer pessoa possa utilizar. Possui uma licença de código aberto, no qual recebe-se o direito de estudar, modificar e distribuir o software de graça para qualquer um e para qualquer finalidade.

[3] Sigla para *Central Process Unit*, ou Unidade Central de Processamento.

[4] Sigla para *Graphics Processing Unit*, ou a Unidade de Processamento Gráfico.

Sendo um pouco mais específico, o YOLOv3 faz uso do *Darknet-53*. Essa rede utiliza 53 camadas de convolução, onde é construída com camadas de convolução "3x3" e "1x1" consecutivamente. A arquitetura apresentada para o *Darknet-53* pode ser visualizada na Figura 3. (REDMON; FARHADI, 2018)

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figura 3 – Arquitetura Darknet-53.

Fonte: Redmon e Farhadi (2018)

Por sua vez, o **detector de recursos** é alimentado pela saída extrator de recursos, que após a aplicação de escala das imagens, ele passa a detecção dos objetos dentro da imagem aplicando, assim, as *bounding boxes* juntamente com a identificação da classe (MANTRIPRAGADA, 2020).

Dessa forma, para melhor exemplificar o funcionamento do YOLOv3, de uma maneira básica, ele pode ser acompanhado por meio da Figura 4. Nota-se que essa figura possui numerações. Cada etapa será apresentada.

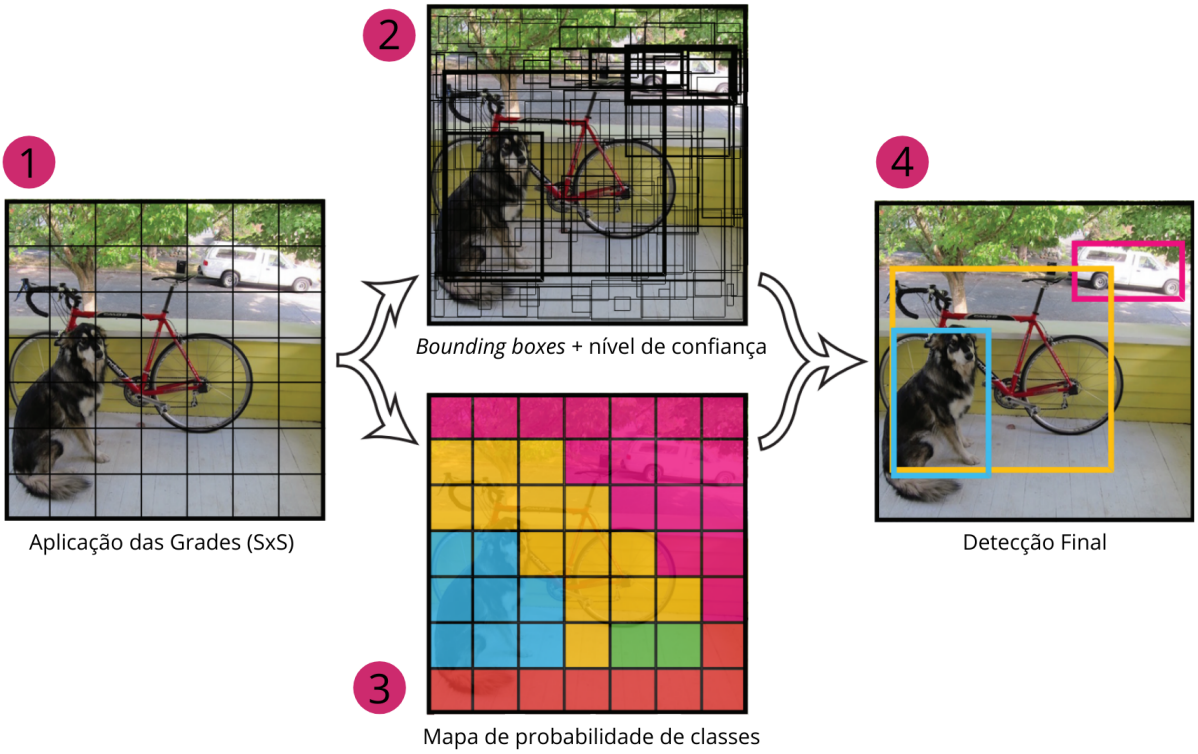


Figura 4 – Exemplo de como funciona o YOLOv3.

Fonte: Adaptado do Redmon et al. (2016)

1. O primeiro processo que o YOLOv3 realiza é selecionar a imagem na qual será feita a predição, e dividi-la em vários *grids*[5]. Por padrão, a imagem é dividida em "7 x 7", mas Redmon e Farhadi (2018) explicam que a rede neural pode ser adaptada para que ela divida a imagem em "S x S"por exemplo. Sendo assim, qualquer dimensão pode ser aplicada. Porém quanto mais divisões, mais complexa se tornará a rede.

2. Nesse momento, ele começa a aplicar as *bounding boxes* e tentar encontrar o objeto na imagem. Então, para cada *grid*, ele tentará fazer várias predições utilizando inúmeras *bounding boxes* ao mesmo tempo. Finalmente, ele entregará apenas a *bounding box* que tiver a maior acurácia, ou seja, a maior taxa de acerto.

3. Por outro lado, essa etapa tentará calcular a probabilidade do objeto ser realmente o que ele está esperando. Por exemplo, ele irá tentar predizer se o cachorro apresentado na figura realmente é um cachorro.

4. Por fim, é feita a detecção final dos objetos, onde o YOLO mantém apenas as *bounding boxes* que tiveram a maior acurácia, e com isso a classificação do rótulo do objeto.

O YOLOv3 ainda possui diversas informações ao seu respeito, mas os conceitos básicos que serão inferidos durante o desenvolvimento deste TCC foram abordados.

[5] Significa "grades" em português.

2.2 Matriz de Confusão

A matriz de confusão, segundo Prina e Trentin (2015), é uma forma de representação da qualidade obtida de uma classificação digital de imagem, no caso deste trabalho. Dessa forma, a matriz de confusão é expressa por meio da correlação de informações, sendo elas os dados de referência reais e os dados da predição. Assim, a matriz resume o desempenho de classificação de um classificador com relação a alguns dados de teste (TING, 2011).

De acordo com Fawcett (2005), a matriz de confusão é uma matriz com dimensões "MxM", em que "M"é o número de classes, as linhas representam os valores reais, e as colunas os valores preditos. A Figura 5 apresenta um exemplo genérico de matriz de confusão. Observando a imagem, é possível ver a presença dos erros e acertos, são eles: verdadeiro positivo, falso negativo, falso positivo e verdadeiro negativo. Ainda conforme o autor citado, serão apresentados brevemente estes conceitos.

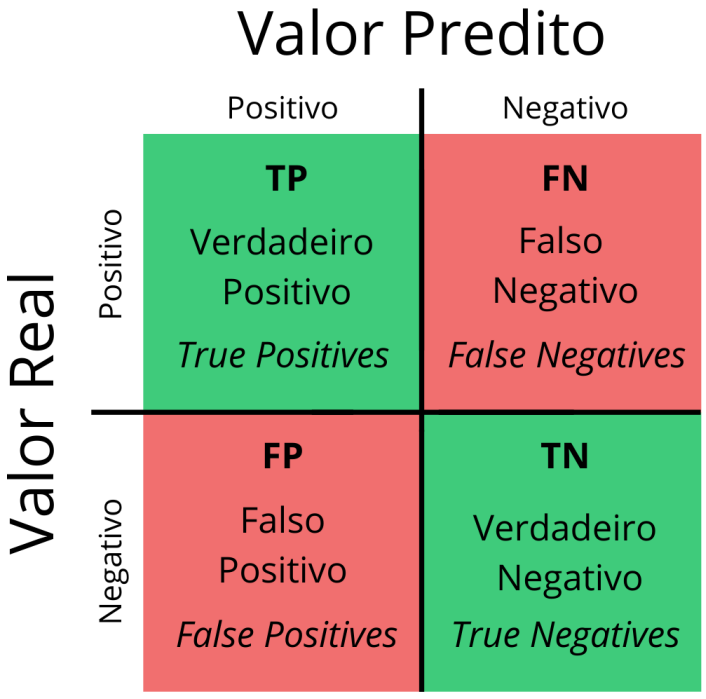


Figura 5 – Exemplo de matriz de confusão.

Fonte: Autor

- **Verdadeiro positivo (true positive — TP):** é quando o valor previsto corresponde ao valor real; e o valor real foi previsto e o modelo previu um valor positivo. Por exemplo, é quando o YOLO prediz que na imagem existe uma placa de trânsito brasileira, e a imagem realmente possui uma placa de trânsito brasileira.

- **Verdadeiro negativo (true negative — TN):** é quando o valor previsto corresponde ao valor real; e o valor real era negativo e o modelo previu um valor negativo. Um exemplo básico disso é quando o YOLO prevê que na imagem não existe placa de trânsito brasileira alguma, e a imagem realmente não possui uma placa de trânsito.
- **Falso positivo (false positive — FP):** o valor previsto foi falsamente previsto; o valor real era negativo, mas o modelo previu um valor positivo. Nesse exemplo, o YOLO prevê um objeto qualquer como uma placa de trânsito brasileira.
- **Falso negativo (false negative — FN):** o valor previsto foi falsamente previsto; o valor real foi positivo, mas o modelo previu um valor negativo. Por exemplo, o YOLO não consegue identificar uma placa de trânsito brasileira na imagem, mas ali existe uma placa de trânsito brasileira.

Com base nesses conceitos apresentados, é possível realizar várias métricas de avaliação de modelos de classificação. Para esse trabalho, foram selecionados três tipos de métrica: **acurácia**, **precisão**, ***recall***.

- **Acurácia:** a acurácia relata o percentual de acertos que um modelo teve. Dessa forma, pode ser definida como a proporção de previsões verdadeiras de uma categoria em relação a todas as outras previsões feitas. Sendo assim, calcula-se utilizando a equação:

$$Acurácia = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.1}$$

- **Precisão:** pode-se definir a precisão de um modelo como a proporção de predições corretas de uma categoria em relação a todas as previsões feitas dessa categoria. Desse modo, calcula-se a precisão utilizando a seguinte equação:

$$Precisão = \frac{TP}{TP + FP} \tag{2.2}$$

- **Recall:** a medida de *recall* de um modelo é definida como a proporção de previsões da categoria. Sendo assim, *verdadeiros positivos* em relação à soma dos *verdadeiros positivos* com os *falsos negativos*. Assim, forma-se a equação do *recall*:

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

2.3 Placas de Trânsito

As placas de trânsito têm uma grande importância na sociedade. Elas servem para orientar os condutores sobre as regras de tráfego, informar as condições da via e alertar sobre proibições, restrições e obrigações, dentre outras funções. No meio de veículos autônomos, há uma dependência ainda maior, pois o veículo precisa dessas sinalizações para poder agir com base nas informações de cada placa de trânsito coletada nas avenidas, ruas e vias.

Quando falamos de placas de trânsito no mundo, estamos tratando de uma diversidade enorme, visto que cada país possui sua própria cultura, leis e legislação de trânsito distintas das quais são conhecidas no Brasil. E em algumas situações, mesmo em um país existem legislações específicas para cada estado, por exemplo, nos Estados Unidos, a legislação tem variações dependendo da região de acordo com o site[6] do governo americano.

Dessa forma, as placas de trânsito não são globalizadas, e cada parte do mundo tem as suas peculiaridades. Como podemos ver na Figura 6, as diferenças ficam mais evidentes, apesar de expressarem a mesma informação, que é "proibido estacionar".

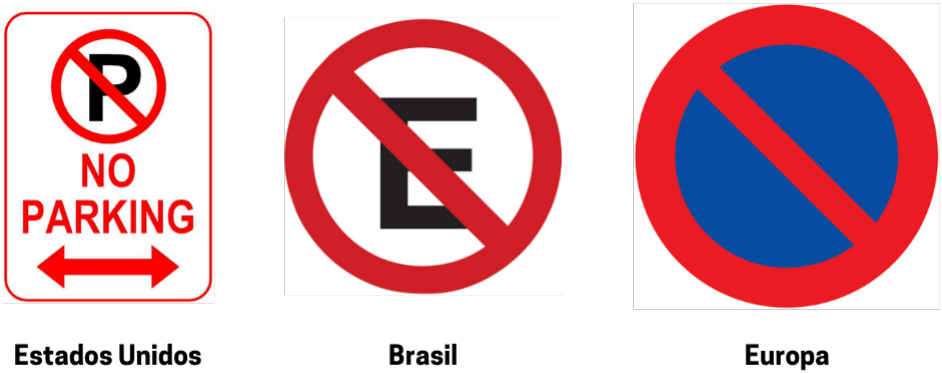


Figura 6 – Exemplo da placa de "proibido estacionar" nos Estados Unidos, Brasil e Europa.

Fonte: Autor

Sendo assim, por serem tão diferentes, é difícil para que um modelo consiga identificar placas de trânsito de diferentes países. Ainda assim, existem muitas pesquisas voltadas para detecções de placas de trânsito, porém grande parte é destinada para placas de trânsito estrangeiras, assim como nas pesquisas de Luyao et al. (2020) e Thipsanthia, Chamchong e Songram (2019), que são voltadas para placas chinesas e tailandesas, respectivamente.

[6] <https://www.usa.gov/motor-vehicle-services>

Tendo em vista essas informações, esse trabalho tende a enriquecer as pesquisas brasileiras voltadas para detecção e identificação de placas de trânsito brasileiras no contexto de veículos autônomos.

2.3.1 Placas de Trânsito Brasileiras

No Brasil, quando trata-se de trânsito, as leis são regidas pelo Código de Trânsito Brasileiro[7] (CTB). De acordo com a Resolução Nº 160/2004 do CTB as placas de sinalização de trânsito têm três categorias (BRASIL, 1997). São elas:

- **Sinalização de regulamentação:** tem por finalidade informar aos usuários as condições, proibições, obrigações ou restrições no uso das vias.
- **Sinalização de advertência:** tem por finalidade alertar os usuários da via para condições potencialmente perigosas, indicando sua natureza.
- **Sinalização de indicação:** tem por finalidade identificar as vias e os locais de interesse, bem como orientar condutores de veículos quanto aos percursos, os destinos, as distâncias e os serviços auxiliares, podendo também ter como função a educação do usuário. Essa categoria possui ainda cinco sub-categorias: placas de identificação, de orientação de destino, educativas, de serviços auxiliares e de atrativos turísticos.

Do ponto de vista de um veículo autônomo, as categorias de sinalização de regulamentação e advertência são as mais relevantes, a partir do ponto que são essas as sinalizações que exigem uma ação do condutor, neste caso, de um veículo autônomo. A sinalização de indicação, por ter uma natureza mais informativa, não foi incluída nas placas selecionadas para este trabalho.

Sendo assim, as placas de trânsito selecionadas para serem incluídas no modelo de treinamento são: *Duplo sentido de circulação, Parada obrigatória, Passagem obrigatória, Passagem sinalizada de pedestres, Proibido estacionar, Proibido trânsito de pedestres, Proibido virar à direita, Saliência ou lombada, Sentido de circulação da via ou pista, Sentido proibido, Trânsito de pedestres* e *Velocidade máxima permitida.*

De acordo com o manual de sinalização vertical de regulamentação CONTRAN (2007a), a categoria de sinalização de regulamentação possui um total de 51 sinais de regulamentação. Já no manual de sinalização vertical de advertência CONTRAN (2007b), a categoria de sinalização de advertência possui um total de 69 sinais de advertência. As duas categorias juntas totalizam 120 sinais de trânsito. Levando em consideração as placas selecionadas para este trabalho, foi englobado cerca de 10% do total das duas categorias de sinalização de trânsito, que são relevantes para os veículos autônomos.

[7] O CTB é a lei que contempla as principais regras de trânsito em vigor no Brasil.

Parte II

Metodologia

3 Metodologia

A metodologia, segundo Prodanov e Freitas (2013), é a aplicação de métodos e técnicas que possibilitam a coleta de informações, a compreensão, o entendimento e o aprofundamento de um determinado conhecimento ou objeto de estudo para a resolução de problemas ou questões a serem investigadas. Todo esse conjunto de características é parte do que conhecemos como pesquisa científica.

O conhecimento científico é construído por meio da investigação científica, que possui métodos e procedimentos, e que permite a elaboração conceitual da realidade. A mesma deve estar em constante enriquecimento e reformulação (FONSECA, 2002). Dessa forma, a pesquisa científica escolhida para esse trabalho possui alguns pontos de vista, que, de acordo com Prodanov e Freitas (2013), são: da forma de abordagem do problema, da sua natureza, dos seus objetivos e dos procedimentos técnicos. Esses pontos serão mais explorados no decorrer deste mesmo capítulo.

3.1 Classificação da Pesquisa

A classificação da pesquisa é um importante conceito da metodologia científica. Gerhardt e Silveira (2009) dizem que a classificação pode ser feita nas seguintes categorias: abordagem da pesquisa, natureza da pesquisa, objetivos da pesquisa, e procedimentos. Elas serão descritas nas seções a seguir.

3.1.1 Abordagem da Pesquisa

Quanto à abordagem, dentre todas as categorias existentes, a escolhida para essa pesquisa foi uma abordagem híbrida, *qualitativa-quantitativa*. Segundo Oliveira (2011), a ideia é que as abordagens quantitativa e qualitativa devem ser encaradas como complementares e não excludentes. O propósito é retirar os benefícios de ambas, para servir da melhor forma ao objetivo da pesquisa.

A pesquisa quantitativa, ainda segundo o autor, é construída pela utilização de elementos quantificáveis, tanto no desenvolvimento da pesquisa quanto na sua análise de resultados, usando a estatística como meio para o tratamento das informações.

A pesquisa quantitativa é aplicada para determinar também um perfil do objeto de estudo, tomando como base características e semelhanças a serem manuseadas. Em um grupo de pessoas, por exemplo, podem ser feitas análises demográficas. Através desse conjunto de dados é possível, por meio de técnicas estatísticas, inferir e predizer determinadas

ações, formas e traços com base nas observações feitas no objeto de estudo (MORESI, 2003).

Já a pesquisa qualitativa, de acordo com Prodanov e Freitas (2013), não utiliza dados estatísticos para a sua análise de um problema, não tendo como prioridade medir ou enumerar algo. Preocupa-se muito mais com o processo do que com o produto final. E na análise dos dados não existe preocupação em comprovar hipóteses previamente estabelecidas.

Portanto, o emprego da abordagem quantitativa, neste trabalho, tem como objetivo coletar e mensurar a precisão e a acurácia, assim como o poder de dedução ao identificar as placas de trânsito no *dataset* de imagens coletadas, que será discutido no decorrer deste TCC. E, dessa forma, a análise que será feita com o embasamento dos dados coletados será realizada pela abordagem qualitativa. Dessa maneira, será possível identificar o quão satisfatório está performando a rotulagem e o reconhecimento das imagens analisadas.

3.1.2 Natureza da Pesquisa

A natureza da pesquisa pode ser classificada em duas: básica ou aplicada. De acordo com Prodanov e Freitas (2013), o objetivo da pesquisa básica é gerar novos conhecimentos úteis para a sociedade, mas sem uma aplicação prática. Já a pesquisa aplicada gera conhecimentos para aplicações práticas com o enfoque na resolução de problemas específicos. Dessa forma, a natureza da pesquisa escolhida para este presente TCC é a *pesquisa aplicada*, visto que ela também tem como objetivo a solução de um problema específico.

3.1.3 Objetivo da Pesquisa

Para Gil (2002), os objetivos de uma pesquisa podem ser classificados em três categorias. Dentre elas, a mais apropriada para este trabalho é a *pesquisa exploratória*. O autor ainda alega que essa pesquisa tem como fundamentação trazer maior entendimento e familiaridade para o problema e torná-lo mais explícito.

Geralmente, utiliza-se o método exploratório como meio para alcançar o aprimoramento de ideias ou a descoberta de intuições (GIL, 2002). A pesquisa exploratória "procura esclarecer e definir a natureza de um problema e gerar mais informações que possam ser adquiridas para a realização de futuras pesquisas conclusivas."(OLIVEIRA, 2011)

O uso da pesquisa exploratória, neste trabalho, se dá pela necessidade do pesquisador poder ter um entendimento mais profundo do objeto de estudo, e, dessa forma, conseguir visualizar diversas formas de aplicabilidade da solução do problema para que, no fim, a sociedade se beneficie da pesquisa.

3.1.4 Procedimentos

Quanto aos procedimentos técnicos da pesquisa, foram escolhidos dois para esse trabalho, e podem ser divididos em duas etapas. Durante a primeira etapa, o procedimento atendido foi a *pesquisa bibliográfica*. De acordo com Prodanov e Freitas (2013), sua elaboração consiste na consulta de vários materiais já publicados, como, por exemplo, periódicos e artigos científicos, jornais, teses, entre outros. Essa pesquisa, ainda segundo o autor, tem como objetivo colocar o pesquisador em contato com o material de pesquisa do assunto abordado.

Já na segunda etapa da pesquisa, o procedimento seguido foi o *cenário de uso*. Segundo Rezende (2003), um cenário é uma narrativa, textual ou pictórica, de uma situação, muitas vezes de uso de uma aplicação. Envolve em seu ambiente muitas variáveis, tais como, usuários, processos e dados reais. O autor ainda relata que, de forma simplória, cenários são instâncias de casos de uso, ricas em detalhes contextuais.

A utilização do cenário de uso no desenvolvimento deste trabalho é necessária para descrever o contexto em que é realizada a investigação proposta pela pesquisa. Dessa forma, tal procedimento é de extrema importância para a construção do trabalho, com o enfoque nos métodos de coleta, análise e interpretação dos dados.

Com isso, finalizamos a classificação da pesquisa. A seguir, encontra-se a Tabela 1 que apresenta, de maneira resumida, todas as categorias e abordagens definidas para a realização deste trabalho.

Tabela 1 – Metodologia da Pesquisa.

Abordagem da pesquisa	**Natureza da pesquisa**	**Objetivos da pesquisa**	**Procedimentos**
Qualitativa/ Quantitativa	Aplicada	Exploratória	Pesquisa bibliográfica/ Cenário de Uso

Concluída a classificação da pesquisa, o desenvolvimento fica mais simplificado, visto que os processos de construção da pesquisa já estão definidos. Desse modo, a próxima seção irá retratar o desenvolvimento do processo metodológico.

3.2 Processo Metodológico

Visando melhorar o controle das atividades desenvolvidas ao longo da pesquisa, foi utilizado o *Kanban*. O *Kanban* é uma ferramenta que por meio visual consegue mostrar o status atual de execução de um conjunto de atividades. O mesmo é frequentemente consumido em projetos que envolvem metodologias ágeis (KNIBERG, 2009). Assim, para

essa pesquisa foi usado o *Trello*[1], uma ferramenta que, por sua vez, consiste em um sistema de quadro virtual para gerenciamento de tarefas que segue o método *Kanban*. Além de poder ser compartilhado com o orientador desse trabalho, facilitando na transparência da execução das atividades, o **Trello** possibilitou montar o cronograma do projeto.

Dessa forma, é possível extrair as macro atividades que serão realizadas neste TCC. A seguir, situa-se a Tabela 2. Esta tabela apresenta os principais marcos do projeto. Vale ressaltar que o conteúdo da Tabela 2 representa somente a condução do Trabalho de Conclusão de Curso 1 (TCC1), visto que, na UNB, a realização do TCC é dividida em duas partes, sendo elas o TCC1 e o TCC2.

Logo, o cronograma do TCC1 pode ser subdividido em 7 atividades: *definir tema, determinar escopo da pesquisa, pesquisar artigos e livros, estabelecer abordagem da metodologia, executar processo de desenvolvimento da pesquisa, documentar processo de desenvolvimento da pesquisa* e *apresentar TCC1*.

Tabela 2 – Cronograma de Atividades do TCC.

Cronograma	**Agosto**	**Setembro**	**Outubro**	**Novembro**	**Dezembro**
Definir tema	X				
Determinar escopo da pesquisa	X				
Pesquisar artigos e livros	X	X			
Estabelecer abordagem da Metodologia		X	X		
Realizar pesquisa		X	X	X	
Documentar processo de desenvolvimento da pesquisa			X	X	X
Apresentar TCC 1					X

Já a segunda etapa do trabalho, durante a realização do TCC2, o cronograma de atividades seguiu o apresentado na Tabela 3, e pode ser subdividido em 6 atividades: *melhorar o modelo de treinamento das placas de trânsito, coletar os cenários em vídeos que o modelo de treinamento será submetido, coletar da distância entre a placa de trânsito e a reação do modelo, avaliar o desempenho do modelo no reconhecimento de placas em vídeo, documentar processo de desenvolvimento da pesquisa* e *Apresentar TCC 2*.

[1] <https://trello.com/pt-BR>

Tabela 3 – Cronograma de Atividades para o TCC2.

Cronograma	Fevereiro	Março	Abril	Maio
Melhorar o modelo de treinamento das placas de trânsito	X	X		
Coletar os cenários em vídeos que o modelo de treinamento será submetido	X	X		
Coletar da distância entre a placa de trânsito e a reação do modelo			X	
Avaliar o desempenho do modelo no reconhecimento de placas em vídeo		X		
Documentar processo de desenvolvimento da pesquisa		X	X	
Apresentar TCC 2				X

Para melhor definir o processo de desenvolvimento, e, assim, ter um maior controle e uma visão completa sobre as etapas a serem executadas no TCC1, foi modelado o processo metodológico, que encontra-se na Figura 7.

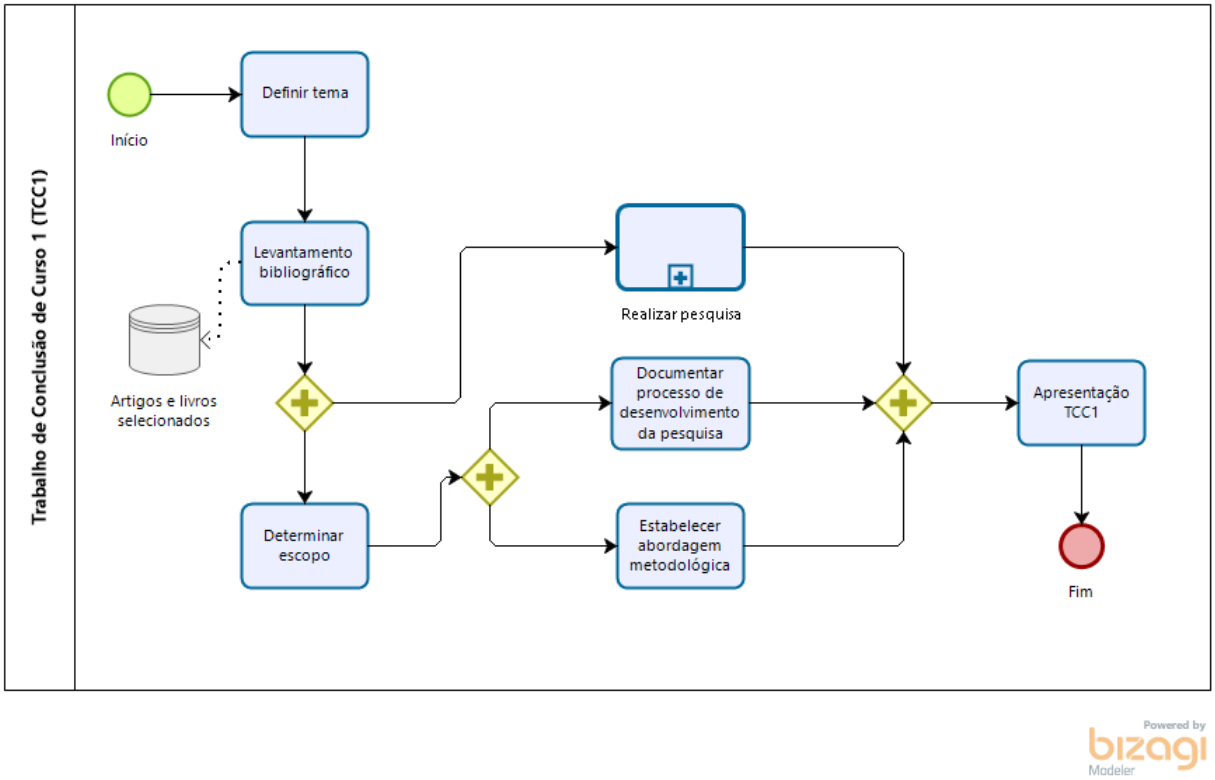


Figura 7 – Processo Metodológico para condução do TCC1.

Fonte: Autor

- **Definir Tema:**
  A atividade propõe a escolha do tema que será trabalhado, bem como o contexto no qual deseja ser abordado. Nessa etapa, o orientador deste TCC apresentou diversos temas para que, no fim, fosse escolhido um com maior afinidade com o pesquisador. Após a escolha, sucederam-se os próximos passos desse processo.

- **Levantamento Bibliográfico:**
  Com o tema previamente decidido, é necessário buscar referências (bases científicas) na área a fim de aumentar o conhecimento do pesquisador sobre o assunto. Para isso, foi utilizado o procedimento de *pesquisa bibliográfica* para garantir o entendimento do contexto, que é preciso para especificar mais adequadamente o escopo. Esse processo pode ser encontrado na seção 3.2.1 - *Levantamento Bibliográfico*.

- **Determinar Escopo:**
  Após conhecer várias abordagens sobre o tema proposto, esse é o momento de definir os limites da pesquisa, descrevendo de maneira clara a contextualização do tema, o objetivo geral e específico da pesquisa, bem como a justificativa e a metodologia da pesquisa.

- **Realizar Pesquisa:**
  Essa atividade é um subprocesso que pode ser visto na seção 3.2.2 - *Realizar Pesquisa*. De maneira geral, engloba o *cenário de uso*, que é o momento no qual serão realizadas as etapas para coletar, analisar e interpretar dados que serão gerados por meio da pesquisa.

- **Documentar processo de desenvolvimento da pesquisa:**
  Trata-se da escrita dos dados e informações obtidos durante o processo de desenvolvimento da pesquisa, tornando possível, assim, apresentar a documentação da pesquisa de forma concisa e coerente.

- **Estabelecer abordagem da Metodologia:**
  Durante esta atividade, é realizada a metodologia da pesquisa. Ela está presente na construção do desenvolvimento da pesquisa, sendo constantemente aprimorada ao longo da realização do trabalho para, dessa forma, alcançar o objetivo proposto.

- **Apresentar TCC 1:**
  Relatar os resultados obtidos da pesquisa para o professor orientador e a banca examinadora.

Já o processo de desenvolvimento previsto para o TCC2, encontra-se na Figura 8. Essas são as etapas macro a serem executadas.

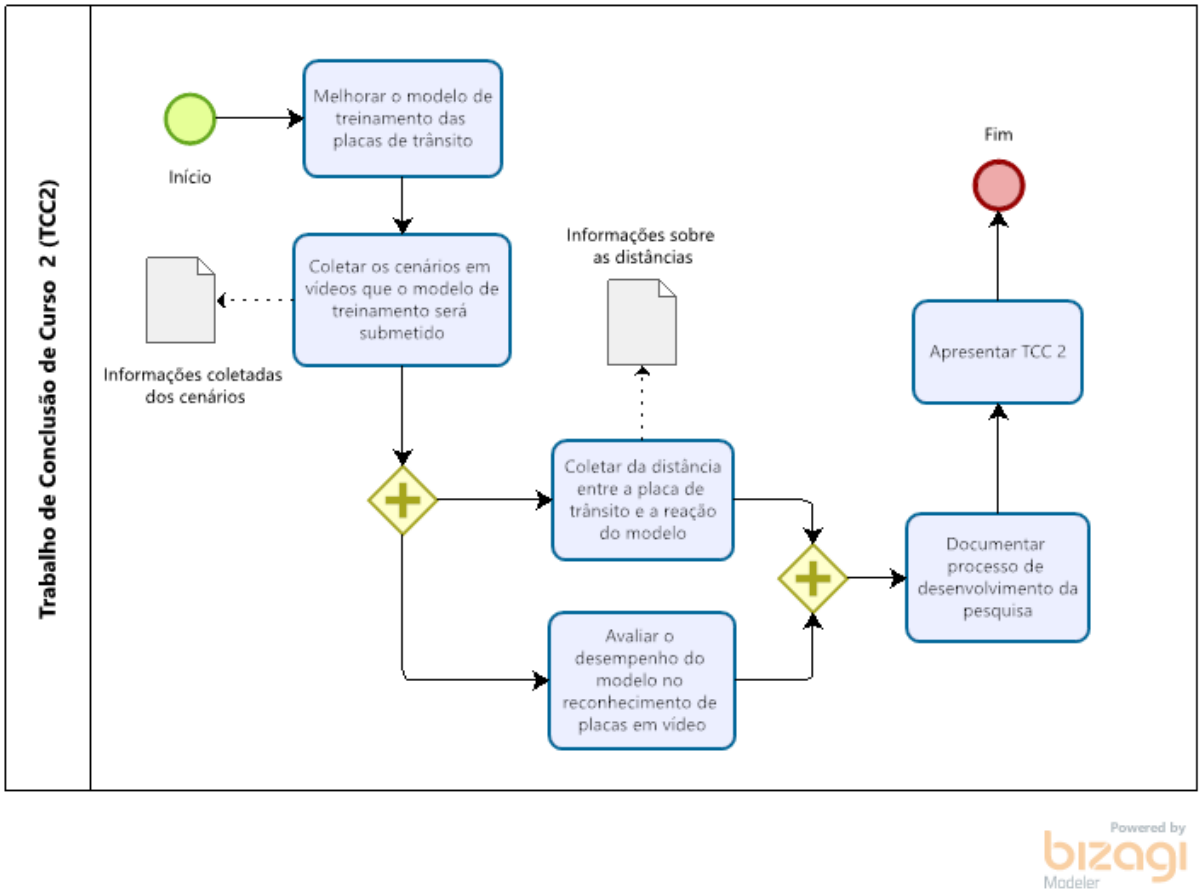


Figura 8 – Processo Metodológico para condução do TCC2.

Fonte: Autor

- **Melhorar modelo de treinamento:**
  Essa etapa consiste em coletar mais imagens para incrementar o treinamento de placas de trânsito brasileiras para que, assim, a detecção seja efetiva para ser utilizada em vídeos. Ainda, adicionar as placas de velocidade no modelo de treinamento.

- **Coletar os cenários em vídeo:**
  Essa atividade é importante, pois é nela que serão gravados e coletados os vídeos de um percurso pré definido, e será exposto a vários cenários diferentes.

- **Coletar a distância:**
  Após a coleta do vídeo, é possível descobrir de forma aproximada qual é a distância que o modelo de treinamento começa a reconhecer uma placa de trânsito brasileira.

- **Avaliar o desempenho do modelo nos vídeos:**
  Nessa etapa, os vídeos dos cenários coletados serão submetidos a várias avaliações, e a partir disso será coletado um resultado através da matriz de confusão.

- **Documentar processo de desenvolvimento da pesquisa:**
  Trata-se da escrita dos dados e informações obtidos durante o processo de desenvolvimento da pesquisa, tornando possível, assim, apresentar a documentação da pesquisa de forma concisa e coerente.

- **Apresentar TCC 2:**
  Relatar os resultados obtidos da pesquisa para o professor orientador e a banca examinadora.

3.2.1 Levantamento Bibliográfico

A atividade *Levantamento Bibliográfico* foi construída por meio da técnica de pesquisa bibliográfica, que, segundo Fonseca (2002), é feita a partir do levantamento de referências teóricas publicadas e analisadas, podendo ser artigos, livros, páginas de web, sites, entre outros. O objetivo é permitir ao pesquisador conhecer mais sobre o assunto a ser estudado.

Dessa forma, o procedimento seguiu do seguinte modo. Foram captados artigos científicos do site oficial do YOLO[2], que serviu de embasamento para este TCC. Com base nos artigos encontrados na base do site, foram selecionados três, que podem ser vistos na Tabela 4. O critério para a escolha destes artigos foi por eles abordarem como tema principal o funcionamento do YOLO.

Tabela 4 – Resultado da pesquisa no web site do Yolo.

Título	**Autor(es)**	**Data**
Real-Time Grasp Detection Using Convolutional Neural Networks	Joseph Redmon e Anelia Angelova	2015
You Only Look Once: Unified, Real-Time Object Detection	Joseph Redmon, Santosh Divvala, Ross Girshick, e Ali Farhadi	2016
YOLOv3: An Incremental Improvement	Joseph Redmon e Ali Farhadi	

[2] <https://pjreddie.com/publications/>. Acessado em: 22 de Abril de 2021

Também foi feito um levantamento de referências na base de dados do Periódico Capes[3]. Realizou-se uma busca avançada na base utilizando algumas palavras-chave, são elas: “Yolo v3”, “Yolo v3 and Deep Learning”, “Detecção, reconhecimento de placas de trânsito” e “Yolo v3 and Traffic sign”. A única restrição aplicada foi a data de publicação que, nesse contexto, foram pesquisas publicadas nos últimos cinco anos. O resultado da busca pode ser conferido na Tabela 5.

Tabela 5 – Resultado da busca no Periódico Capes.

Título	**Autor(es)**	**Data**
A Comparison of Embedded Deep Learning Methods for Person Detection3	Chloe Eunhyang Kim, Mahdi Maktab Dar Oghaz, Jiri Fajtl, Vasileios Argyriou e Paolo Remagnino	2019
A Deep Learning Approach of Vehicle Multitarget Detection from Traffic Video	Xun Li, Yao Liu, Zhengfan Zhao, Yue Zhang, e Li He	2018
A Fast Learning Method for Accurate and Robust Lane Detection Using Two-Stage Feature Extraction with YOLO v3	Xiang Zhang, Wei Yang, Xiaolin Tang, e Jie Liu	2018
Aprendizado de máquina aplicado para auxílio ao motorista utilizando Raspberry pi	Luan Lourenço Esteves, Francisco Assis da Silva, Leandro Luiz de Almeida, Danillo Roberto Pereira, Mário Augusto Pazoti, Almir Olivette Artero2	2019
Real Time Detection and Classification of Traffic Signs Based on Yolo Version 3 Algorithm	V.N. Sichkar, S.A. Kolyubin	2020

[3] <http://www-periodicos-capes-gov-br.ezl.periodicos.capes.gov.br/>. Acessado em: 16 de Setembro de 2020

Com base na leitura desses artigos listados na Tabela 5, mais dois artigos foram encontrados através do referencial teórico dos artigos lidos. Esses novos artigos podem ser encontrados na Tabela 6.

Tabela 6 – Artigos encontrados no Referencial Teórico de outros artigos.

Título	**Autor(es)**	**Data**
Deep Learning for Large-Scale Traffic-Sign Detection and Recognition	Domen Tabernik e Danijel Skocaj	2020
Real-Time Traffic Sign Recognition Based on Shape and Color Classification	Tuğhan Çağlayan, Habibullah Ahmadzay e Gökhan Kofraz	2020

Por fim, também foi realizado um levantamento na base de dados do Scopus[4]. Utilizando a *string* de busca "(Yolov3 AND (detection OR dentif* OR recogni*) AND traffic sign)", o resultado trouxe vários artigos relevantes quanto a temática. Podem ser vistos na Tabela 7.

Tabela 7 – Resultado da busca no Scopus.

Título	**Autor(es)**	**Data**
Detecting Small Chinese Traffic Signs via Improved YOLOv3 Method	Zhang, B. e Wang, G. e Wang, H. e Xu, C. e Li, Y. e Xu, L.	2021
Improved detection method for traffic signs in real scenes applied in intelligent and connected vehicles	Du, L. e Ji, J. e Pei, Z. e Zheng, H. e Fu, S. e Kong, H. e Chen, W.	2020
Traffic signs detection and recognition under low-illumination conditions	Zhao, K. e Liu, L. e Meng, Y. e Sun, R.-C.	2020
Road Sign Detection and Recognition of Thai Traffic Based on YOLOv3	Thipsanthia, P. e Chamchong, R. e Songram, P.	2019

O processo de pesquisa bibliográfica teve como resultado um maior entendimento sobre o tema abordado. Sendo assim, seu objetivo primordial foi alcançado, já que houve um aumento na maturidade sobre o assunto da pesquisa.

[4] <https://www-scopus.ez54.periodicos.capes.gov.br/>. Acessado em: 16 de Setembro de 2020

3.2.2 Realizar Pesquisa

A atividade *Realizar Pesquisa* é construída a partir do cenário de uso. Nela, é possível coletar, analisar e interpretar os dados. Na Figura 9, é perceptível observar o processo abordado para o desenvolvimento deste trabalho. Em seguida, é descrito o que ocorre em cada passo das atividades.

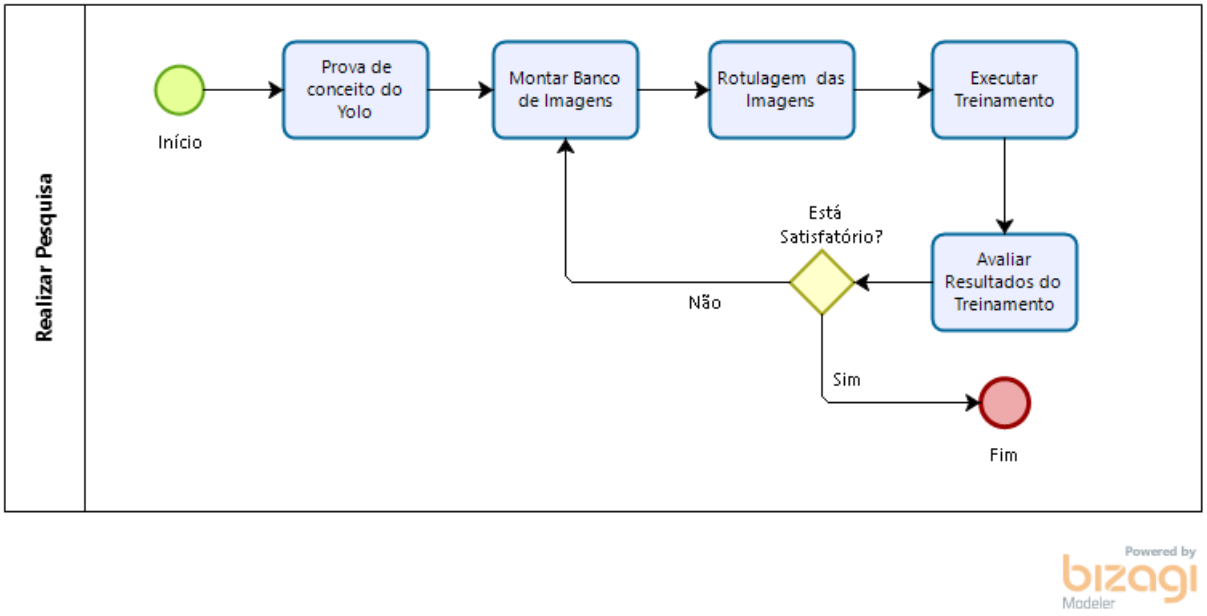


Figura 9 – Processo Metodológico de realização da pesquisa do TCC1.

Fonte: Autor

- **Prova de Conceito do YOLO:**
  Essa atividade é necessária para que o pesquisador ambientalize a utilização do YOLO. É nesse momento que são realizados os primeiros testes sobre como funciona e como se executa de fato o YOLO, utilizando um pequeno teste para validar o funcionamento da ferramenta.

- **Montar Banco de Imagens:**
  Trata-se de montar um *dataset* de imagens de placas de trânsito. Durante essa atividade, foram utilizadas várias formas de procura e coletas de imagens para que, dessa forma, fosse possível realizar o treinamento do *dataset*.

- **Rotulagem das Imagens:**
  Após a coleta das imagens, é preciso realizar a rotulagem das mesmas. Sendo assim, cada imagem deve conter um arquivo de texto descrevendo as coordenadas do objeto de interesse dentro das imagens. Estas coordenadas devem estar especificadas para o YOLO, para que, dessa forma, ele consiga reconhecer e entender.

- **Executar Treinamento:**
  Durante essa atividade, ocorre o treinamento do *dataset* já com a rotulagem realizada. Assim, é configurado tudo o que precisa para que, de fato, o treinamento ocorra. Esse momento é o que mais despende tempo, já que um treinamento demora para ser concluído dependendo do volume de imagens no *dataset*.

- **Avaliar Resultados do Treinamento:**
  A avaliação consiste na análise da precisão de resposta do treinamento realizado. Para medir essa precisão, será utilizada a matriz de confusão. Caso a avaliação do treinamento seja insatisfatória, as etapas desse processo são refeitas.

O próximo capítulo irá detalhar mais sobre como ocorreu o desenvolvimento do segundo procedimento escolhido para esse TCC, o cenário de uso. Nele, será possível observar os resultados obtidos da pesquisa.

Parte III

Resultados

4 Resultados Obtidos

Neste capítulo, são apresentadas as etapas do processo de "realizar pesquisa", que foi mostrado no Capítulo 3 - Metodologia, e situa-se na Figura 9. De modo respectivo, o processo de condução do TCC2, mostrado na Figura 8, será apresentado.

Estarão presentes o desenvolvimento do projeto para alcançar, de fato, o objetivo do trabalho, que é construir um procedimento que seja replicável para que outros pesquisadores sejam capazes de realizar a rotulagem; o treinamento adequado com a finalidade de obter bons resultados na detecção de placas de trânsito nas imagens; o reconhecimento de placas de trânsito em vídeo e em diferentes contextos; além de verificar a distância máxima em que é identificada a detecção e a classificação das placas de trânsito brasileiras.

Sendo assim, este capítulo visa acentuar o caminho que foi percorrido para formular os resultados que serão apresentados. Dito isso, o capítulo irá descrever as seguintes etapas: *Prova de Conceito do YOLO*, *Montar Banco de Imagens*, *Rotulagem das Imagens*, *Executar Treinamento*, *Avaliar Resultados do Treinamento*, *Aplicação do modelo de treinamento em vídeos* e *Distância máxima do reconhecimento do modelo treinado*.

4.1 Prova de Conceito do YOLO

Segundo Guimarães (2008), "define-se prova de conceito (usa-se no texto a sigla POC, do inglês Proof of Concept) como uma técnica que permite demonstrar que uma determinada ideia é tecnicamente possível, ou seja, pode ajudar a verificar se uma arquitetura é 'construível'". Dessa forma, o objetivo dessa POC é colocar o pesquisador em contato com o sistema do YOLOv3 e, porventura, testar o funcionamento que o algoritmo se propõe a fazer, que basicamente é realizar a detecção de objetos (REDMON et al., 2016).

Para que essa POC obtenha sucesso, é necessário que o YOLOv3, ao final, cumpra o papel fundamental de detectar um objeto corretamente na imagem proposta pelo pesquisador. Para isso, serão utilizados como teste os passos de execução do sistema descrito no web site oficial do YOLOv3[1] disponibilizado por Redmon e Farhadi (2018).

Para esse teste, foi utilizado um notebook da marca *Samsung* modelo 370E4K, suas especificações são: processador Intel core i5-5200U, 8 GB[2] de memória RAM[3], placa gráfica Intel Graphics 5500 e 1 TB[4] de espaço para armazenamento. O Linux foi utilizado

[1] <http://https://pjreddie.com/darknet/yolo/>. Acessado em: 09 de Setembro de 2020
[2] Símbolo referente a Gigabyte, que é uma unidade de medida de informação.
[3] É uma sigla, em inglês, para *Random Access Memory*, que significa "Memória de Acesso Aleatório".
[4] Símbolo referente a Terabyte, que é uma unidade de medida de informação.

para o sistema operacional. Explicado o ambiente de desenvolvimento utilizado, basta seguir o passo a passo que será apontado.

1. A primeira etapa a ser feita é a instalação do *Darknet* que foi abordado na Seção 2.1. A forma mais simples de fazer a instalação é clonando o repositório do *Darknet* na sua máquina. Após o clone, basta entrar na pasta e executar o *Makefile* do projeto, que é onde algumas configurações importantes ficam armazenadas. Os comandos de terminal deste primeiro passo são respectivamente:

```
$ git clone https://github.com/pjreddie/darknet
$ cd darknet
$ make
```

2. No segundo momento, faz-se necessário o *download* de um arquivo pré-treinado de 237 MB. Esse arquivo contém o treinamento de *Dataset* do COCO[5], que possui aproximadamente 80 categorias de identificação de objetos. Dessa forma, este será utilizado para essa POC para fazer o *download*. Basta usufruir da linha de comando.

```
$ wget https://pjreddie.com/media/files/yolov3.weights
```

3. O último passo refere-se à execução da detecção na imagem escolhida para a predição.

```
$ ./darknet detect cfg/yolov3.cfg yolov3.weights data/
   stop_sign.jpg
```

Após a execução do comando do detector, é mostrada uma resposta. Nessa situação, a resposta esperada é vista na Figura 10. Nota-se que ao fim da saída de resposta, o *Darknet* fornece o tempo que foi gasto para detectar, e também mostra o percentual de confiança dos objetos.

Seguidamente da impressão da resposta, é gerada uma imagem que contém as *bounding boxes* em volta dos objetos detectados. Essa imagem é salva no "predictions.png". Logo após abrir essa imagem que havia sido salva, é exibido o resultado que pode ser visto na Figura 11.

[5] <https://cocodataset.org/>. Acessado em: 10 de Setembro de 2020

```
layer   filters  size/strd(dil)      input                output
1 conv     64       3 x 3/ 2    416 x 416 x  32 ->  208 x 208 x  64 1.595 BF
2 conv     32       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
  ..............
104 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
105 conv     51       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x  51 0.071 BF
106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.384
avg_outputs = 518912
 Allocate additional workspace_size = 52.43 MB
Loading weights from /mydrive/yolov3/backup/yolov3_custom_last.weights...
 seen 64, trained: 1536 K-images (24 Kilo-batches_64)
Done! Loaded 107 layers from weights-file
 Detection layer: 82 - type = 28
 Detection layer: 94 - type = 28
 Detection layer: 106 - type = 28
data/stop.jpg: Predicted in 11.133000 milli-seconds.
stop_sign: 100%
```

Figura 10 – Saída esperada após executar o comando do detector.

Fonte: Autor.

Figura 11 – Exemplo de predição feita pelo YOLOv3.

Fonte: Autor.

Com o desenvolvimento da POC, e através dos resultados obtidos após a sua execução, é notório perceber que os critérios iniciais programados para essa pesquisa foram um sucesso. É factível observar também que o YOLOv3 entrega realmente o que o pesquisador esperava, tornando possível, assim, a continuação da pesquisa.

4.2 Montar Banco de Imagens

A montagem do banco de imagens é primordial para que o treinamento seja feito, visto que, sem dados, torna-se difícil o trabalho de um algoritmo de rede neural, por exemplo. Visando o objetivo do trabalho, que é fazer um treinamento para detecção de placas de trânsito brasileiras, faz-se necessário um banco de dados com tais placas.

O primeiro passo aplicado para essa etapa do trabalho foi procurar por bases de dados e verificar se já existe um conjunto de imagens pré montadas sobre placas de trânsito brasileiras, que poderiam ser utilizadas nessa pesquisa. A busca foi feita em várias bases de dados, como pode ser observado na Tabela 8. Infelizmente, não foi encontrado banco de imagens que contenha placas de trânsito brasileiras que pudesse ser utilizado nesta pesquisa.

Tabela 8 – Levantamento nas Bases de dados.

Acessado em: 10 de Setembro de 2020.

Base de Dado	**Url do Web Site**
Kaggle	<https://www.kaggle.com/datasets>
COCO Explorer	<https://cocodataset.org/#explore>
Open Images Dataset v6	<https://storage.googleapis.com/openimages/web/index.html>
Five Thirty Eight	<https://data.fivethirtyeight.com/>
BuzzFeed	<https://github.com/BuzzFeedNews/everything#standalone-datasets>
Data.gov	<https://catalog.data.gov/dataset/>
Brasil.io	<https://brasil.io/datasets/>
Dados.gov.br	<http://dados.gov.br/dataset>
Reddit	<https://www.reddit.com/r/datasets/>
Academic Torrents	<https://academictorrents.com/>

Não sendo possível encontrar uma base de dados pré pronta que pudesse ser utilizada nessa pesquisa, a possibilidade a ser acolhida foi construir o próprio banco de imagens para o trabalho. A partir dessa decisão, é necessário diminuir o número de amostras, ou seja, escolher um pequeno conjunto de placas de trânsito brasileiras que será aplicado neste trabalho, visto que diversas imagens deverão ser coletadas para cada tipo de placa.

Desse modo, foram selecionados 12 tipos de placas de trânsito brasileiras, são elas: *Duplo sentido de circulação, Parada obrigatória, Passagem obrigatória, Passagem sinalizada de pedestres, Proibido estacionar, Proibido trânsito de pedestres, Proibido virar à direita, Saliência ou lombada, Sentido de circulação da via ou pista, Sentido proibido,*

Trânsito de pedestres e *Velocidade máxima permitida*. As imagens das respectivas placas de trânsito constam na coluna "Imagens das placas" da Tabela 9.

Tabela 9 – Quantidade de imagens de placas coletadas.

Imagens das placas	**Nome das placas**	**Quantidade de Imagens**
	Duplo sentido de circulação	66
	Parada obrigatória	165
	Passagem obrigatória	195
	Passagem sinalizada de pedestres	297
	Proibido estacionar	123
	Proibido trânsito de pedestres	30
	Proibido virar à direita	106
	Saliência ou lombada	210
	Sentido de circulação da via ou pista	163
	Sentido proibido	59
	Trânsito de pedestres	36
	Velocidade máxima permitida	130
	TOTAL	**1580**

Logo após a seleção das placas de trânsito, foi realizada uma saída de campo com o objetivo de coletar tais placas. A atividade consiste em procurar as placas selecionadas em avenidas, ruas e vias, e, assim que encontradas, tirar uma fotografia dessas placas. Um dos motivos desta saída de campo mostrar-se necessária é pelas escassas imagens

encontradas no *Google Imagens*. Sendo assim, julgou-se essencial a realização dessa busca e coletas de placas de trânsito brasileiras.

É importante ressaltar ainda, que para as imagens de placas de trânsito de *velocidade máxima permitida* foram coletadas, todos os tipo de limite de quilometragem, e não apenas o de 90 km/h, por exemplo. Sendo assim, independente do limite estabelecido na placa de trânsito, ela será incluída como uma placa de *velocidade máxima permitida*.

Dessa forma, ao fim do período planejado para essa tarefa, o resultado foi a coleta de 1580 imagens no total. Na Tabela 9, pode ser conferido o detalhamento da quantidade de imagens coletadas para cada tipo de placa de trânsito.

É válido ressaltar que em algumas imagens há mais de um tipo de placa, deixando a quantidade de imagens coletadas com uma pequena margem de erro, visto que elas não foram contabilizadas mais de uma vez. Um exemplo pode ser encontrado na Figura 12, que possui tanto a placa de *proibido estacionar* quanto a placa de *passagem sinalizada de pedestres*.

Figura 12 – Exemplo de fotografia com duas placas de trânsito diferentes.

Fonte: Autor

As imagens coletadas foram fotografadas utilizando um aparelho celular em formato retrato, ou seja, na vertical. Ainda é importante observar que foram fotografados vários ângulos de uma mesma placa de trânsito, enriquecendo o *dataset* montado. Por fim, os critérios de qualidade utilizados para selecionar as melhores fotos foram os mesmos do Griffin, Holub e Perona (2007), que definem basicamente 3 critérios subjetivos:

- **Bom:** um visual limpo do objeto fotografado. Essas são as melhores e mais eficazes, e foram utilizadas para o TCC.

- **Ruim:** um visual mais confuso, com pouca iluminação e de difícil entendimento. Algumas ainda são possíveis de serem utilizadas, mas no geral são descartadas.

- **Não aplicável:** são fotografias nas quais o objeto não existe. Essas imagens são inutilizáveis.

4.3 Rotulagem das Imagens

Assim que o *dataset* é construído, o próximo passo é fazer a rotulagem/*Labeling*. A rotulagem nada mais é do que fazer a marcação das *bounding boxes* nas imagens, ou seja, é identificar manualmente os objetos de interesse dentro de uma imagem para que, dessa maneira, seja possível realizar o treinamento. No contexto deste trabalho, os objetos de interesse a serem marcados são placas de trânsito brasileiras. Conforme descrito na seção 2.1, pode ser observado um exemplo de marcação de uma *bounding box* em uma imagem de placa de trânsito na Figura 1.

Para realizar a rotulagem das imagens coletadas, foi necessário utilizar a ferramenta "LabelImg" desenvolvida por Tzutalin (2005). Com essa ferramenta, é possível fazer as marcações das *bounding boxes* e gerar o arquivo necessário com as notações do YOLOv3, para que, dessa forma, a rotulagem do *dataset* seja concluída.

O passo a passo para a instalação da ferramenta LabelImg[6] ocorreu em um ambiente com o sistema operacional Linux, mas o LabelImg disponibiliza também versões para "Windows" e "MacOs". A instalação no Linux realizou-se da seguinte maneira:

1. Antes de tudo, é necessário que a sua máquina já possua pré instalado o "Python 3" ou versões superiores, bem como "o pip3". Dessa forma, é possível prosseguir com os requerimentos básicos. A primeira coisa a ser feita é a cópia do LabelImg para sua máquina. Essa etapa pode ser realizada executando no terminal o comando:

   ```
   $ git clone https://github.com/tzutalin/labelImg.git
   ```

2. Após clonar o repositório, faça a instalação de uma ferramenta de desenvolvimento, e também das dependências requeridas pelo LabelImg. Depois, basta executar o "make" para compilar o programa. Esses passos podem ser realizados respectivamente, executando os comandos:

   ```
   $ sudo apt-get install pyqt5-dev-tools
   ```

[6] <https://github.com/tzutalin/labelImg>

```
$ sudo pip3 install -r requirements/requirements-linux-
   python3.txt
$ make qt5py3
```

3. Por fim, basta executar o LabelImg de fato utilizando o comando:

```
$ python3 labelImg.py [Caminho_da_pasta_com_o_dataset] [
   Arquivo_de_classes]
```

Nota-se que dois argumentos são exigidos neste último comando. O primeiro é o caminho no qual encontra-se a pasta com o conjunto de imagens coletas. Uma possibilidade para conseguir essa informação é caminhar via terminal até a pasta em que se encontra o *dataset*. Após isso, execute o comando:

```
$ pwd
```

Com isso, será possível saber o caminho necessário até o seu conjunto de imagens. Um exemplo de saída é "/home/bruno/TCC/Dataset". Tendo essa informação, basta substituir o [Caminho_da_pasta_com_o_dataset] no comando de execução do LabelImg pelo caminho que foi encontrado.

O segundo argumento essencial é o nome do arquivo de classes, que geralmente é denominado de "classes.txt" por padrão, e deve estar localizado dentro da pasta do LabelImg. Esse arquivo de texto deve conter as categorias ou classes do *dataset* para que, por exemplo, este TCC consiga detectar e classificar o tipo de placa de trânsito identificada na predição.

Dessa forma, é necessário que as onze placas de trânsito brasileiras, selecionadas na seção 4.2, estejam contidas dentro do arquivo de texto. Como pode ser observado na "Tabela 10" , a coluna "Classes e Categorias" contém as categorias que estarão dentro do arquivo "classes.txt". Percebe-se que os nomes das placas de trânsito foram traduzidos para o inglês, e foi aplicado o *Snake Case*[7]. Por fim, é importante salientar que cada linha do arquivo "classes.txt" representa uma categoria.

Com esses dois argumentos concluídos, é possível, enfim, executar o último comando dos passos de instalação. Como resposta, o programa do LabelImag irá abrir, e a interface recebida deverá ser algo semelhante à Figura 13. Nesta mesma figura, é possível observar algumas marcações coloridas em conjunto com uma letra representante. Essas caixas coloridas englobam os principais pontos de funcionamento da interface da ferramenta. Cada ponto de interesse será apresentado e explicado de uma maneira breve.

[7] Refere-se ao estilo de escrita no qual o espaço é substituído por um carácter de sublinhado, e a escrita das palavras é feita em caixa-baixa.

Tabela 10 – Categorias das placas de trânsito do arquivo de classes.

Nome das placas	**Classes ou Categorias**
Duplo sentido de circulação	double_direction_of_circulation_sign
Parada obrigatória	stop_sign
Passagem obrigatória	obligatory_passage_sign
Passagem sinalizada de pedestres	pedestrian_crossing_sign
Proibido estacionar	no_parking _sign
Proibido trânsito de pedestres	no_pedestrian _traffic_sign
Proibido virar à direita	forbidden_to_turn_right_sign
Saliência ou lombada	bump_or_spine_sign
Sentido de circulação da via ou pista	bump_or_spine_sign
Sentido proibido	wrong_way_sign
Trânsito de pedestres	pedestrian_traffic_sign
Limite de velocidade	speed_limit_sign

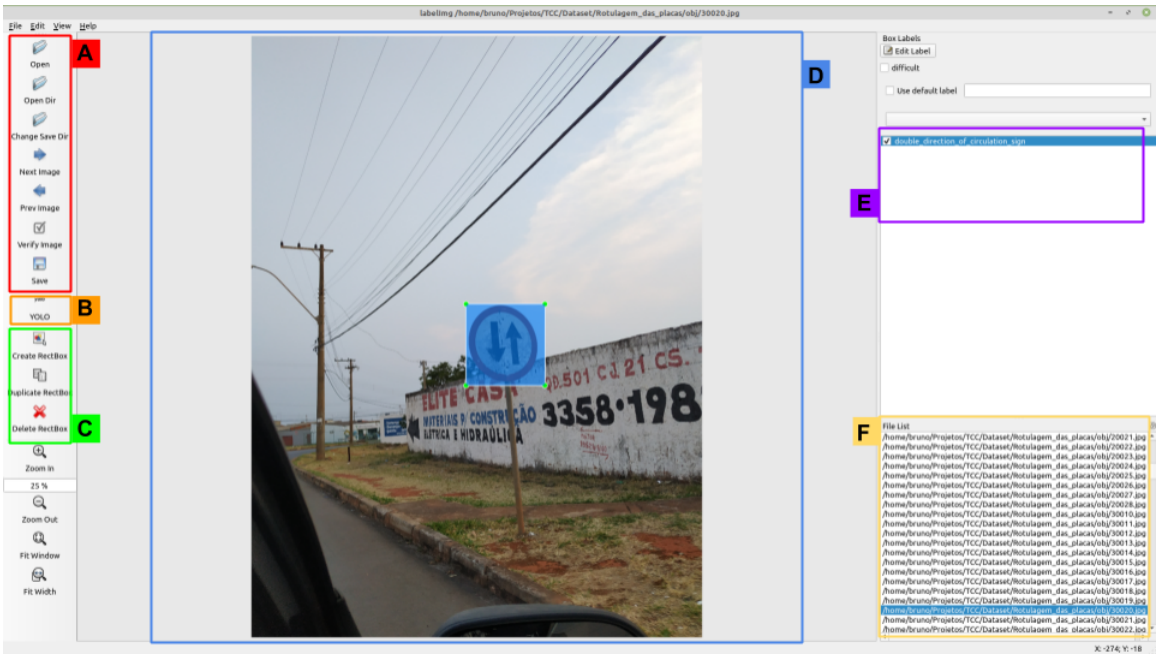


Figura 13 – Interface do programa LabelImag.

Fonte: Autor

- **A:** Este menu de ferramentas possibilita algumas ações importantes dentro do LabelImg. Os três primeiros ícones da lista são utilizados para carregar novas imagens para dentro do programa. Os ícones em formato de “seta”servem para ir para a próxima imagem, ou acessar a anterior. Neste menu, a função mais importante é a de “Salvar”, sendo esta a que consolida a rotulagem de cada imagem.

- **B:** Esse botão é muito importante, pois é ele que define a formatação de saída do YOLOv3. Dessa maneira, basta configurar para o modo YOLO. Essa formatação será apresentada em breve.
- **C:** neste menu de ferramentas existe a função "Create ReactBox". Ela é responsável por criar a *bounding box* na imagem.
- **D:** é a área principal de trabalho. É nela que consta a imagem, na qual será feita a marcação da *bounding box*, como é mostrado na Figura 13: o quadrado azul com bordas verdes representa uma marcação de uma placa de trânsito brasileira. Em uma única imagem pode ser feita mais de uma rotulagem. É nesse momento, também, que é selecionado o tipo de categoria. No caso desse exemplo, a categoria ou classe selecionada é a *double_direction_of_circulation_sign*.
- **E:** esse painel informa as categorias que foram classificadas em cada imagem.
- **F:** por fim, esse painel mostra todas as imagens que foram carregadas no programa, sendo possível acessá-las a qualquer momento.

Entendendo um pouco mais sobre o funcionamento básico do LabelImg, basta, agora, saber o tipo de resultado que ele entrega. Ao salvar uma imagem com a *bounding box* já marcada, será gerado um arquivo de texto com o mesmo nome da imagem na qual foi realizada a rotulagem. Tanto a imagem quanto o arquivo de texto são salvas no mesmo local. O arquivo de texto gerado contém as coordenadas de identificação das *bounding boxes* da imagem, por isso que na etapa **B** é importante selecionar o "YOLO". Esse arquivo é gerado para cada imagem em que é realizada a rotulagem.

Para entender o funcionamento das coordenadas contidas no arquivo de texto, é importante observar a Figura 14, que mostra um exemplo de coordenadas. Percebe-se que existe uma série de números em apenas uma linha, e que cada argumento de informação é separado por um "espaço em branco". Essa sequência é a localização de uma *bounding box* marcada na imagem. Cada linha no arquivo de texto representa uma *bounding box* diferente em uma mesma imagem.

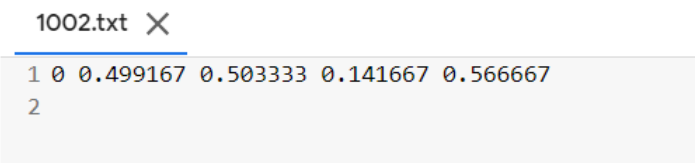


Figura 14 – Exemplo de arquivo de texto com as coordenadas de formatação do YOLO.

Fonte: Autor

Examinando ainda a Figura 14, tem-se que, nesse exemplo, o primeiro elemento é o número "0". Este número representa a classe, na qual foi realizada a marcação da

bounding box. Esse número tem correlação com o arquivo de classes mostrado na Tabela 10. Como explicado anteriormente, no arquivo "classes.txt", cada linha representa uma categoria. Dessa forma, o número "0" é retirado. Ele é a posição da primeira categoria do arquivo de classes, sendo que a contagem se inicia no número "0", ou seja, a primeira categoria é representada pelo numeral "0", e a contagem continua de uma forma crescente. Caso haja uma segunda classe dentro do arquivo de classes, ela será representada pelo número "1".

Por fim, ainda há mais quatro elementos dentro do arquivo de coordenada. Com o auxílio da Figura 15, é possível perceber que cada elemento representa um ponto do plano cartesiano, sendo que X varia entre [0,1], onde (0) é o pixel[8] mais à esquerda e (1) é o pixel mais à direita. Igualmente, as coordenadas em Y estão entre [0,1] e vão do pixel superior (0) para o pixel inferior (1), (VITTORIO, 2018). Dessa forma, a anotação feita no arquivo de coordenadas vista na Figura 14 é representada assim:

```
representacao_da_classe esquerda topo direita inferior
```

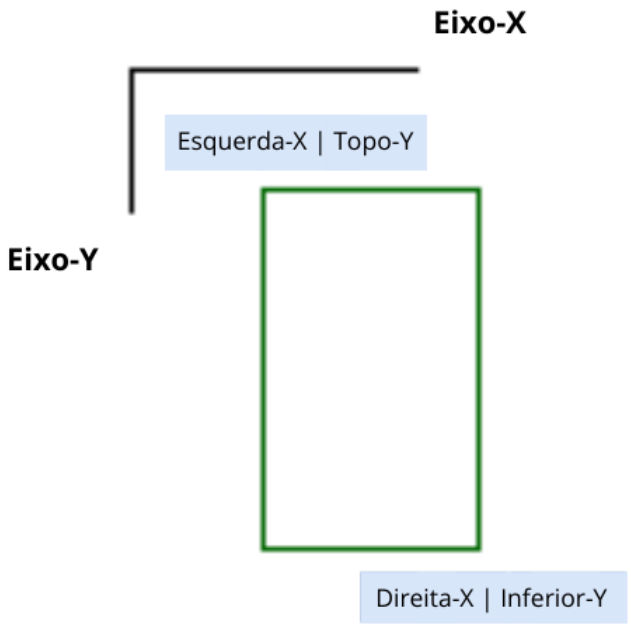


Figura 15 – Coordenadas de uma *bounding box*.

Fonte: Adaptado do Vittorio (2018).

Com isso, são finalizados os principais conceitos que serão abordados neste trabalho à respeito de rotulagens. Essa etapa é crucial para que a fase do treinamento seja executada da melhor forma.

[8] É o menor ponto que forma uma imagem digital.

4.4 Executar Treinamento

Esta seção tem como objetivo apresentar a forma em que procedeu o treinamento do *dataset*. Antes de adentrar em seu desenvolvimento, é importante ressaltar sobre o ambiente de desenvolvimento. Para essa etapa, não será mais utilizada a máquina, cujas especificações foram mostradas na Seção 4.1, por motivo de processamento computacional. Para fazer o treinamento de uma forma mais otimizada, é preciso de uma GPU.

Desse modo, a solução encontrada foi a utilização do Google Colab[9]. Ele nada mais é do que um ambiente virtual, onde o próprio Google disponibiliza notebooks Jupyter[10], que não requer configuração e é executado na nuvem. O Colab é muito acessível, pois disponibiliza versões de máquinas virtuais com GPU de forma gratuita. Sendo assim, para o desenvolvimento, o treinamento será por meio de um ambiente virtual.

Neste trabalho, foram utilizados tanto o serviço gratuito quanto o pago, que oferece alguns benefícios a mais. As GPUs usadas foram a T4 e P100 no serviço pago, e a K80s na assinatura gratuita. É importante ressaltar que as GPUs por padrão no Colab são desabilitadas. Para habilitá-las, basta ir na barra de ferramentas selecionar a opção "Tempo de execução", e em seguida "Alterar tipo de tempo de execução", assim, apenas selecione a opção de GPU.

Por fim, o Colab faz uso de um ambiente interativo de desenvolvimento. Isso diz respeito às células de código, onde são depositadas as instruções que o comando deve executar. Levando em consideração estas peculiaridades do Colab, é possível agora realizar o treinamento das placas de trânsito brasileiras.

A Figura 16 mostra resumidamente o processo de Treinamento do *dataset* de placas de trânsito brasileiras. O processo servirá como um guia para as etapas que serão apresentadas. Destaca-se, ainda, que as etapas para esse processo de treinamento foram baseadas no próprio site oficial do YOLO, sendo que as instruções foram geradas pelo Redmon e Farhadi (2018). Desse modo, o detalhamento do desenvolvimento do "*Guia de Execução do Treinamento de um Dataset no Google Drive*" consta no apêndice A.

[9] <https://colab.research.google.com/notebooks/intro.ipynb#scrollTo=5fCEDCU_qrC0>. Acessado em: 01 de Maio de 2021

[10] Jupyter é um projeto com o propósito de desenvolver *software* de código aberto, por meio de computação interativa em dezenas de linguagens de programação.

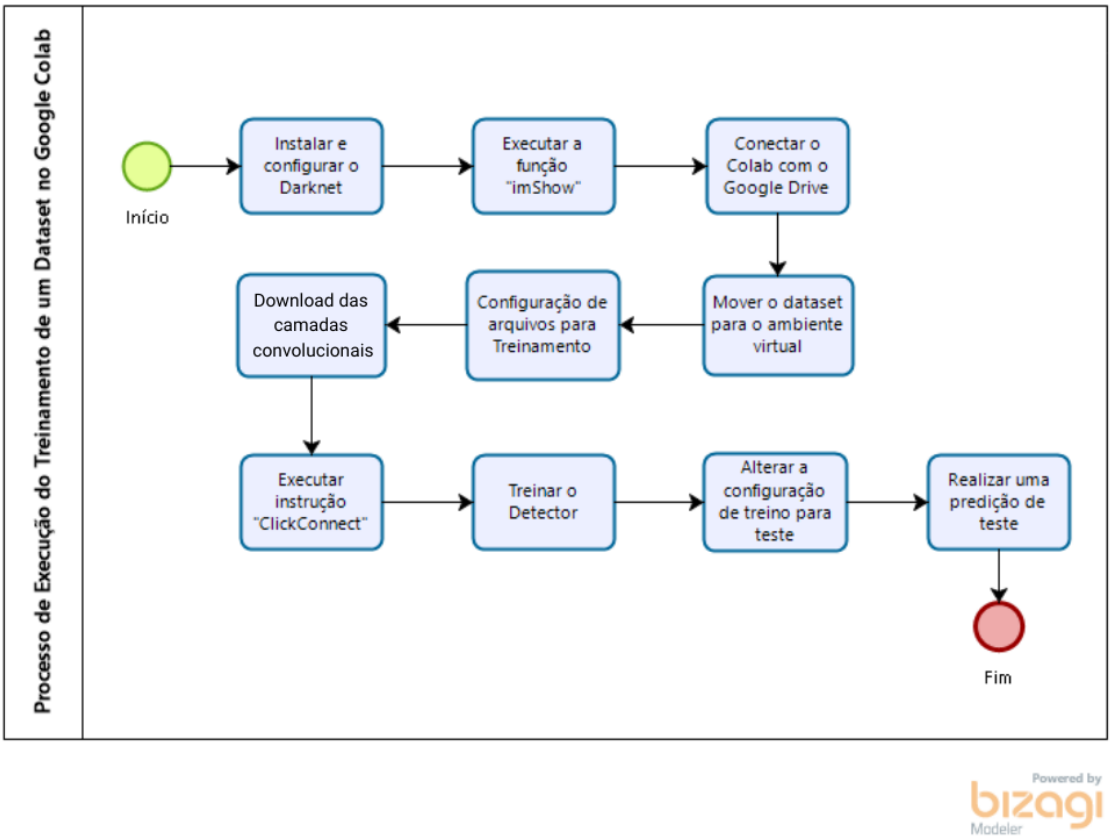


Figura 16 – Processo de execução do treinamento de um *Dataset*.

Fonte: Autor.

Para finalizar a etapa do treinamento, foram gerados dois treinamentos diferentes de um mesmo *dataset* de placas de trânsito brasileiras para este TCC. Eles diferem-se no processo de rotulagem, onde em um dos *datasets* foi realizada a marcação do *bounding box* de toda a placa, sendo chamado de "rotulagem total da placa". No outro caso, foi feita a marcação da *bounding box* apenas no ícone dentro da placa, sendo chamado de "rotulagem do ícone da placa". O exemplo da marcação dessas placas pode ser visto respectivamente na Figura 17.

Figura 17 – Modelos de rotulagem realizado no TCC.

Fonte: Autor.

Sendo assim, na Seção 4.5, será comparada o desempenho dos dois modelos de treinamento de placas de trânsito brasileiras.

4.5 Avaliar Resultados do Treinamento

A avaliação do treinamento utiliza para o cálculo de métrica a matriz de confusão, cujos principais conceitos foram vistos no Capítulo 2 deste trabalho. Para a realização da avaliação, será feito um comparativo de dois treinamentos utilizando o mesmo *dataset*. A sua diferença, como observado na Figura 17, é apenas no processo de rotulagem.

Para que seja possível realizar a avaliação destes treinamentos, é necessário fazer testes. Estes testes são executados da seguinte forma. Para cada treinamento realizado, são executadas várias predições e assim, é feita uma análise da predição. É importante ressaltar que o *dataset* utilizado na fase de testes deve ser diferente da fase de treinamento.

Sendo assim, foram coletadas mais algumas imagens para serem utilizadas como testes para as predições. A coleta das novas imagens de placas de trânsito brasileiras foi realizada da mesma forma que foi vista na seção 4.2. Na Tabela 11, é apresentada a quantidade de imagens de cada placa de trânsito. Ao todo, foram utilizadas 36 imagens para serem feitas as predições.

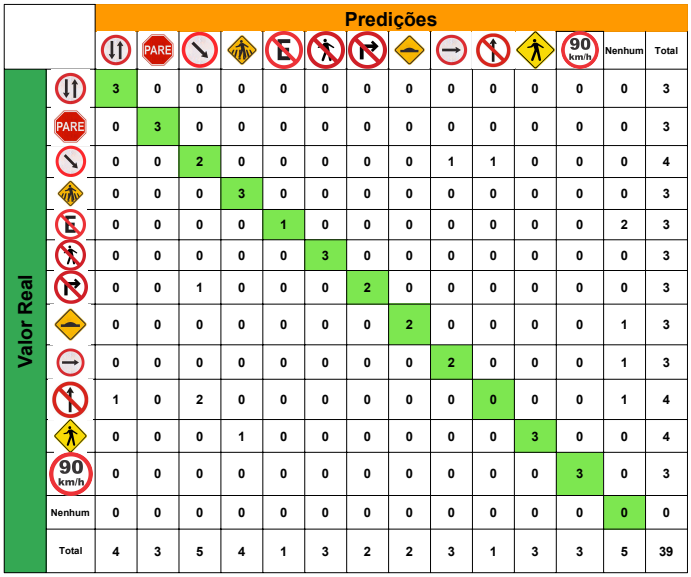
Tanto o modelo de treinamento feito pela "Rotulagem total da placa", quanto o feito pela "Rotulagem do ícone da placa", irão executar o novo *dataset* para que, dessa forma, seja feito um comparativo das métricas. As métricas que foram retiradas após a predição das imagens, e com o auxílio da matriz de confusão, foram: **acurácia**, **precisão**, ***recall***.

Dessa forma, as métricas apresentadas serviram para avaliar o comparativo entre os dois modelos que serão testados.

Tabela 11 – Quantidade de imagens de placas coletadas para realizar a etapa de teste.

Imagens das placas	**Nome das placas**	**Quantidade de Imagens**
	Duplo sentido de circulação	3
	Parada obrigatória	3
	Passagem obrigatória	3
	Passagem sinalizada de pedestres	3
	Proibido estacionar	3
	Proibido trânsito de pedestres	3
	Proibido virar à direita	3
	Saliência ou lombada	3
	Sentido de circulação da via ou pista	3
	Sentido proibido	3
	Trânsito de pedestres	3
	Velocidade máxima permitida	3
	TOTAL	**36**

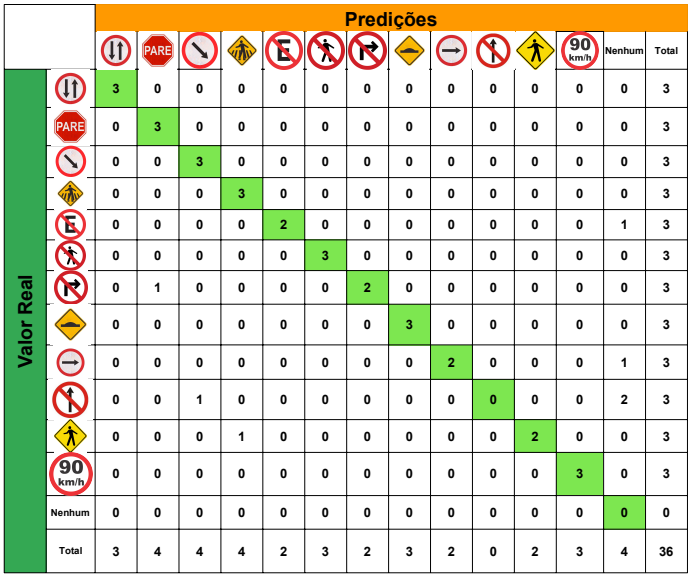
Logo após a realização das predições nos dois modelos de treinamento, é possível inferir a matriz de confusão de cada um deles. O modelo de "Rotulagem total da placa" pode ser visto na Figura 18, já o de "Rotulagem do ícone da placa" é observado na Figura 19. Dessa forma, são concebíveis algumas informações importantes analisando as matrizes. É importante esclarecer que as métricas que serão avaliadas são do modelo como um todo, e não em cada uma de suas categorias.



Valor Real \ Predições													Nenhum	Total
		3	0	0	0	0	0	0	0	0	0	0	0	3
	PARE	0	3	0	0	0	0	0	0	0	0	0	0	3
		0	0	2	0	0	0	0	1	1	0	0	0	4
		0	0	0	3	0	0	0	0	0	0	0	0	3
		0	0	0	0	1	0	0	0	0	0	0	2	3
		0	0	0	0	0	3	0	0	0	0	0	0	3
		0	0	1	0	0	0	2	0	0	0	0	0	3
		0	0	0	0	0	0	0	2	0	0	0	1	3
		0	0	0	0	0	0	0	0	2	0	0	1	3
		1	0	2	0	0	0	0	0	0	0	0	1	4
		0	0	0	1	0	0	0	0	0	0	3	0	4
	90 km/h	0	0	0	0	0	0	0	0	0	0	0	3	0	3
	Nenhum	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Total	4	3	5	4	1	3	2	2	3	1	3	3	5	39

Figura 18 – Matriz de confusão do treinamento da “Rotulagem total da placa”.

Fonte: Autor.



Valor Real \ Predições													Nenhum	Total
		3	0	0	0	0	0	0	0	0	0	0	0	3
	PARE	0	3	0	0	0	0	0	0	0	0	0	0	3
		0	0	3	0	0	0	0	0	0	0	0	0	3
		0	0	0	3	0	0	0	0	0	0	0	0	3
		0	0	0	0	2	0	0	0	0	0	0	1	3
		0	0	0	0	0	3	0	0	0	0	0	0	3
		0	1	0	0	0	0	2	0	0	0	0	0	3
		0	0	0	0	0	0	0	3	0	0	0	0	3
		0	0	0	0	0	0	0	0	2	0	0	1	3
		0	0	1	0	0	0	0	0	0	0	0	2	3
		0	0	0	1	0	0	0	0	0	0	2	0	0	3
	90 km/h	0	0	0	0	0	0	0	0	0	0	0	3	0	3
	Nenhum	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Total	3	4	4	4	2	3	2	3	2	0	2	3	4	36

Figura 19 – Matriz de confusão do treinamento da “Rotulagem do ícone da placa”.

Fonte: Autor.

Começando pela matriz gerada pela “Rotulagem total da placa”. A diagonal principal, que está destacada de verdade na figura da matriz de confusão, representa os *verdadeiros positivos*, que são aqueles em que as predições correspondem ao valor real, sendo assim, o “TP = 24”.

Todos os que estão fora da diagonal principal são os *verdadeiros negativos*, já que

o modelo detectou uma placa de trânsito, mas errou a sua categoria, dessa maneira, o "TN = 7".

Onde está descrito "Nenhum" na linha do valor real representa os *falsos positivos*, que, por sua vez, ocorrem quando não há nada na imagem, mas o modelo detecta algo, mesmo não tendo nada, dessa forma, o "FP = 0".

Para finalizar, o local em que está escrito "Nenhum", na coluna das predições, representa os *falsos negativos*, são aqueles nos quais existia uma placa de trânsito na imagem, mas o modelo não detectou nada, sendo assim, o "FN = 5".

Agora, com as variáveis disponíveis é possível fazer as inferências das métricas de acurácia, precisão e *recall*. Suas respectivas fórmulas encontram-se no Capítulo 2. Sendo assim, o resultado de cada uma das métricas é: "acurácia = 86%", "precisão = 100%" e "*recall* = 82%".

Da mesma forma, as métricas são inferidas para o modelo de treinamento "Rotulagem do ícone da placa". Toda análise referente à "Rotulagem total da placa" aplica-se na matriz de confusão do "Rotulagem do ícone da placa". Sendo assim, os valores das variáveis são: "TP = 27", "TN = 3", "FP = 0" e "FN = 4". É possível agora calcular as métricas, sendo: "acurácia = 88% ", "precisão = 100%" e "*recall* = 87%".

Comparando as métricas dos dois modelos de treinamento, é possível notar que a "Rotulagem do ícone da placa" tem uma taxa de acerto mais alta que a "Rotulagem total da placa", apesar da diferença ser miníma. Ainda assim, os dois modelos são passíveis a serem utilizados.

4.6 Aplicação do modelo de treinamento em vídeos

Tendo um modelo de treinamento para detectar e identificar placas de trânsito brasileiras, o próximo passo que este trabalho abordou foi aplicar o modelo de treinamento em diferentes cenários. O cenário, como visto na secção de 3.1.4, nada mais é que uma narrativa de uma situação que envolve um ambiente, variáveis e dados. Seu uso mostra-se necessário pelo principal motivo de que, segundo Kun et al. (2020), os estudos na área de detecção e reconhecimento de sinais de trânsito, em sua maioria, não consideram as condições de baixa iluminação na aplicação prática, ou possíveis acontecimentos comuns no tráfego de trânsito.

Dito isso, o objetivo central desta seção, é mostrar cenários que os brasileiros passam durante o tráfego de trânsito, além de expor o modelo de treinamento a diferentes situações, seja por questões de iluminação, volume de carros na pista, clima, entre outros. Para isso, serão apresentados 4 cenários gravados em vídeo para que o modelo treinado possa detectar as placas de trânsito. Os cenários são: *às 12 horas do dia, ao pôr do sol às 18 horas e 30 minutos, a noite às 20 horas e por fim em um dia de chuva às 16 horas.*

Para a filmagem dos cenários, alguns pontos foram pré-definidos. Em todos os cenários, foi utilizado o mesmo percurso, porém gravado em dias e momentos diferentes. O percurso gravado tem a distância de 5 quilômetros. O percurso foi escolhido de um modo que consiga contemplar o máximo de placas possíveis escolhidas para este TCC. Na Figura 20, é possível observar todas as placas que estão contidas no percurso proposto.

	PLACAS												
		PARE										90 km/h	Total
QUANTIDADE	2	2	1	31	2	0	1	8	0	0	5	22	74

Figura 20 – Quantidade de placas de trânsito contidas no percurso.

Fonte: Autor.

O equipamento usado para gravar os vídeos foi um telefone celular, utilizando a câmera traseira de 12 megapixels, na posição horizontal. As configurações de filmagem foram Full HD 1080p gravado a 60 fps[11]. Além disso, as gravações foram feitas dentro de um veículo, onde o celular estava preso ao retrovisor[12] interno do carro.

[11] Fotos por segundo.

[12] É um espelho encontrado no exterior e interior de veículos automotivos para auxiliar o motorista a enxergar áreas para trás e para os lados do veículo.

À princípio, o celular estava preso por um suporte de celular colocado no para-brisa[13], mas os modelos utilizados acabavam desprendendo do para-brisa, o que dificultava as gravações. Desse modo, foi feita uma base improvisada para o celular, que pode ser observada na Figura 21. O posicionamento, dessa forma, ficou perfeito, pois ficou centralizado e, assim, a câmera conseguiu uma visão ampla para a gravação.

Figura 21 – Base improvisada para o celular.

Fonte: Autor.

Após a gravação dos vídeos, foi necessário fazer com que o modelo de treinamento realizasse a detecção e identificação das placas. Para isso, faz-se uso mais uma vez do Colab. A primeira etapa foi copiar o vídeo do google drive para a máquina virtual do Colab, bastando utilizar o código:

```
$ !cp /mydrive/video-yolo/chuva.mp4 ./
```

Neste exemplo, o vídeo do cenário de dia de chuva foi copiado para o ambiente do Colab. Após isso, executa-se o seguinte comando:

```
$ with open('chuvaSaida.txt', 'w') as arquivo:
    saida = !./darknet detector demo data/obj.data cfg/
       yolov3_custom.cfg /mydrive/yolov3/backup/
       yolov3_custom_last.weights -dont_show /mydrive/
       video-yolo/chuva.mp4 -out_filename resultado-chuva
       .avi -thresh 0.5
print(saida, file=arquivo)
```

[13] é um vidro colocado na parte da frente dos automóveis ou de aeronaves, oferecendo ao condutor uma proteção.

Esse comando faz com que o modelo comece a detectar as placas no vídeo, e, ao final, foi geraro um vídeo resultado com as *bounding boxes*. Além disso, a saída gerada pelo modelo é passada para um arquivo. A saída consiste na leitura de *frame* a *frame* do vídeo, e mostra os objetos encontrados no *frame* visitado. Um exemplo de saída pode ser visto na Figura 22. Na figura, é possível observar que, em alguns dos *frames*, foram encontrados algumas categorias de placas de trânsito.

```
FPS:22.0        AVG_FPS:20.9

 cvWriteFrame
Objects:

double_direction_of_circulation_sign: 87%

FPS:21.4        AVG_FPS:20.9

 cvWriteFrame
Objects:

no_parking_sign: 55%

FPS:21.3        AVG_FPS:20.9

 cvWriteFrame
Objects:

no_parking_sign: 65%

FPS:20.6        AVG_FPS:20.9

 cvWriteFrame
Objects:

no_parking_sign: 84%

FPS:20.1        AVG_FPS:20.9

 cvWriteFrame
Objects:
```

Figura 22 – Exemplo do arquivo de saída com a detecção de placas por *frame*

Fonte: Autor.

Por fim, basta executar:

```
$ !cp ./resultado-chuva.avi /mydrive/video-yolo/resultado
   -chuva.avi
$ !cp ./chuvaSaida.txt /mydrive/video-yolo/chuvaSaida.txt
```

Este comando faz com que o resultado, ou seja, o vídeo gerado com as *bounding boxes* e o arquivo de saída que contém os *frames* sejam copiados para o drive, e, assim, serem salvos para visualização. Caso queira visualizar os vídeos gravados bem como seus resultados, basta acessar o Drive[14].

Agora, é possível realizar a análise de cada cenário coletado. Entretanto, antes, é necessário ressaltar que foram aplicados alguns critérios de análise. O primeiro é que

[14] (<https://drive.google.com/drive/folders/1WNBrLmPv4jKydyNGwAzcHh-Uecl_xJ3R?usp=sharing>)

apenas aparecerá no vídeo de resultado as *bounding boxes* que possuem uma precisão superior a 50% de acerto. Sendo assim, as detecções feitas com baixa precisão serão ignoradas.

O outro critério estabelecido envolve o arquivo de saída gerado, sendo assim, serão consideradas como uma detecção do modelo de treinamento apenas as identificações feitas com no mínimo 3 *frames* respectivos de uma mesma categoria de placa. Isso evita que o modelo reconheça objetos aleatórios em que é identificado apenas por um *frame*. Dito isso, é possível realizar as análises de cada cenário coletado.

4.6.1 Às 12 horas do dia

Este cenário foi gravado ao meio dia (12 horas). Em suma, este deveria ser o melhor cenário possível para detecção das placas de trânsito. Isso se dá por conta da melhor iluminação da luz solar, visto que o sol está posicionado ao mais alto possível. E neste dia em específico da gravação estava ensolarado, como pode ser observado na Figura 23.

Figura 23 – Exemplo do cenário ao meio-dia.

Fonte: Autor.

Dessa forma, as métricas de acurácia, precisão, *recall* foram extraídas da matriz de confusão, na qual os dados foram coletados a partir do vídeo resultado do cenário. A matriz de confusão do cenário de *12 horas do dia* está apresentada na Figura 24.

		Predições													
			PARE										90 km/h	Nenhum	Total
Valor Real		1	0	0	0	0	0	0	0	0	0	0	1	0	2
	PARE	0	2	0	0	0	0	0	0	0	0	0	0	0	2
		0	0	1	0	0	0	0	0	0	0	0	0	0	1
		0	0	0	28	0	0	0	2	0	0	0	0	0	30
		0	0	0	0	2	0	0	0	0	0	0	1	0	3
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	1	1
		0	0	0	1	0	0	0	8	0	0	0	0	0	9
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	3	0	0	0	0	0	0	5	0	0	8
	90 km/h	0	0	0	0	0	0	0	0	0	0	0	17	4	21
	Nenhum	0	1	0	2	0	0	0	3	0	0	0	2	0	8
	Total	1	3	1	34	2	0	0	13	0	0	5	21	5	85

Figura 24 – Matriz de confusão do cenário ao meio-dia.

Fonte: Autor.

Na Figura 24, a diagonal principal, que está destacada de verdade na figura da matriz de confusão, representa os *verdadeiros positivos*, que são aqueles em que as predições correspondem ao valor real, sendo assim, o "TP = 64".

Todos os que estão fora da diagonal principal são os *verdadeiros negativos*, já que o modelo detectou uma placa de trânsito, mas errou a sua categoria, dessa maneira, o "TN = 8".

Onde está descrito "Nenhum" na linha do valor real representa os *falsos positivos*, que, por sua vez, ocorrem quando não há nada na imagem, mas o modelo detecta algo, mesmo não tendo nada, dessa forma, o "FP = 8".

Para finalizar, o local em que está escrito "Nenhum", na coluna da predições, representa os *falsos negativos*, são aqueles nos quais existia uma placa de trânsito na imagem, mas o modelo não detectou nada, sendo assim, o "FN = 5".

Agora, com as variáveis disponíveis, é possível fazer as inferências das métricas de acurácia, precisão e *recall*. Suas respectivas fórmulas encontram-se no Capítulo 2. Sendo assim, o resultado de cada uma das métricas é: "acurácia = 84%", "precisão = 88%" e "*recall* = 92%".

4.6.2 Ao pôr do sol às 18 horas e 30 minutos

Este cenário foi gravado às 18 horas e 30 minutos ao pôr do sol. Este cenário, em específico, possui uma peculiaridade sobre a sua iluminação, quando o veículo está em direção ao sol ou na direção contrária ao sol. Quando o veículo está em direção ao sol, a iluminação para a gravação do vídeo fica comprometida, já que o ambiente escure. Essa situação pode ser vista na Figura 25. Já quando o veículo está na direção contrária ao sol, o ambiente fica mais adequado para a gravação, sendo a melhor possibilidade possível, visto que a luz natural do sol está de frente para a placa. Esta situação pode ser conferida na Figura 26.

Figura 25 – Exemplo do cenário ao pôr do sol, com o veículo em direção ao sol.

Fonte: Autor.

Figura 26 – Exemplo do cenário ao pôr do sol, com o veículo na direção contrária ao sol.

Fonte: Autor.

Sabendo disso, as métricas de acurácia, precisão, *recall* serão extraídas da matriz

de confusão, na qual os dados foram coletados a partir do vídeo resultado do cenário. A matriz de confusão do cenário do *pôr do sol às 18 horas e 30 minutos* está apresentado na Figura 27.

Valor Real \ Predições													Nenhum	Total
	1	0	0	0	0	0	0	0	0	0	0	0	1	2
PARE	0	2	0	0	0	0	0	0	0	0	0	0	0	2
	0	0	1	0	0	0	0	0	0	0	0	0	0	1
	0	0	0	23	0	0	0	2	0	0	0	0	4	29
	0	0	0	0	2	0	0	0	0	0	0	0	0	2
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	1	1
	0	0	0	0	0	0	0	5	0	0	0	0	3	8
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	3	0	1	4
90 km/h	0	0	0	0	0	0	0	0	0	0	0	12	10	22
Nenhum	0	1	0	4	0	0	0	0	0	0	0	0	0	5
Total	1	3	1	27	2	0	0	7	0	0	3	12	20	76

Figura 27 – Matriz de confusão do cenário ao pôr do sol.

Fonte: Autor.

Na Figura 27, a diagonal principal, que está destacada de verdade na figura da matriz de confusão, representa os *verdadeiros positivos*, que são aqueles em que as predições correspondem ao valor real, sendo assim, o "TP = 48".

Todos os que estão fora da diagonal principal são os *verdadeiros negativos*, já que o modelo detectou uma placa de trânsito, mas errou a sua categoria, dessa maneira, o "TN = 2".

Onde está descrito "Nenhum" na linha do valor real representa os *falsos positivos*, que, por sua vez, ocorrem quando não há nada na imagem, mas o modelo detecta algo, mesmo não tendo nada, dessa forma, o "FP = 5".

Para finalizar, o local em que está escrito "Nenhum", na coluna das predições, representa os *falsos negativos*, são aqueles nos quais existia uma placa de trânsito na imagem, mas o modelo não detectou nada, sendo assim, o "FN = 20".

Agora, com as variáveis disponíveis é possível fazer as inferências das métricas de acurácia, precisão e *recall*. Suas respectivas fórmulas encontram-se no Capítulo 2. Sendo assim, o resultado de cada uma das métricas é: "acurácia = 66%", "precisão = 90%" e

"$recall = 70\%$".

4.6.3 A noite às 20 horas

O cenário gravado às 20h tem suas particularidades, uma delas é o ambiente estar escuro. Como consequência disso, as luzes dos "postes de luz" das avenidas são ligadas, e isso faz com que gere um feixe de luz no vídeo, como pode ser visto na Figura 28. Um outro ponto é que, devido ao alto teor de luz em alguns trechos do percurso, a câmera cria um desfoque, que, por sua vez, impossibilita a detecção de placas de trânsito naquele momento. Essa situação pode ser vista na Figura 29.

Figura 28 – Exemplo do cenário noite às 20 horas com iluminação artificial dos postes de eletricidade.

Fonte: Autor.

Figura 29 – Exemplo do cenário noite às 20 horas com desfoque.

Fonte: Autor.

Dessa maneira, as métricas de acurácia, precisão, *recall* serão extraídas da matriz de confusão, na qual os dados foram coletados a partir do vídeo resultado do cenário. A matriz de confusão do cenário de *noite às 20 horas* está apresentado na Figura 30.

		Predições													
														Nenhum	Total
Valor Real		1	0	0	0	0	0	0	0	0	0	0	1	0	2
		0	1	0	0	0	0	0	0	0	0	0	0	1	2
		0	0	1	0	0	0	0	0	0	0	0	0	1	2
		0	0	0	24	0	0	0	6	0	0	0	0	5	35
		0	0	0	0	2	0	0	0	0	0	0	0	0	2
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	1	0	0	0	0	0	0	0	0	1
		0	0	0	1	0	0	0	7	0	0	0	0	1	9
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	3	0	0	0	0	0	0	4	0	0	7
		0	0	2	0	1	0	0	0	0	0	0	12	9	24
	Nenhum	0	1	0	1	0	0	0	0	3	0	0	0	0	5
	Total	1	2	3	29	4	0	0	13	3	0	4	13	17	89

Figura 30 – Matriz de confusão do cenário à noite.

Fonte: Autor.

Na Figura 30, a diagonal principal, que está destacada de verdade na figura da matriz de confusão, representa os *verdadeiros positivos*, que são aqueles em que as predições correspondem ao valor real, sendo assim, o "TP = 52".

Todos os que estão fora da diagonal principal são os *verdadeiros negativos*, já que o modelo detectou uma placa de trânsito, mas errou a sua categoria, dessa maneira, o "TN = 15".

Onde está descrito "Nenhum" na linha do valor real representa os *falsos positivos*, que, por sua vez, ocorrem quando não há nada na imagem, mas o modelo detecta algo, mesmo não tendo nada, dessa forma, o "FP = 5".

Para finalizar, o local em que está escrito "Nenhum", na coluna das predições, representa os *falsos negativos*, são aqueles nos quais existia uma placa de trânsito na imagem, mas o modelo não detectou nada, sendo assim, o "FN = 17".

Agora, com as variáveis disponíveis é possível fazer as inferências das métricas de acurácia, precisão e *recall*. Suas respectivas fórmulas encontram-se no Capítulo 2. Sendo assim, o resultado de cada uma das métricas é: "acurácia = 75%", "precisão = 91%" e

"*recall* = 75%".

4.6.4 Um dia de chuva às 16 horas

Por fim, o cenário da chuva às 16 horas. Neste cenário, a visibilidade é bem comprometida devido ao clima do ambiente, e principalmente às gotas de chuva que batem no para-brisa. A situação pode ser vista na Figura 31. Essas gotas acabam distorcendo a imagem, fazendo com que o modelo de treinamento detecte objetos que não são placas de trânsito com uma maior frequência.

Figura 31 – Exemplo do cenário de chuva às 16 horas.

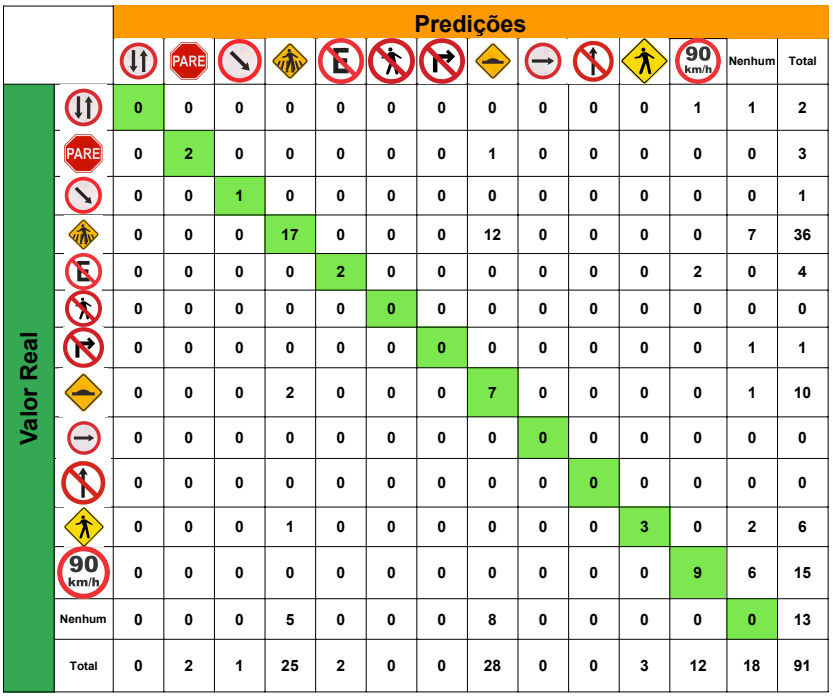
Fonte: Autor.

Dessa maneira, as métricas de acurácia, precisão, *recall* serão extraídas da matriz de confusão, na qual os dados foram coletados a partir do vídeo resultado do cenário. A matriz de confusão do cenário de *um dia de chuva às 16 horas* está apresentado na Figura 32.

Na Figura 32, a diagonal principal, que está destacada de verdade na figura da matriz de confusão, representa os *verdadeiros positivos*, que são aqueles em que as predições correspondem ao valor real, sendo assim, o "TP = 41".

Todos os que estão fora da diagonal principal são os *verdadeiros negativos*, já que o modelo detectou uma placa de trânsito, mas errou a sua categoria, dessa maneira, o "TN = 19".

Onde está descrito "Nenhum" na linha do valor real representa os *falsos positivos*, que, por sua vez, ocorrem quando não há nada na imagem, mas o modelo detecta algo, mesmo não tendo nada, dessa forma, o "FP = 13".



Valor Real \ Predições														
	0	0	0	0	0	0	0	0	0	0	0	1	1	2
PARE	0	2	0	0	0	0	0	1	0	0	0	0	0	3
	0	0	1	0	0	0	0	0	0	0	0	0	0	1
	0	0	0	17	0	0	0	12	0	0	0	0	7	36
	0	0	0	0	2	0	0	0	0	0	0	2	0	4
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	1	1
	0	0	0	2	0	0	0	7	0	0	0	0	1	10
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	0	0	0	3	0	2	6
90 km/h	0	0	0	0	0	0	0	0	0	0	0	9	6	15
Nenhum	0	0	0	5	0	0	0	8	0	0	0	0	0	13
Total	0	2	1	25	2	0	0	28	0	0	3	12	18	91

Figura 32 – Matriz de confusão do cenário de chuva.

Fonte: Autor.

Para finalizar, o local em que está escrito “Nenhum”, na coluna das predições, representa os *falsos negativos*, são aqueles nos quais existia uma placa de trânsito na imagem, mas o modelo não detectou nada, sendo assim, o “FN = 18”.

Agora, com as variáveis disponíveis é possível fazer as inferências das métricas de acurácia, precisão e *recall*. Suas respectivas fórmulas encontram-se no Capítulo 2. Sendo assim, o resultado de cada uma das métricas é: “acurácia = 65%”, “precisão = 75%” e “*recall* = 69%”.

4.6.5 Considerações dos Cenários

Após a coleta de métricas dos cenários, é possível observar algumas informações. A Tabela 12 apresenta um quadro comparativo entre as métricas de cada cenário, juntamente com os resultados obtidos na Seção 4.5, em especial o modelo de “Rotulagem total da placa”. Nesse modelo, é perceptível que ele possui os melhores resultados, pois ele não foi submetido a diversas situações, sendo assim ele é o resultado de referência para essa pesquisa.

Comparando os cenários ao modelo de “Rotulagem total da placa”, observa-se que as métricas coletadas sofreram alteração de desempenho. Os cenários gravados em conjunto com suas características impactam diretamente nos resultados alcançados.

Tabela 12 – Comparativo das métricas de cada cenário.

Cenários	**Acurácia**	**Precisão**	***Recall***
Rotulagem total da placa	88%	100%	87%
às 12 horas do dia	84%	88%	92%
ao pôr do sol às 18 horas e 30 minutos	66%	90%	70%
a noite às 20 horas	75%	91%	75%
um dia de chuva às 16 horas	65%	75%	69%

O cenário de *às 12 horas do dia* foi o que demonstrou ser o mais constante, já que o ambiente em que foi gravado o trajeto está iluminado pela luz do sol de forma homogênea, e isso facilita a detecção das placas de trânsito. Já no cenário do *pôr do sol às 18 horas e 30 minutos*, o resultado deu-se pelo motivo principal da iluminação do sol, quando o ambiente está em direção ao sol, a detecção das placas de trânsito é muito efetiva. Porém, se o ambiente estiver na direção contrária ao sol, o modelo tem muita dificuldade para identificar as placas de trânsito.

No cenário de *noite às 20 horas*, a redução da sua acurácia e precisão ocorreu por conta da iluminação não natural, dos postes de luz, da faixa de lojas. E isso faz com que a câmera desfoque frequentemente durante a gravação. Por fim, no cenário de *um dia de chuva às 16 horas*, houve uma redução muito significativa nos seus resultados pelo motivo de que as gotas de água que batem no para-brisa acabam deformando a imagem fazendo com que o modelo de treinamento identifique placas de trânsito onde não existe.

Dado os resultados de cada cenário, pode-se concluir que diferentes ambientes e situações podem interferir diretamente no modelo de detecção de placas de trânsito. Além das dificuldades já apresentadas, existem diversas outras restrições neste trabalho. Uma delas são os equipamentos de trabalho, visto que, com uma câmera de melhor qualidade, seria possível coletar uma filmagem mais nítida e limpa. A qualidade de gravação pode fazer com que os resultados dos cenários melhorem consideravelmente.

Para além de restrições de equipamentos, existem as questões de interferência em um ambiente de trânsito. Dentre essas situações, pode ocorrer que alguns veículos no trânsito interfiram na visão da placa. Ou que, como mostrado na Figura 33 mostra, placas danificadas atrapalhem o modelo a realizar a detecção. Ou ainda, a placa de trânsito pode estar envelhecida, ou seu conteúdo interno pode estar alterado ou apagado, seja por ação do ser humano ou do ambiente, como visto na Figura 34.

Figura 33 – Exemplo placa de trânsito danificada.

Fonte: Autor.

Figura 34 – Exemplo de placa de trânsito alterada pela ação do tempo.

Fonte: Autor.

Ainda existem situações em que a placa de trânsito está mal posicionada, ou simplesmente não existe, como pode ser visto na Figura 35, que mostra uma placa de "proibido estacionar" ao lado de uma lombada, e não há sinalização que indique que existe uma lombada nessa região em específico. Adicionalmente, existe a situação em que a placa de trânsito está obstruída por árvores, plantas ou algo relacionado à força da natureza, como podemos ver na Figura 36.

Figura 35 – Exemplo de placa de trânsito mal posicionada.

Fonte: Autor.

Figura 36 – Exemplo de placa de trânsito obstruída por plantas.

Fonte: Autor.

Dessa forma, existem dezenas de variáveis que devem ser levadas em consideração para que o modelo de treinamento consiga realizar de fato a detecção e identificação de placas de trânsito em uma condição de implementação para veículos autônomos reais.

4.7 Distância máxima do reconhecimento do modelo treinado

Primeiramente, os dados que serão apresentados são imprecisos devido a diversas variáveis, por exemplo, processos manuais e ferramentas de medição imprecisas. Sendo assim, as informações observadas nessa seção tem como objetivo mostrar, de forma básica e miníma, como o modelo de treinamento se comporta, em relação à distância da placa

e ao primeiro momento de reação que o modelo teve para identificar de forma correta a categoria da placa de trânsito.

Dito isso, a pesquisa sucedeu a partir do momento em que o modelo de treinamento realizou a identificação das placas em vídeos, apresentados na subseção 4.6 Aplicação do modelo de treinamento em vídeos. Dessa forma, a intenção é saber qual a distância do momento da detecção até a placa de trânsito identificada.

De acordo com Zhang et al. (2020), a detecção de longa distância de sinais de trânsito fornece um maior tempo de reação, o que é algo eficaz para reduzir a probabilidade de acidentes repentinos. Imagine um veículo autônomo, que necessita da identificação de placas de trânsito para realizar ações. Neste cenário, o veículo está em uma avenida e depara-se com uma placa de lombada, que indica a existência de um "quebra-mola" [15] próximo a placa. O veículo autônomo que transita a uma certa velocidade deve identificar a placa de trânsito com antecedência para que consiga reduzir a velocidade ao passar na lombada. Caso o veículo apenas reconheça a placa a uma curta distância, ele não terá um tempo suficiente para realizar a ação de redução de velocidade, e, assim, podendo provocar um acidente, ferir as pessoas dentro do veículo ou comprometer o próprio veículo.

Dessa forma, para seguir o desenvolvimento do trabalho, foram utilizados como simulação os cenários de testes coletados na subseção 4.6 Aplicação do modelo de treinamento em vídeos. Sendo mais específico, foi avaliado um evento retirado do trajeto proposto recortado do vídeo. A situação a ser observada pode ser encontrada na Figura 37. Nela, é possível visualizar uma placa de lombada e a distância que o veículo conseguiu detectar o primeiro *frame* da placa de trânsito.

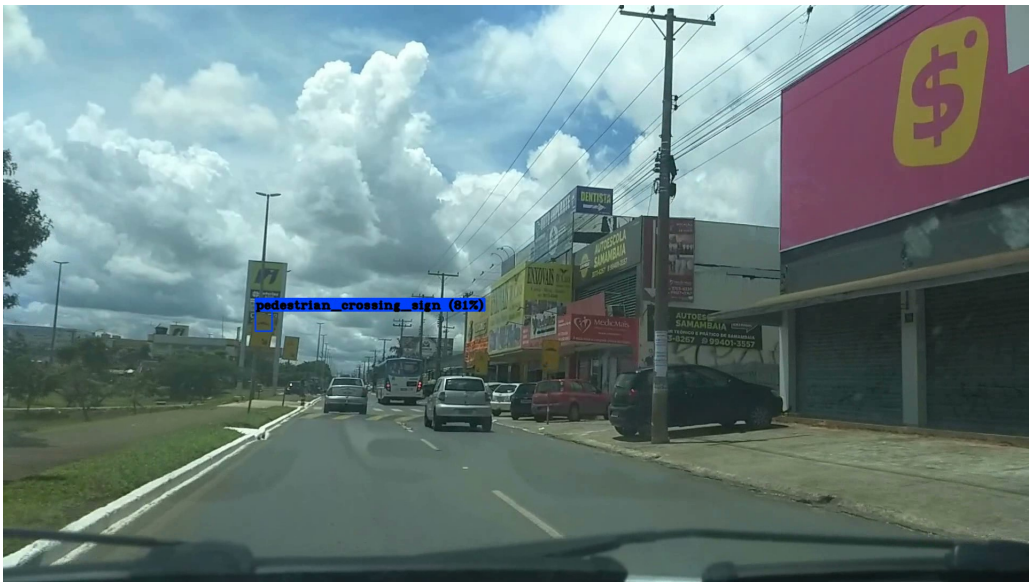


Figura 37 – Trecho do percurso que será submetido a teste.

Fonte: Autor.

[15] Lombada construída para diminuir velocidade de veículos.

O objetivo neste primeiro momento é descobrir qual a distância da placa de trânsito até o veículo a partir do momento que houve a detecção da placa. Para isso, foram utilizados um método de medição de distância. Este foi através do Google Maps[16], ele é capaz, através de imagens de satélite, pesquisar e visualizar mapas. Sendo assim, o google maps possui uma funcionalidade de medir distâncias em seu aplicativo, como pode ver na Figura 38. Ele possibilita marcar dois pontos no mapa e medir a distância entre esses pontos. Desse modo, foi marcado no mapa o ponto da placa de trânsito e o ponto que o veículo passou a reconhecer a placa de trânsito.

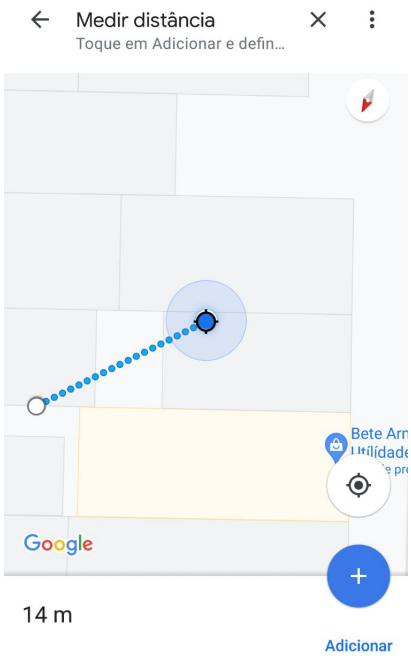


Figura 38 – Exemplo de como é feita a medição da distância entre 2 pontos no aplicativo do google maps.

Fonte: Autor.

É válido relembrar que o método de coleta da distância é aproximada, pois é impreciso. E que a coleta foi realizada para os 4 cenários: *às 12 horas do dia, ao pôr do sol às 18 horas e 30 minutos, a noite às 20 horas e por fim em um dia de chuva às 16 horas*. O resultado da coleta da distância pode ser visualizado na Tabela 13.

[16] https://www.google.com.br/maps

Tabela 13 – Resultado da medição das distâncias entre a placa e o momento que o modelo detectou a placa.

Cenários	**Google Maps**
às 12 horas do dia	15 metros
ao pôr do sol às 18 horas e 30 minutos	32 metros
a noite às 20 horas	11 metros
um dia de chuva às 16 horas	10 metros

De acordo com a RESOLUÇÃO Nº 600, DE 24 DE MAIO DE 2016, regulamentada pelo Conselho Nacional de Trânsito CONTRAN (2016), existem dois tipos de quebra mola. O do tipo A é mais suave, com 3,7 metros de comprimento, e reduz a velocidade de um veículo a 30 km/h. Já o tipo B é mais angular, com 1,5 metro de comprimento, reduzindo a velocidade para 20 km/h. Na simulação que está sendo analisada, o quebra-mola se enquadra no Tipo A. Com essa informação, sabemos que o veículo deve chegar com a velocidade de no mínimo 30 km/h para realizar uma travessia recomendada em uma lombada.

Ainda, sabendo que o veículo estava sendo conduzido a 50 km/h em todos os cenários. Dessa forma, é necessário calcular o tempo de reação, ou seja, o tempo de redução de velocidade, de acordo com a distância coletada entre a placa e o momento da identificação. Na Figura 39, é possível encontrar uma exemplificação do cenário.

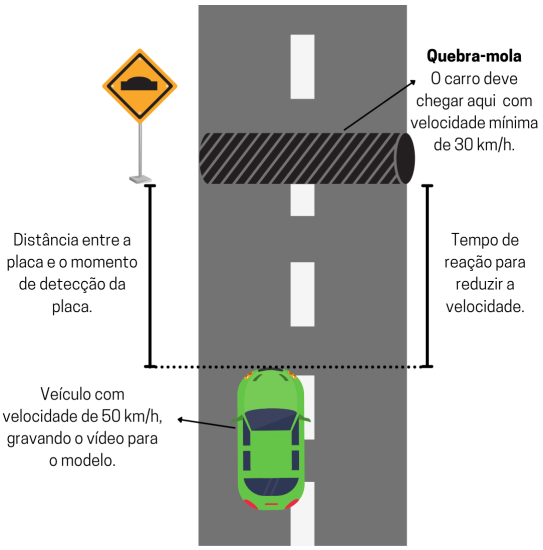


Figura 39 – Exemplificação do cenário retratado.

Fonte: Autor.

Para realizar este cálculo será necessária a equação de Torricelli. Pois é a partir

dela que iremos descobrir a aceleração do veículo, ou, neste caso, a desaceleração, visto que o veículo tende a perder velocidade. A Equação 4.1 consiste em:

$$V^2 = V_o^2 + 2.a.d \tag{4.1}$$

Onde:

- V é a velocidade final;
- Vo é a velocidade inicial;
- a é a aceleração;
- d é a distância percorrida pelo móvel;

Após o calculo, e com a aceleração em mãos, é possível determinar o tempo em que o veículo se locomoveu até o quebra-mola. Para isso, foi utilizada a seguinte Equação 4.2. Onde a variável t representa a grandeza do tempo.

$$V = V_o + a.t \tag{4.2}$$

Por fim, com todos os dados necessários, basta calcular a velocidade média, que irá ter como resposta a variação da velocidade do carro até o quebra-mola, ou seja, a velocidade final reduzida pela aceleração durante a distância coletada na Tabela 13. Sendo assim, basta utilizar a Equação 4.3.

$$V_M = \frac{d}{t} \tag{4.3}$$

O resultado para cada cenário pode ser observado na Tabela 14.

Tabela 14 – Resultado da velocidade final em que o veículo chegou na lombada.

Cenários	**Google Maps - velocidade com a qual o veículo chega na lombada**
às 12 horas do dia	39,9 km/h
ao pôr do sol às 18 horas e 30 minutos	39,8 km/h
a noite às 20 horas	40,3 km/h
um dia de chuva às 16 horas	40,4 km/h

Com os resultados obtidos, é notável que a média geral de todos os cenários seja semelhante, e tende a uma média de 40 km/h. Sendo assim, dada a situação abordada, podemos concluir que o veículo não conseguiria chegar a velocidade miníma desejada, ou seja, a distância na qual o modelo reconhece a placa de trânsito é muito curta, e não dá um tempo de reação suficiente para desacelerar a velocidade até o recomendado, que é 30 km/h.

Ainda assim, este é um resultado aproximado que depende de diversas variáveis, seja por equipamentos de medições adequados para o contexto, ou até por condições climáticas e iluminação. Para consolidar de forma mais adequada o resultado, seria necessário realizar vários experimentos para, assim, aferir dados mais consistentes e poder de fato realizar conclusões.

Dessa forma, é importante ressaltar que esse resultado não contempla todo o modelo de detecção, e sim, somente essa situação retirada do trajeto gravado para os cenários. Neste pequeno espaço amostral, o objetivo é mostrar a importância de treinar um modelo que possibilite o reconhecimento de placas de trânsito a uma distância adequada, tornando, assim, a reação do veículo satisfatória.

Por fim, durante este capítulo, foram apresentadas etapas, juntamente com os conceitos e a parte técnica da solução desenvolvida durante a pesquisa, para que, no fim, o processo esteja o mais funcional possível, sendo capaz de alcançar os objetivos do trabalho.

Parte IV

Conclusão

5 Conclusão

Percebe-se que, no desenvolvimento da Seção 4 - Resultados, alguns fatos podem ser concretizados em relação aos objetivos propostos para esse trabalho. Retomando o que visava-se alcançar, pode-se observar nos objetivos específicos:

- **Definir um processo de rotulagem e treinamento do YOLOv3 para que outros pesquisadores sejam capazes de recriar as etapas da pesquisa, e assim conseguir detectar e classificar placas de trânsito brasileiras.**

  Neste objetivo específico, podemos ver claramente a sua realização nas Seções 4.3 - Rotulagem e 4.4 - Executar treinamento, visto que existe um processo que pode e deve ser replicado para outros pesquisadores, já que neste trabalho foram abordados apenas 12 modelos de placas de trânsito brasileiras, o que é equivalente a cerca de 10% do total de placas pertinentes para os veículos autônomos. Sendo assim, isso pode ser facilmente expandido, uma vez que a legislação de trânsito brasileira possui vários modelos diferentes de placas de trânsito.

  Dito isso, há uma preocupação pertinente em relação à desvantagem do YOLO. Nesse sentido, quanto mais classes possuírem o modelo de treinamento, mais demorada será a execução do treinamento, mais complexo será o modelo, e mais tempo será utilizado para realizar a classificação do objeto, além de que é possível que haja uma redução da acurácia do modelo de treino. Dessa maneira, é preciso refinar ao máximo as classes que estarão presentes no *dataset* de treino.

  Finalizando esse objetivo específico, é ainda importante ressaltar um benefício alcançado. O processo de rotulagem e treinamento realizado neste trabalho pode ser replicável também para outros problemas que envolvam detecção e classificação de objetos.

- **Determinar um conjunto de métricas que poderão indicar a qualidade de detecção e classificação das placas de trânsito brasileiras.**

  Esse objetivo é tangível através das métricas de acurácia, precisão e *recall* que são geradas a partir dos conceitos da matriz de confusão. Então, esse objetivo também foi alcançado. O único adendo como melhoria é gerar uma visão mais analítica e crítica dos resultados obtidos por essas métricas. Um último ponto válido a ser ressaltado como melhoria é utilizar um software que automatize a criação da matriz de confusão. Dessa forma, mais imagens podem ser utilizadas na fase de teste, aumentando significativamente a confiabilidade dos dados.

- **Executar um conjunto de simulações em vídeo para identificar a capacidade de detecção e classificação das placas de trânsito brasileiras em diferentes contextos.**

  Através dos cenários construídos na Seção 4.6 deste trabalho, o objetivo proposto foi alcançado. As situações realizadas para esses cenários foram as seguintes: *às 12 horas do dia, ao pôr do sol às 18 horas e 30 minutos, à noite às 20 horas e por fim em um dia de chuva às 16 horas* . Cada situação proposta mostrou características específicas, nas quais o modelo de treinamento de placa de trânsito teve que ser exposto.

  No geral, os melhores resultados ocorrem por conta da iluminação, principalmente a luz do sol, e a uma boa visibilidade. Podemos observar que o cenário cuja as métricas de acurácia e precisão foram mais altas e constantes foi na situação gravada às 12 horas do dia, pois é nesse momento que a posição do sol está propícia para iluminar o ambiente de forma homogênea.

  Para além destas questões, ainda há algumas restrições nesse trabalho que impactam diretamente com o resultado, como por exemplo, uma câmera de qualidade profissional, onde retrata uma imagem de gravação mais nítida e fiel, e ainda que possa gravar com uma boa performance mesmo em locais escuros ou com baixa iluminação. Esses equipamentos fazem diferença na detecção das placas de trânsito. Até mesmo no cenário de *um dia de chuva às 16h*, que foi o mais prejudicado em relação às métricas, isso por conta da má visibilidade que a chuva proporciona, com um equipamento adequado é possível ter imagens de melhor nitidez fazendo com que o modelo de treinamento seja capaz de detectar de forma mais eficiente.

  Ainda assim, nestes cenários, foram apresentadas diversas dificuldades. Entre elas, existem situações adversas que independem do modelo de treinamento de placas de trânsito proposto por esse trabalho. Por exemplo, a obstrução de placa de trânsito por outros veículos, ou objetos no ambiente que acabam sendo posicionados na frente da placa de trânsito, como árvores e plantas. Além de placas danificadas ou pela ação do homem ou da natureza. Outra situação é o mal posicionamento de placas de trânsito, ou a ausência delas. Todas essas situações impossibilitam que o modelo detecte placas de trânsito, e isso em uma situação real, envolvendo veículos autônomos, pode ser passível a acidentes.

- **Verificar a distância máxima que o modelo de treinamento detecta, identifica e classifica as placas de trânsito brasileiras.**

  Este objetivo, em específico, não foi alcançado de forma completa. Visto que o cenário proposto foi realizado em um espaço amostra do trajeto estabelecido para os cenários na Seção 4.6. Essa parcela não define todo o resultado que o modelo de treinamento de placas de trânsito pode oferecer.

Dito isso, o objetivo real foi compreender a importância de ter um modelo de treinamento que consiga detectar placas de trânsito a longas distâncias para que, assim, o veículo tenha tempo suficiente de realizar uma ação de acordo com a placa de trânsito identificada.

Dessa forma, o ideal é que fosse realizada uma série de testes, para que, assim, possamos validar de fato a distância, a distância máxima de detecção que este modelo de treinamento pode alcançar. Além disso, existem diversas variáveis que podem interferir nessa detecção, como a iluminação, o clima, o ambiente.

Ainda assim, a forma que foi realizado o teste neste pequeno espaço amostra é válida e pode ser replicada para outros espaços amostrais, e gerar um procedente comparativo entre esses espaços amostrais.

Sobre o objetivo geral deste trabalho **documentar um processo de rotulagem e treinamento utilizando o sistema de identificação de objetos, YOLOv3, para realizar o reconhecimento de placas de trânsito brasileiras**, é factível enunciar que o objetivo central foi realizado. Ele foi alcançado com embasamento nos fundamentos técnicos e processuais mostrados nos resultados do presente trabalho, tão bem como nos resultados satisfatórios dos objetivos específicos.

Em suma, percebe-se que o Brasil tem um longo caminho ainda para realizar a implementação de veículos autônomos no cotidiano das pessoas. Visto que existem muitas dificuldades impostas no ambiente em que há tráfego de automóveis. Esse trabalho foi apenas mais um passo, para que possa contribuir com a área de atuação de veículos autônomos e tornar essa tecnologia uma realidade para os brasileiros.

5.1 Oportunidade para trabalhos futuros

Ainda existem várias formas de evoluir e enriquecer ainda mais o trabalho, um desses pontos é o refinamento do modelo de treinamento para reconhecimento de placas de trânsito. Esse modelo engloba 12 tipos de placas de trânsito diferentes. Ainda existem várias categorias. Dessa forma, há uma oportunidade de melhoria, e fazer com que o modelo seja capaz de identificar cada vez mais tipos de placas de trânsito.

Um segundo ponto de melhoria está relacionado ao anterior, o YOLOv3 possui uma certa “limitação”, por exemplo, quanto mais classes ele é capaz de identificar, mais lento ele fica. Dessa forma, é interessante buscar uma forma de otimizar o algoritmo do YOLOv3, ou melhorar a forma que é feita a categorização das classes, por exemplo, ao invés de fazer uma classe por tipo de placa de trânsito, fazer com que uma classe seja um conjunto de placas de trânsito. Ainda seguindo com um exemplo, seria fazer com que o

YOLOv3 identificasse placas de Advertência ou Regulamentação, e com um outro sistema realizar a identificação do conteúdo da placa identificada pelo YOLOv3.

Esse processo citado é interessante para esse trabalho, pois as placas de *Limite de velocidade* podem ser consideradas como um conjunto de placas, visto que independente da placa de *Limite de velocidade* que o modelo encontrar ela será identificada como tal, e não como *Limite de velocidade de 90km/h*, por exemplo. Então, nesse sentido, um sistema, ou algoritmo que extraia o conteúdo interno da placa após uma detecção prévia seria bastante útil.

Um outro viés que pode vir a ser abordado em trabalhos futuros é realizar todos os mesmos cenários, porém com equipamentos profissionais ou configurações diferentes. Por exemplo, no cenário da chuva, poderia utilizar no para-brisa do carro um produto que inibe as gotas de chuva no vidro, isso provavelmente faria com que o acurácia nesse modelo aumentasse. Ainda, melhorar o hardware aplicado neste trabalho, como a adição de sensores, câmera adequada, equipamentos de medição mais precisos, tudo isso impacta diretamente no desempenho do modelo de detecção.

Outra forma de melhoria seria a realização de um experimento, que submeteria vários espaços amostrais, para analisar a distância na qual o modelo de treinamento consegue detectar e identificar uma placa de trânsito, como apresentado na Seção 4.7, a importância de ter um modelo que consiga detectar a longas distâncias, para que assim o veículo tenha um tempo de reação satisfatório e que não sofra nenhuma risco de acidente.

E por fim, uma possível melhoria é a aplicação de filtros nos vídeos gravados para os cenários, visto na Seção 4.6, a intenção é aplicar filtros no processamento das imagens, como contraste, saturação, brilho, escala cinza, entre outros. O objetivo é identificar se a aplicação de filtros em diferente situações melhoraria a métrica de acurácia e precisão do modelo.

Referências

BRASIL, C. d. T. B. *LEI Nº 9.503, DE 23 DE SETEMBRO DE 1997*. [S.l.], 1997. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/l9503.htm>. Citado na página 32.

CONTRAN, C. N. de T. Manual Brasileiro de Sinalização de Trânsito, Volume 1 – Sinalização Vertical de Regulamentação. Brasil, 2007. Citado na página 32.

CONTRAN, C. N. de T. Manual Brasileiro de Sinalização de Trânsito, Volume 2 – Sinalização Vertical de Advertência. Brasil, 2007. Citado na página 32.

CONTRAN, C. N. de T. RESOLUÇÃO Nº 600, DE 24 DE MAIO DE 2016. [S.l.], 2016. Disponível em: <https://www.in.gov.br/web/dou/-/resolucao-n-600-de-24-de-maio-de-2016-22921310>. Citado na página 82.

FAWCETT, T. *An introduction to ROC analysis*. Staunton Court, Palo Alto, CA 94306, USA, 2005. Citado na página 29.

FONSECA, J. J. S. *Metodologia da pesquisa científica*. São Carlos: Serviço de Biblioteca e Informação, 2002. 1-127 p. Disponível em: <http://www.ia.ufrrj.br/ppgea/conteudo/conteudo-2012-1/1SF/Sandra/apostilaMetodologia.pdf>. Citado 2 vezes nas páginas 35 e 42.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de pesquisa*. 1. ed. Universidade Federal do Rio Grande do Sul, Porto Alegre: Editora da UFRGS, 2009. ISBN 978-85-386-0071-8. Citado na página 35.

GIL, A. C. *Como Elaborar Projetos de Pesquisa*. 4. ed. São Paulo: EDITORA ATLAS, 2002. ISBN 85-224-3169-8. Citado na página 36.

GRIFFIN, G.; HOLUB, A.; PERONA, P. *Caltech-256 Object Category Dataset*. [S.l.], 2007. Disponível em: <https://authors.library.caltech.edu/7694/1/CNS-TR-2007-001.pdf>. Citado na página 54.

GUIMARãES, J. H. d. N. e. O. *Método para Manutenção de sistema de software utilizando técnicas arquiteturais*. São Paulo, 2008. 101 p. Disponível em: <https://www.teses.usp.br/teses/disponiveis/3/3141/tde-29012009-134316/publico/Dissertacao_JulioHNOG_200810202304_Revisoes.pdf>. Citado na página 49.

KNIBERG, H. *Kanban and Scrum - Making the Most of Both*. [S.l.]: InfoQ, 2009. 120 p. ISBN 978-0-557-13832-6. Citado na página 37.

KUN, Z. et al. *Traffic signs detection and recognition under low-illumination conditions*. China, 2020. Citado na página 66.

LI, E. Y. *Dive Really Deep into YOLO v3: A Beginner's Guide*. 2019. <https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e>. Citado 2 vezes nas páginas 25 e 26.

LUYAO, D. et al. *Improved detection method for traffic signs in real scenes applied in intelligent and connected vehicles*. China, 2020. Citado na página 31.

MANTRIPRAGADA, M. *Digging deep into YOLO V3 - A hands-on guide Part 1*. 2020. <https://towardsdatascience.com/digging-deep-into-yolo-v3-a-hands-on-guide-part-1-78681f2c7e29>. Citado 2 vezes nas páginas 25 e 27.

MORESI, E. *Metodologia da Pesquisa*. Universidade Católica de Brasília - DF, 2003. Citado na página 36.

OLIVEIRA, M. F. *METODOLOGIA CIENTÍFICA: um manual para a realização de pesquisas em administração*. Catalão - GO, 2011. Citado 2 vezes nas páginas 35 e 36.

OSóRIO, F.; HEINEN, F.; FORTES, L. *Controle Inteligente de Veículos Autônomos: Automatização do Processo de Estacionamento de Carros*. São Leopoldo - RS, 2001. Citado na página 23.

PRINA, B. Z.; TRENTIN, R. Gmc: Geração de matriz de confusão a partir de uma classificação digital de imagem do arcgis. SBSR, Santa Maria - RS, Brasil, 2015. Citado na página 29.

PRODANOV, C. C.; FREITAS, E. C. d. *Metodologia do Trabalho científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico*. 2. ed. Novo Humbugo, Rio Grande do Sul, Brasil: Editora Feevale, 2013. ISBN 978-85-7717-158-3. Citado 3 vezes nas páginas 35, 36 e 37.

REDMON, J. *Darknet: Open Source Neural Networks in C*. 2013–2016. <http://pjreddie.com/darknet/>. Citado na página 26.

REDMON, J. et al. *You Only Look Once: Unified, Real-Time Object Detection*. Las Vegas, 2016. Citado 4 vezes nas páginas 25, 26, 28 e 49.

REDMON, J.; FARHADI, A. *YOLO9000: Better, Faster, Stronger*. [S.l.], 2017. Citado na página 25.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv*, 2018. Citado 4 vezes nas páginas 27, 28, 49 e 60.

REZENDE, J. L. d. *Aplicando Técnicas de Conversação para a Facilitação de Debates no Ambiente AulaNet*. Rio de Janeiro, 2003. Disponível em: <http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0115649_03_cap_08.pdf>. Citado na página 37.

SANTOS, L. C. B. D. *IMPLANTAÇÃO DE VEÍCULOS AUTÔNOMOS NO CONTEXTO BRASILEIRO: AVALIAÇÃO DOS FATORES QUE INFLUENCIAM NO INTERESSE DE USO COM EQUAÇÕES ESTRUTURAIS*. Brasília, 2017. Citado na página 23.

THIPSANTHIA, P.; CHAMCHONG, R.; SONGRAM, P. *Road Sign Detection and Recognition of Thai Traffic Based on YOLOv3*. Tailândia, 2019. Citado na página 31.

TING, K. M. *Confusion Matrix. Encyclopedia of Machine Learning*. Springer, Boston, MA, 2011. Citado na página 29.

TZUTALIN. *LabelImg*. [S.l.], 2005. Disponível em: <https://github.com/tzutalin/labelImg>. Citado na página 55.

VITTORIO, A. *Toolkit to download and visualize single or multiple classes from the huge Open Images v4 dataset*. [S.l.]: Github, 2018. <https://github.com/EscVM/OIDv4_ToolKit>. Citado na página 59.

ZHANG, G. W. et al. *Detecting Small Chinese Traffic Signs via Improved YOLOv3 Method*. China, 2020. Citado na página 80.

Apêndices

APÊNDICE A – Guia para execução de um Treinamento de um Dataset no Google Colab

O processo destacado na Figura 16 apresentado na Secção 4.4 *Executar Treinamento* será detalhado neste apêndice A. As etapas que serão apresentadas têm como objetivo mostrar como realizar um processo de treinamento de Dataset utilizando o Google Colab na prática.

1. **Instalar e configurar o *Darknet*:**

   O primeiro passo, como visto na Seção 4.1, é instalar o *darknet*. Para isso, basta clonar o repositório, executando o comando:

   ```
   $ !git clone https://github.com/AlexeyAB/darknet
   ```

   Nota-se que os comandos de *bash* podem ser executados no Colab prefixando o comando com "!". Após clonar o repositório para dentro do ambiente de desenvolvimento do Colab, basta fazer uma alteração nas configurações do arquivo "Makefile". Essa alteração consiste em habilitar a GPU ao modelo. Desse modo, a rede neural do *darknet* é otimizada e realiza uma melhor performance. Essa mudança pode ser feita a partir da instrução:

   ```
   $ %cd darknet
   $ !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
   $ !sed -i 's/GPU=0/GPU=1/' Makefile
   $ !sed -i 's/CUDNN=0/CUDNN=1/' Makefile
   ```

   Depois de ter reconfigurado o arquivo do "Makefile", basta compilá-lo usando o:

   ```
   $ !make
   ```

2. **Executar a função *imShow*:**

   Nessa etapa, executa-se uma função *imShow* que está escrita na linguagem de programação *python*. Essa instrução é responsável por mostrar ao pesquisador o resultado de uma imagem predita. Então, ele basicamente irá plotar uma imagem com as *bounding boxes*. Essa função será utilizada em uma próxima etapa. A *imShow* pode ser observada a seguir:

   ```
   def imShow(path):
     import cv2
     import matplotlib.pyplot as plt
     %matplotlib inline
   ```

```
image = cv2.imread(path)
height, width = image.shape[:2]
resized_image = cv2.resize(image,(3*width, 3*height),
                           interpolation = cv2.INTER_CUBIC)
fig = plt.gcf()
fig.set_size_inches(18, 10)
plt.axis("off")
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
plt.show()
```

3. **Conectar o Colab com o Google Drive:**

   Essa etapa é de extrema importância, visto que a máquina virtual cedida pelo Colab é desconectada após um tempo decorrido. Sendo assim, todas as informações contidas no ambiente virtual são perdidas. A solução para que isso não ocorra é utilizar o google drive[1], que irá manter todos os dados a salvo e em nuvem para serem usados a qualquer momento.

   Dessa forma, basta passar as informações do drive para a máquina de desenvolvimento do Colab. Para haver a conexão entre os dois serviços, é preciso que o pesquisador permita tal acesso. Para isso, basta executar a instrução:

   ```
   $ %cd ..
   $ from google.colab import drive
   $ drive.mount('/content/gdrive')
   ```

   Em vista de facilitar o desenvolvimento, o caminho que referencia a pasta do *Drive* dentro do Colab pode ser alterado por um *link* simbólico. Sendo assim, a pasta pode ser referenciada usando apenas "/mydrive". Para que isso ocorra, execute o comando:

   ```
   $ !ln -s /content/gdrive/My\ Drive/ /mydrive
   $ !ls /mydrive
   ```

4. **Mover o *dataset* para o ambiente virtual:**

   É nesse momento em que os passos realizados nas Seções 4.2 e 4.3 são utilizados, de fato, dentro do treinamento. Essas duas seções geraram um *dataset* rotulado com as anotações do YOLO. Dessa forma, é necessário que a pasta com as imagens geradas pelas seções anteriores esteja no *drive*, tornando as imagens acessíveis para o treinamento. Neste trabalho, essa ação procedeu-se da seguinte forma:

   - Pasta do *dataset* no seu computador deve estar nomeada com a palavra "obj' de preferência.

[1] É um serviço de armazenamento e sincronização de arquivos em nuvem.

- Depois basta zipar[2] a pasta do *dataset*.
- Crie uma pasta na raíz do Google drive com o nome de "yolov3", de preferência.
- Por fim, basta carregar o arquivo "obj.zip" para a pasta do drive.

Tendo realizado esses passos, basta transferir o "obj.zip" para a máquina virtual do Colab. Essa etapa pode ser feita a partir do comando:

```
$ !cp /mydrive/yolov3/obj.zip ../
```

Finalmente, é necessário que descompacte o arquivo "obj.zip" na máquina virtual dentro da pasta "/data" do *darknet*. Para isso, utiliza-se:

```
$ !unzip ../obj.zip -d data/
```

5. **Configuração de arquivos para treinamento:**

   Esta etapa envolve a configuração adequada do arquivo "yolov3_custom.cfg", assim como o "obj.data", "obj.names" e o arquivo de "generate_train.txt". De início, o primeiro a ser configurado é o "yolov3_custom.cfg", ele é um dos arquivos mais importantes, pois é nele onde está a maior parte das configurações dos treinamentos, como, por exemplo, as camadas convolucionais.

   No momento em que é feita a instalação do *darknet*, o arquivo em questão já vem pré-configurado e chamado de "yolov3.cfg", mas para o *dataset* de placas de trânsito brasileiras algumas mudanças são necessárias. O primeiro passo é selecionar o arquivo "yolov3.cfg", copiado para a pasta no Google drive, e mudar o nome para "yolov3_custom.cfg", já que serão feitas algumas configurações nele. Para isso, execute:

   ```
   $ !cp cfg/yolov3.cfg /mydrive/yolov3/yolov3_custom.
     cfg
   ```

   Agora que está no drive, é possível acessar o arquivo "yolov3_custom.cfg" para realizar as configurações. Dentro do arquivo, a primeira alteração é feita no início, onde ele será colocado em modo de Treino. Para isso, basta habilitar a opção. Seguindo, nas variáveis de "max_batches" e "steps", neste trabalho, foram colocadas respectivamente "max_batches = 24000" e "steps = 19200, 21600". Essas variáveis definem quantas iterações o treinamento irá realizar.

   Por fim, nas três camadas de YOLO do arquivo, que são identificadas por "[yolo]", basta que na variável "classes" seja colocado o número de categorias. No caso deste presente TCC, fica assim, "classes=12". Com as configurações do arquivo "yolov3_custom.cfg" concluídas, é necessário transferi-lo para a máquina virtual do Colab. Sendo assim, execute o comando:

[2] No universo computacional, é o ato de compactar para, assim, reduzir o tamanho de um arquivo.

```
$ !cp /mydrive/yolov3/yolov3_custom.cfg ./cfg
```

Os próximos arquivos a serem configurados são os “obj.names” e “obj.data”. O “obj.names” deve conter todas as classes do *dataset*. Dessa forma, ele será exatamente igual ao arquivo de texto “classes.txt” gerado na seção 4.3. O resultado “obj.names” para esse trabalho pode ser visto na Figura 40.

```
wrong_way_sign
no_pedestrian_traffic_sign
double_direction_of_circulation_sign
pedestrian_crossing_sign
bump_or_spine_sign
stop_sign
obligatory_passage_sign
no_parking_sign
forbidden_to_turn_right_sign
track_circulation_direction_sign
pedestrian_traffic_sign
speed_limit_sign
```

Figura 40 – Conteúdo do arquivo de configuração “obj.names”.

Fonte: Autor.

O “obj.data” deverá conter as informações apresentadas na Figura 41, sendo necessário o número de classes do *dataset*. Nota-se que precisa inserir a localização do *backup*. Esse caminho de backup é onde serão salvos os pesos do modelo durante o treinamento. Crie uma pasta de backup no Google Drive e coloque o caminho correto neste arquivo.

```
classes = 12
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = /mydrive/yolov3/backup/
```

Figura 41 – Conteúdo do arquivo de configuração “obj.data”.

Fonte: Autor

Após ter gerado os dois arquivos apresentados, é preciso que eles sejam transferidos do drive para a máquina virtual do Colab, execute:

```
$ !cp /mydrive/yolov3/obj.names ./data
$ !cp /mydrive/yolov3/obj.data  ./data
```

O último arquivo de configuração necessário antes de começar a treinar o detector é o arquivo “generate_train.txt”, que detém os caminhos relativos das imagens do *dataset* que será utilizado no treinamento. O *script* responsável para que isso ocorra pode ser visto a seguir:

```
import os

image_files = []
os.chdir(os.path.join("data", "obj"))
for filename in os.listdir(os.getcwd()):
    if filename.endswith(".jpg"):
        image_files.append("data/obj/" + filename)
os.chdir("..")
with open("train.txt", "w") as outfile:
    for image in image_files:
        outfile.write(image)
        outfile.write("\n")
    outfile.close()
os.chdir("..")
```

Por fim, é necessário transferir o arquivo "generate_train.txt" do drive para o Colab. E executar o arquivo que foi escrito na linguagem de programação do *python*. Para que essas ações ocorram, basta executar, respectivamente, os comandos:

```
$ !cp /mydrive/yolov3/generate_train.py ./
$ !python generate_train.py
```

Com isso, as configurações estão concluídas. Por motivos didáticos, a Figura 42 mostra todos os arquivos que a pasta do Google Drive deve conter para que esse treinamento seja realizado.

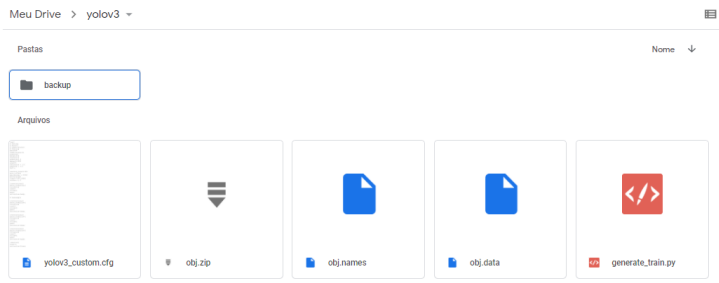


Figura 42 – Arquivos necessários para executar um treinamento.

Fonte: Autor

6. ***Download* das camadas convolucionais:**

Para o treinamento, são utilizados pesos convolucionais que são pré-treinados no Imagenet. Nesse caso, são usados pesos do modelo *darknet53*. Eles irão ajudar o detector de objetos a ser mais preciso, sem ter um tempo de treinamento maior. Para fazer o *download* dos pesos para as camadas convolucionais, execute:

```
$ !wget http://pjreddie.com/media/files/KKKK.conv.74
```

7. **Executar a instrução *ClickConnect*:**

   Os notebooks do Colab possuem um tempo de inatividade. Caso o usuário fique inativo entre 30-90 minutos, o serviço é desconectado, e quando isso ocorre, tudo o que foi executado é perdido. Ainda é importante ressaltar que o treinamento do detector leva várias horas e até mesmo dias, e é necessário que o notebook, em conjunto com a máquina virtual do Colab, continue funcionando sem se desconectar.

   Uma solução pensada para isso é utilizar o *script*:

```
function ClickConnect(){
console.log("Working");
document.querySelector("colab-toolbar-button#connect").click()
}
setInterval(ClickConnect,60000)
```

   Esse *script* simula um clique na página do notebook a cada 10 minutos. Dessa forma, o Colab entende que o usuário ainda está ativo na página, mesmo não estando. Para que esse *script*, escrito na linguagem de programação *javascript*, funcione, basta abrir o "inspecionar" da página do notebook segurando as teclas (CTRL + SHIFT + i).

   Depois, copie o *script* na opção de "Console", execute, e já estará funcionando. Ainda assim, é importante salientar que o Colab obrigatoriamente desconecta a máquina virtual após 12h para uma conta de usuário gratuita, e 24h para usuários com contas pagas.

8. **Treinar o Detector:**

   Com os preparativos prontos, finalmente é possível executar o treinamento de fato, bastando executar o comando:

```
$ !./darknet detector train data/obj.data cfg/
   yolov3_custom.cfg darknet53.conv.74 -dont_show
```

   O comando gera, então, um treinamento que dura muito tempo, dependendo da quantidade de imagens no *dataset*, e também da quantidade de classes. O treinamento do *dataset* de placas de trânsito brasileiras executado por este TCC teve uma duração média de 60 horas. Um exemplo da saída que o treinamento produz pode ser visto na Figura 43.

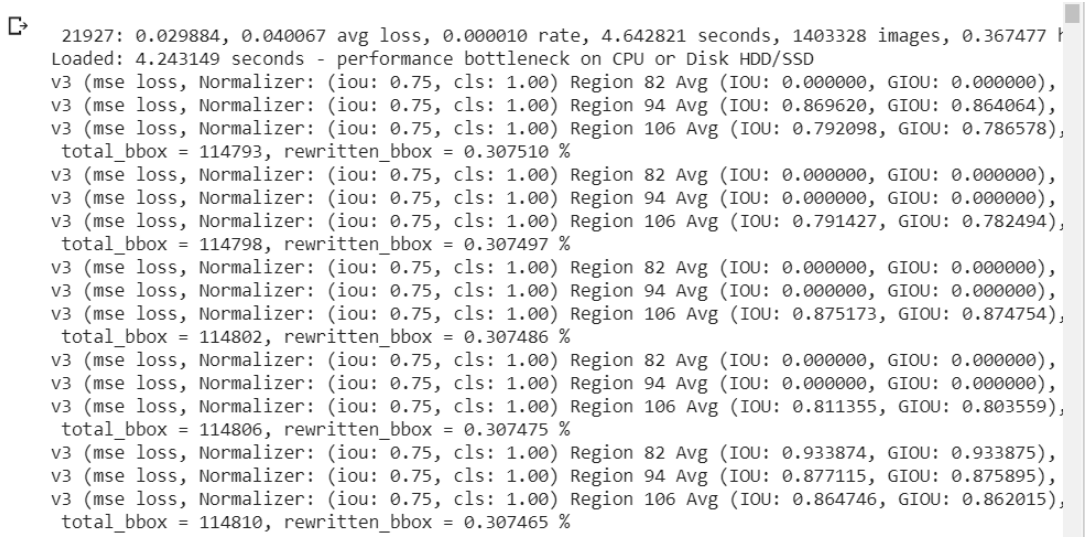


Figura 43 – Exemplo de saída gerada pelo treinamento.

Fonte: Autor.

Um ponto crucial a ser dito é que caso aconteça algum problema durante o treinamento, por exemplo, o Colab parar de responder ou por algum motivo for necessário parar o treinamento no meio da sua execução, não precisa se preocupar. Pois, a cada 100 iterações realizadas pelo treinamento, é feito um salvamento parcial do "yolov3_custom.weight". Esse peso é salvo na pasta *backup* do seu drive. Dessa forma, todo progresso de treino não é perdido. Caso queira continuar de onde parou, basta executar:

```
$ !./darknet detector train data/obj.data cfg/
   yolov3_custom.cfg /mydrive/yolov3/backup/
   yolov3_custom_last.weights -dont_show
```

Finalmente, quando o treinamento terminar, o "yolov3_custom.weight" estará pronto para realizar as predições.

9. **Alterar a configuração de treinamento para teste:**

   Para que o peso que foi treinado possa finalmente fazer a detecção e classificação, é necessário configurar o arquivo "yolov3_custom.cfg" para o modo de testes. Execute o comando para que isso ocorra:

```
$ %cd cfg
$ !sed -i 's/batch=64/batch=1/' yolov3_custom.cfg
$ !sed -i 's/subdivisions=16/subdivisions=1/'
   yolov3_custom.cfg
$ %cd ..
```

10. **Realizar uma predição de teste:**

A última etapa é realizar uma predição usando o modelo treinado. É possível, assim, fazer a detecção e classificação de placas de trânsito brasileiras. Para realizar uma predição de qualquer imagem, deve-se criar uma pasta na raiz do seu drive com o nome *images*, por exemplo. Essa pasta deve conter as imagens que desejar para fazer as predições. Após isso, basta executar o seguinte comando para realizar a predição:

```
$ !./darknet detector test data/obj.data cfg/
   yolov3_custom.cfg /mydrive/yolov3/backup/
   yolov3_custom_last.weights /mydrive/images/placas-
   transito.jpg -thresh 0.3
$ imShow('predictions.jpg')
```

Nota-se que a função "imShow" está presente na instrução apresentada. Ela que é responsável por mostrar o resultado da predição no notebook do Colab.

Depois de realizar todo esse processo, finalmente é possível fazer predições de placas de trânsito brasileiras. Repare que o processo de treinamento é, de certa forma, genérico, e pode ser replicado por diversos outros contextos, não só na identificação de placas de trânsito brasileiras. O processo das etapas que foram descritas pode ser visto em sua amplitude no notebook do Colab desenvolvido pelo pesquisador. Para visualizar todos os passos de execução de um treinamento, basta acessar o link (<https://colab.research.google.com/drive/1WB2EDto-CgksEdUFdUHuVLGcw4dCzLtw?usp=sharing>).