Jean Pablo Vieira de Mello

# Detecção profunda de semáforo por sobreposição de contexto sintético em imagens naturais arbitrárias

Vitória, ES

2021

Jean Pablo Vieira de Mello

# Detecção profunda de semáforo por sobreposição de contexto sintético em imagens naturais arbitrárias

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Prof. Dr. Thiago Oliveira dos Santos

Vitória, ES

2021

Jean Pablo Vieira de Mello

Detecção profunda de semáforo por sobreposição de contexto sintético em imagens naturais arbitrárias/ Jean Pablo Vieira de Mello. – Vitória, ES, 2021-
92 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Thiago Oliveira dos Santos

Dissertação de Mestrado – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Programa de Pós-Graduação em Informática, 2021.

1. Semáforo. 2. Contexto sintético. 3. Detecção profunda. 4. Computação Gráfica. 5. Imagens naturais. I. Oliveira-Santos, Thiago. II. Universidade Federal do Espírito Santo. IV. Detecção profunda de semáforo por sobreposição de contexto sintético em imagens naturais arbitrárias

Jean Pablo Vieira de Mello

# Detecção profunda de semáforo por sobreposição de contexto sintético em imagens naturais arbitrárias

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Trabalho aprovado. Vitória, ES, _____ de _____ de 2021:

---

**Prof. Dr. Thiago Oliveira dos Santos**
Orientador

---

**Prof. Dr. Alberto Ferreira de Souza**
Convidado 1

---

**Prof. Dr. Jefersson Alex dos Santos**
Convidado 2

Vitória, ES
2021

*Dedico este trabalho às pessoas mais próximas de mim e a todos que me incentivaram a resistir e avançar, incluindo meu orientador Thiago, meus colegas de laboratório, meus amigos, meus pais Jorge e Marlete, minha irmã Patrícia e, em especial, à minha namorada Gisselle que acompanhou de perto minha trajetória tanto nos momentos mais difíceis quanto nos mais gloriosos.*

# Agradecimentos

Agradeço à Universidade Federal do Espírito Santo (UFES) e ao Laboratório de Computação de Alto Desempenho (LCAD) pela oportunidade de cursar o Mestrado em Informática e desenvolver este estudo. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior — Brasil (CAPES) — Código de Financiamento 001, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES) — concessão 84412844. Agradeço também à NVIDIA Corporation pelas unidades de processamento gráfico (GPU's) doadas ao laboratório e utilizadas neste trabalho. Por fim, sou grato a Deus por permitir que tudo isso fosse possível e a todos que me orientaram e apoiaram moral e financeiramente, incluindo orientador, colegas de laboratório — em especial os que participaram diretamente no desenvolvimento da publicação que inspirou esta dissertação —, amigos, família e namorada.

*"Na verdade 'nada' é uma palavra esperando tradução."*
*(Humberto Gessinger)*

# Publicações e direitos

Esta dissertação é baseada no artigo *Deep traffic light detection by overlaying synthetic context on arbitrary natural images*, apresentado à 33ª *Conference on Graphics, Patterns and Images* (SIBGRAPI 2020) e publicado pela revista *Elsevier Computers & Graphics* (CAG) — ISSN 0097-8493, do qual fui primeiro autor. Por simplificação, o artigo não é repetidamente citado como fonte no trabalho, mas através desta nota declaro que a maior parte do conteúdo deste trabalho, incluindo imagens, é adaptada do artigo, que está reconhecidamente sob direitos autorais da Elsevier.

Informações sobre a publicação:

- Autores: Jean Pablo Vieira de Mello, Lucas Tabelini, Rodrigo F. Berriel, Thiago M. Paixão, Alberto F. de Souza, Claudine Badue, Nicu Sebe e Thiago Oliveira-Santos;
- DOI: <https://doi.org/10.1016/j.cag.2020.09.012>;
- arXiv: <https://arxiv.org/pdf/2011.03841>.

Durante o período de Mestrado em que este trabalho foi desenvolvido, outro trabalho do qual fui primeiro autor — fora do escopo desta dissertação — foi publicado. O artigo *Deep Learning-based Type Identification of Volumetric MRI Sequences* foi apresentado e publicado pela 25ª *International Conference on Pattern Recognition* (ICPR 2020), propondo um algoritmo de classificação de sequências de imagens de ressonância magnética volumétricas.

Informações sobre a publicação:

- Autores: Jean Pablo Vieira de Mello, Thiago M. Paixão, Rodrigo Berriel, Mauricio Reyes, Claudine Badue, Claudine Badue, Alberto De Souza e Thiago Olveira-Santos;
- DOI: <https://doi.org/10.1109/ICPR48806.2021.9413120>.

# Resumo

O uso de Redes Neurais Profundas como solução para problemas relacionados à direção autônoma tem sido cada vez mais considerado pelos pesquisadores. Com esse ferramental, pode-se detectar elementos comuns do trânsito, como pedestres, placas e semáforos com eficácia, bastando fornecer como dados de entrada uma quantidade representativa de imagens que descrevam um contexto real de trânsito. Em particular, a detecção de semáforos e a classificação correta de seu estado são essenciais na prevenção de acidentes. No entanto, coletar e anotar tal conjunto de dados de semáforo pode ser uma tarefa altamente custosa, tanto em tempo quanto em esforços. Para contornar esse problema, propõe-se a montagem de um expressivo conjunto de dados que se utiliza de contextos de trânsito gerados sinteticamente, através de computação gráfica simples, sobrepostos à imagens arbitrárias que apresentem cenas naturais não relacionadas à trânsito, dispensando a necessidade de coleta de dados do mundo real, automatizando a anotação dos semáforos dispostos na cena gerada e, ainda, possibilitando balancear as ocorrências do estado amarelo, que seriam difíceis de capturar, com as dos demais estados. Experimentos revelaram que a utilização do método produz resultados comparáveis aos obtidos utilizando-se dados do mundo real, superando-os em cerca de 4 pontos percentuais de mAP e *F1-score* em média.

**Palavras-chaves**: semáforo. contexto sintético. detecção profunda. Computação Gráfica. imagens naturais.

# Abstract

The use of deep neural networks as a solution to problems related to autonomous driving has been increasingly considered by the researchers. With this tooling, common traffic elements, such as pedestrians, traffic signs and traffic lights can be detected effectively, by simply providing as input data a representative amount of images that describe a real traffic context. In particular, the detection of traffic lights and the correct classification of their state are essential in preventing accidents. However, collecting and annotating such set of traffic light data can be a highly costly task, both in time and effort. To overcome this problem, it is proposed assembling an expressive dataset that overlaps traffic contexts generated synthetically, through simple computer graphics, on arbitrary images containing natural scenes not related to traffic. This dispenses the need for collection of real-world data, automates the annotation of traffic lights arranged in the generated scene, and also makes it possible to balance the occurrences of the yellow state, which would be difficult to capture, with those of the other states. Experiments revealed that using the method yields results comparable to those obtained using real-world data, with average mAP and F1-score about 4 percent points higher.

**Keywords**: traffic light. synthetic context. deep detection. Computer Graphics. natural images.

# Lista de ilustrações

# Lista de tabelas

# Sumário

# 1 Introdução

Esta seção visa introduzir a problemática e proposta de solução abordadas pelo trabalho intitulado "Detecção profunda de semáforo por sobreposição de contexto sintético em imagens naturais arbitrárias". A Seção 1.1 apresenta o problema a ser solucionado; a Seção 1.2 apresenta a solução proposta; a Seção 1.3 destaca os objetivos gerais e específicos deste trabalho; e, por fim, a Seção 1.4 descreve a estruturação do trabalho, auxiliando a leitura das demais seções.

## 1.1 Problemática e motivação

A direção autônoma tem se mostrado um importante tema de pesquisa nos últimos anos (BADUE et al., 2020; BERRIEL et al., 2018; BERRIEL et al., 2017b; MUTZ et al., 2016; LYRIO et al., 2015; BERRIEL et al., 2017a), e para que um veículo autônomo possa transitar com o máximo possível de segurança e de acordo com a legislação de trânsito vigente, é necessário que ele seja capaz de reconhecer elementos comuns do trânsito, como pedestres, placas de trânsito e semáforos. Em particular, os semáforos precisam ser não apenas detectados como também ter seu estado atual (solicitando parada, atenção, prosseguimento, etc.) reconhecido, determinando a decisão correta a ser tomada pelo veículo a fim de evitar acidentes (IIHS, 1996-2021).

Dada a importância da tarefa de reconhecimento de semáforos, muitos trabalhos neste âmbito têm sido propostos, empregando uma variedade de algoritmos que compreendem desde as clássicas Máquinas de Vetor de Suporte (SVM, do inglês *Support Vector Machines*) (JANG et al., 2014) às modernas Redes Neurais Profundas (DNN, do inglês *Deep Neural Networks*) (JENSEN; NASROLLAHI; MOESLUND, 2017). Estas últimas têm se destacado na resolução desta e de outras tarefas similares (TORRES et al., 2019; OUYANG; WANG, 2013; SARCINELLI et al., 2019), visto que redes estado-da-arte como Faster R-CNN (REN et al., 2015) e YOLO (REDMON et al., 2016) apresentam bons resultados na detecção de objetos em geral. No entanto, o sucesso de uma DNN na tarefa para a qual foi designada normalmente depende de seu aprendizado sobre uma quantidade expressiva de dados rotulados, o que significa, no caso de um detector, um grande conjunto de imagens em que o objeto-alvo é exibido e corretamente identificado.

Atualmente, alguns conjuntos de imagens de trânsito de diferentes regiões do mundo com rotulações de semáforos estão disponíveis para acesso público (YU et al., 2018; CHARETTE, 2013; FREGIN et al., 2018). No entanto, o processo de aquisição e rotulação dessas imagens envolve executar um considerável percurso de trânsito registrando imagens e identificar cada um dos semáforos capturados em cada uma das centenas/milhares/milhões

de imagens obtidas, o que é um trabalho altamente custoso em termos de tempo e esforço e que, ainda, está sujeito a um desbalanceamento de classes que prejudica o processo de aprendizado da DNN, visto que, por exemplo, o estado amarelo ocorre muito menos frequentemente em padrões comuns de semáforo. Esses problemas motivam a proposta de um método de geração automática de imagens de trânsito artificiais já rotuladas para detecção e reconhecimento de semáforos.

## 1.2 Proposta

A fim de lidar com os problemas apresentados na Seção 1.1, este trabalho propõe uma adaptação, para o âmbito de semáforos, do método proposto por Torres et al. (2019) para geração de dados para detecção de placas de trânsito: sobrepor imagens de contexto arbitrário com modelos bidimensionais de placas, distribuídas sobre a imagem em posições aleatórias e sempre registradas, como uma forma automática de rotulação. No entanto, vale observar que semáforos normalmente se limitam a fontes de luz de estrutura muito simples, podendo ser facilmente confundidos com faróis de veículos e outras fontes de luz no trânsito. Diante disso, o conjunto de imagens proposto por este trabalho utiliza como primeiro plano não uma distribuição aleatória de semáforos sintéticos, mas toda uma cena de trânsito sintética em 3D, gerada por meio de Computação Gráfica simples. A hipótese a ser explorada é a de que o contexto da cena favorece a detecção do objeto-alvo.

O desempenho do detector proposto e de outros detectores para comparação foi avaliado sobre conjuntos de dados populares na literatura que exibem cenas reais de trânsito. Em média, o detector proposto produz em teste aproximadamente 50% de mAP (métrica melhor descrita posteriormente, que indica melhor desempenho quanto maior seu valor). Este valor é cerca de 4 p.p. mais alto que o obtido com um detector preparado com dados de trânsito reais e quase o dobro do obtido com um detector preparado com dados gerados de forma mais semelhante ao método de Torres et al. (2019): modelos 2D de semáforos espalhados sobre imagens arbitrárias. Além disso, o resultado de mAP médio obtido pelo detector baseado em dados do mundo real pode ser aumentado em mais de 11 p.p. quando esses dados são combinados com o conjunto de dados sintético proposto.

## 1.3 Objetivos

### 1.3.1 Objetivo geral

Este trabalho tem como principal objetivo desenvolver, no âmbito de detecção profunda de semáforos em imagem, um método de geração e rotulação de dados que não envolva coleta e rotulação exaustiva e desbalanceada de imagens de trânsito do mundo real.

### 1.3.2 Objetivos específicos

Pretende-se atingir o objetivo geral do trabalho por meio dos seguintes procedimentos:

- Gerar cenas sintéticas de trânsito sinalizado por semáforos por meio de Computação Gráfica;
- Gerar um conjunto vasto de imagens que combinem as cenas sintéticas à planos de fundo arbitrários;
- Utilizar o conjunto gerado para treinar uma DNN para aprender a detectar semáforos em cenas reais e classificar seu estado;
- Avaliar o desempenho comparado ou em conjunto com detectores treinados com imagens do mundo real exaustivamente coletadas e rotuladas;
- Avaliar o impacto no desempenho do método: (i) da presença de outros elementos de trânsito além dos próprios semáforos na imagem sintética, (ii) do nível de realismo dos semáforos sintéticos e (iii) do contexto representado nas imagens de plano de fundo.

## 1.4 Estrutura

Este trabalho está estruturado de modo que possa detalhar de forma separada, porém conectada: (i) os conceitos teóricos que fundamentam o trabalho, (ii) os princípios da metodologia proposta, (iii) a experimentação e formas de avaliação de desempenho aplicadas e (iv) os resultados obtidos. Cada um destes tópicos é apresentado em sua própria seção, e seus detalhes organizados em subseções.

A Seção 2: **Embasamento teórico** explana brevemente os conceitos científicos nos quais o trabalho se baseia. Em geral, a seção apresenta o conceito de Aprendizado Profundo e a definição e aplicações de uma Rede Neural Artificial, ferramenta utilizada para a construção do modelo proposto de detecção de objetos — conceito também explorado na seção —, mais especificamente voltado à detecção de semáforos. Além disso, também são apresentados princípios de Computação Gráfica e como ela pode ser utilizada para a geração de dados sintéticos como os utilizados nos experimentos propostos. Por fim, são apresentados alguns dos trabalhos de detecção de semáforos que podem ser encontrados na literatura recente.

Por sua vez, a Seção 3: **Detecção de semáforos utilizando contexto sintético** descreve como se desenvolve a metodologia proposta, incluindo a escolha dos planos de fundo, o processo de geração das imagens de primeiro plano e de cada elemento gráfico

que as compõe, o processo de transformação e combinação das imagens de plano de fundo e primeiro plano e, por fim, o procedimento para identificar as posições e estados dos semáforos nas imagens geradas, para que o detector possa ser treinado.

Na Seção 4: **Metodologia Experimental** o leitor encontra informação sobre os materiais, recursos e procedimentos adotados para a execução dos experimentos de avaliação do método proposto. A seção começa com a descrição dos conjuntos de dados (referidos ao longo do texto pelo termo em inglês "*datasets*") utilizados para treinar, validar e testar o detector. Em seguida, apresenta-se a configuração experimental adotada, que inclui a implementação e a definição de parâmetros utilizados para o detector. Adiante, a seção explica como os resultados dos experimentos são avaliados. Como esperado, cada um dos experimentos realizados também é descrito, bem como as análises que se pretende levantar através deles. A seção é concluída com uma breve apresentação dos recursos computacionais utilizados para a execução dos experimentos.

A Seção 5: **Resultados e discussão** é dividida de modo que cada uma de suas subseções descreva os resultados obtidos para cada um dos experimentos descritos na seção anterior. Os resultados são apresentados tanto de forma numérica quanto analítica, promovendo uma discussão dos fatores que os teriam produzido.

Finalmente, o trabalho se encerra com a Seção 6: **Conclusão**, que apresenta as considerações finais a serem feitas sobre a relevância e desempenho do método proposto, o Apêndice A: **Acesso ao projeto**, que instrui o leitor sobre como acessar o repositório de materiais do projeto e que materiais encontrar, e o Apêndice B: **Resultados numéricos**, que tabela os resultados obtidos na parte experimental.

# 2 Embasamento teórico

Embora seja simples para o ser humano, distinguir elementos conhecidos em uma imagem, como um gato, um cão ou um veículo, é uma tarefa nada trivial para uma máquina. Diante deste desafio, muito esforço foi investido na tentativa de prover à máquina uma habilidade de reconhecer imagens inspirada no sistema visual humano. O desenvolvimento de Redes Neurais Convolucionais permitiu o alcance de alto desempenho em tarefas como classificação, detecção e segmentação de objetos em imagens, problemas comuns da chamada **Visão Computacional** (STANFORD, n.d.b; STANFORD, 2017a; GÉRON, 2019).

A Seção 2.1 introduz o conceito de Visão Computacional; a Seção 2.2 apresenta conceitos sobre Redes Neurais, incluindo Redes Neurais Convolucionais, e Aprendizado Profundo; a Seção 2.3 explica como as Redes Neurais podem ser usadas na tarefa de detecção de objetos em imagens; a Seção 2.4 introduz conceitos sobre Computação Gráfica e sua utilização para geração de imagens sintéticas destinadas a auxiliar uma Rede Neural a aprender a detectar objetos; e, finalmente, a Seção 2.5 apresenta exemplos de literatura recente em escopos similares ao deste trabalho.

## 2.1 Visão Computacional

O conceito de Visão Computacional nasce do desejo do ser humano de extrair, computacionalmente, informações sobre o contexto e os elementos ilustrados por imagens. Embora já seja aplicada em tarefas importantes como reconhecimento de objetos e caracteres, prover ao computador a habilidade de "enxegar" e compreender o mundo não é uma tarefa simples (SZELISKI, 2010). Observe a Figura 1[1].

Para o ser humano, fica instantaneamente claro que a imagem retrata um cachorro preso por uma coleira em um campo arborizado. O computador, porém, interpreta essa imagem de forma bem menos intuitiva, como mostra a Figura 2 que, por simplicidade, foca em uma área da imagem de apenas $10 \times 10$ pixeis.

Em um sistema computacional, imagens são geralmente interpretadas como grandes matrizes de pixeis. No exemplo mostrado na Figura 2, as três matrizes representam três canais de cores — vermelha, verde e azul — que, combinados, formam a visualização da imagem final em várias cores e tons. Para isso, cada valor na matriz é interpretado como uma intensidade de cor, comumente variante entre 0 (mais escuro) e 255 (mais claro) (STANFORD, 2017c; STANFORD, n.d.b; GOOGLE, 2020b). Agora, como transfor-

[1] <https://unsplash.com/photos/PoV6mtKJl6s>

Figura 1 – Cachorro encoleirado em um campo arborizado (foto por Paulo Brandão em *Unsplash* (licença livre)).


Figura 2 – Representação em pixeis de uma pequena área da imagem (adaptação da foto de Paulo Brandão em *Unsplash* (licença livre))

mar um conjunto de números em algo que a máquina possa compreender de forma similar ao ser humano?

Em 1981, David Hubel e Torsten Wiesel receberam o Prêmio Nobel de Fisiologia e Medicina por descobertas realizadas nas décadas de 1950 (HUBEL, 1959; HUBEL; WIESEL, 1959) e 1960 (HUBEL; WIESEL, 1968) a respeito do funcionamento do córtex visual de gatos e macacos. De acordo com suas publicações, o reconhecimento de padrões complexos do ambiente deriva da combinação de padrões mais simples reconhecidos, como linhas e bordas (GÉRON, 2019; STANFORD, 2017c). Com base nisso, os primeiros trabalhos no âmbito da Visão Computacional se basearam na extração e compreensão de bordas das imagens e, ao longo tempo, novos métodos foram sendo propostos como a representação de formas complexas através de formas simples, a segmentação de objetos baseada em intensidade de pixel, o reconhecimento de imagens baseado em características e Aprendizado de Máquina (SZELISKI, 2010; STANFORD, 2017a). Em particular, Krizhevsky, Sutskever e Hinton (2012) revolucionaram o âmbito de classificação de imagens ao vencer em 2012 o Desafio ImageNet (RUSSAKOVSKY et al., 2015), de classificação de imagens entre uma quantidade massiva de classes, utilizando uma Rede Neural Convolucional e obtendo desempenho muito superior aos algoritmos vencedores dos anos anteriores e inspirando a vitória de Redes Neurais Convolucionais nos anos seguintes (STANFORD, 2017a).

O presente trabalho se utiliza de uma Rede Neural Convolucional para detecção de objetos. A próxima seção conceitualiza o funcionamento deste tipo de algoritmo. Porém, primeiramente, é importante conhecer os conceitos de Redes Neurais e Aprendizado Profundo.

## 2.2 Redes Neurais e Aprendizado Profundo

Imagine possuir um conjunto de dados como o *Iris* (FISHER, 1936), que apresenta as larguras e comprimentos de sépalas e pétalas de 150 flores de íris de três espécies — Iris-Setosa, Iris-Virginica e Iris-Versicolor — (GÉRON, 2019), e que seja necessário construir um classificador que determine, dada uma nova amostra de flor de íris, se ela pertence ou não à espécie Iris-Setosa. A Figura 3a relaciona duas características do conjunto de dados (largura e comprimento das pétalas) com sua classe correspondente (Iris-Setosa ou não-Iris-Setosa).

Notavelmente, é possível traçar uma reta que separe bem as duas classes, de modo que novas amostras de flores possam ser classificadas de acordo com a posição de suas características no gráfico: de um lado ou de outro da reta. A reta que separa as classes é descrita pela equação

$$y = \mathbf{w}^\intercal \mathbf{x} \tag{2.1}$$

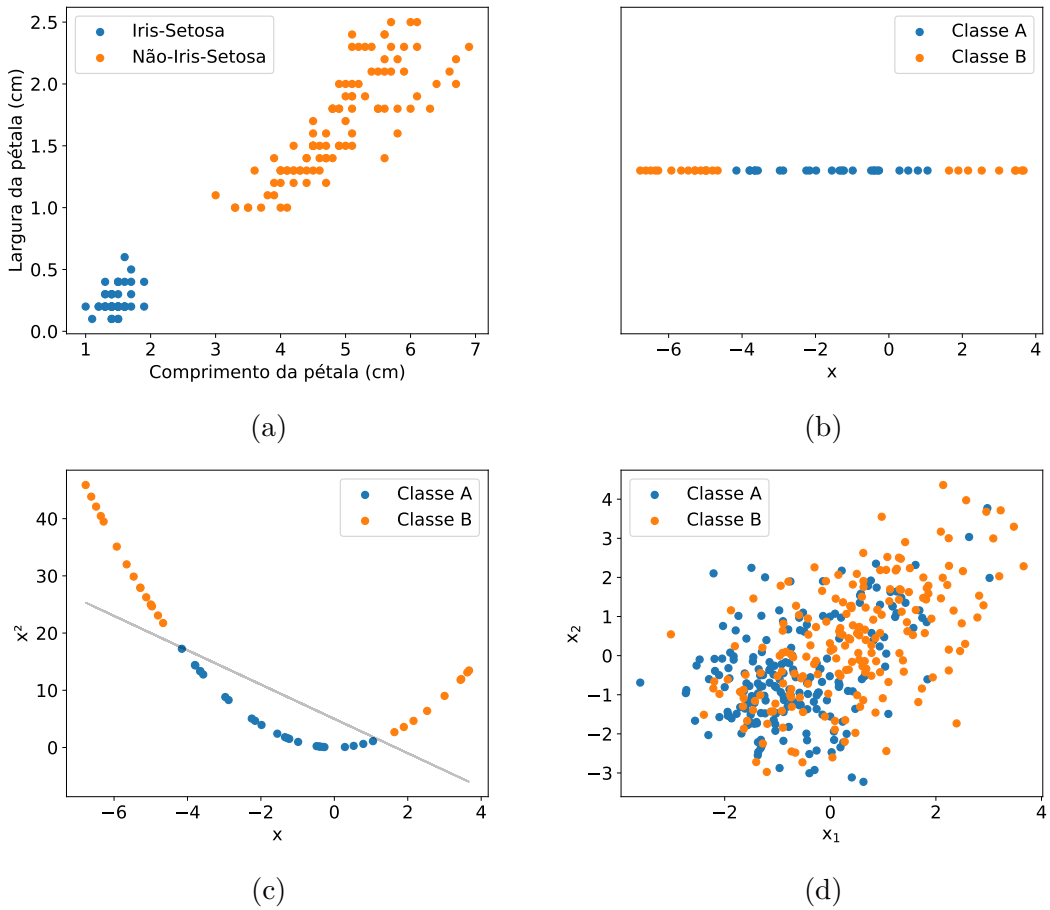



Figura 3 – Exemplos de (a) dados linearmente separáveis (*dataset Iris*); (b) dados não-linearmente separáveis; (c) adição de não-linearidade ao conjunto de dados em *b*; (d) conjunto de dados mais complexo de se classificar.

onde

$$\mathbf{w} = \begin{bmatrix} a & b & c \end{bmatrix}^\intercal$$

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix}^\intercal,$$

em que $\mathbf{w}$ é o vetor de coeficientes da reta, tratado aqui como o vetor que contém os pesos (do inglês *weights*: coeficientes $a$ e $b$ da reta para duas características) e o termo de polarização ou *bias* ($c$), $\mathbf{x}$ é o vetor de características dos dados (com 1 na última posição para proporcionar o produto escalar com $c$) e $y$ é o valor previsto[2]. Treinar um modelo de classificação linear para a resolução deste problema significa encontrar os valores do vetor $\mathbf{w}$ que aproximam a reta que melhor separa as classes. Para cada iteração $i$ sobre uma

[2] O valor previsto pode ser utilizado diretamente, em problemas de regressão (previsão de valores) ou pode ser mapeado para uma função de probabilidade ou para rótulos, em caso de classificação como o exemplificado.

unidade/porção do conjunto de dados, o vetor de pesos e *bias* é atualizado de forma que

$$\mathbf{w}_i = \mathbf{w}_{i-1} - \alpha \nabla C(\mathbf{w}_{i-1}) \tag{2.2}$$

em que:

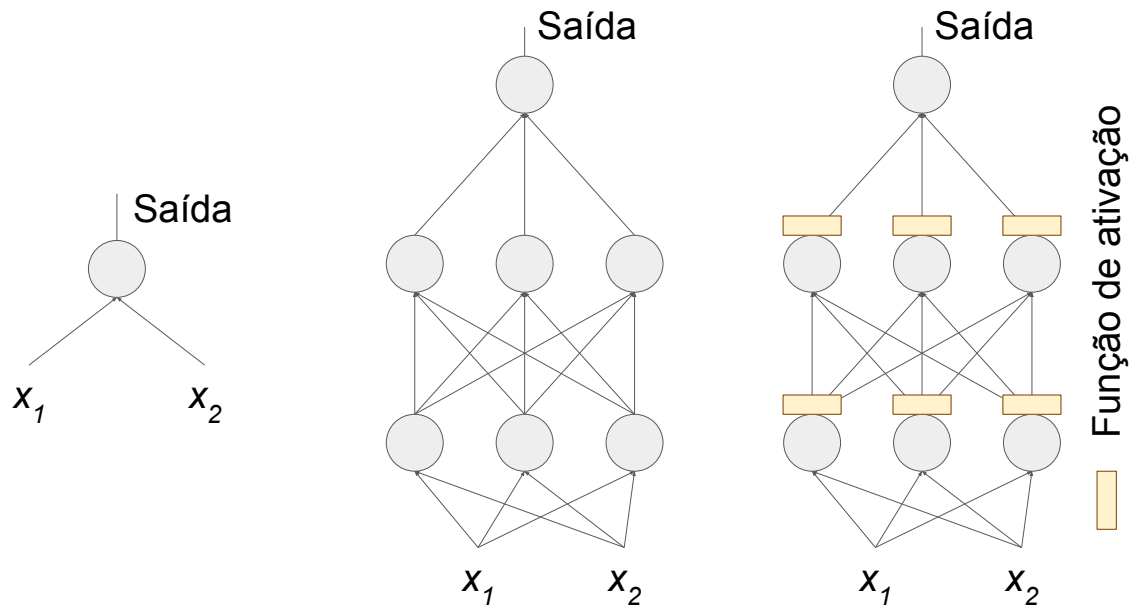
- $C(\mathbf{w})$ denota uma função de custo (*loss function*), que calcula o quão errônea é a predição sobre uma amostra;
- $\nabla$ é o símbolo que descreve o *vetor gradiente* da função $C(\mathbf{w})$, ou seja, o quanto $\mathbf{w}$ deve ser ajustado em relação a cada característica;
- $\alpha$ é a taxa de aprendizado (*learning rate*), que determina o quão intenso será o ajuste dos coeficientes.

Basicamente, a Equação 2.2 determina que os coeficientes são atualizados na direção oposta do aumento da função de custo segundo uma velocidade $\alpha$ que, se muito baixa, prolonga consideravelmente o tempo de aprendizado e, se muito alta, pode oscilar à distâncias cada vez maiores em torno do ponto ótimo de $\mathbf{w}$.

Porém, nem todo conjunto de dados pode ser satisfatoriamente separável por uma simples reta, como no caso da Figura 3b, que mostra um conjunto de dados separado em duas classes e com apenas uma característica $x$. A Figura 3c mostra o que acontece quando $x^2$ é adicionado como caraterística para este conjunto de dados.

Note que adicionar $x^2$ como característica faz com que a reta que separa as classes seja ditada pela Equação 2.1, com $x_1 = x^2$ e $x_2 = x$. Ou seja, adapta-se a solução linear a uma equação quadrática sem alterar sua forma. Isso mostra que é possível resolver problemas de classificação não-lineares inserindo não-linearidade diretamente no modelo linear (GÉRON, 2019; GOOGLE, 2020a).

Alguns problemas, porém, exigem soluções mais complexas, como mostra a Figura 3d. Uma forma eficaz de resolver esse tipo de problema é por meio da utilização de Redes Neurais Artificiais (*Artificial Neural Networks*). A Figura 4a ilustra um neurônio artificial, que representa nada mais do que a relação linear entre as características de entrada e a saída esperada. Já na Figura 4b, tem-se um modelo aparentemente mais complexo, com algumas camadas de neurônio e utilizando as saídas de cada camada como entradas na camada subsequente. No entanto, este modelo combina unidades lineares e, portanto, ainda é linear. A não-linearidade é introduzida nas saídas das camadas ocultas, como ilustra a Figura 4c. Em suma, as saídas de cada neurônio são submetidas a uma função não-linear, chamada de função de ativação (*activation function*), transformando as entradas da próxima camada. Um exemplo de função de ativação muito simples é a




(a) (b) (c)

Figura 4 – (a) Neurônio artificial: combinação linear das características de entrada; (b) Neurônios organizados em camadas, em que as entradas de cada neurônio da camada correspondem às saídas da camada anterior; (c) Similar à *b*, porém introduzindo uma função não-linear a operar sobre as saídas de cada neurônio.

ReLU (do inglês *Rectified Linear Unit*: Unidade Linear Retificada), que preserva o valor da saída caso ele seja não-negativo, e o transforma em zero caso contrário (GÉRON, 2019; GOOGLE, 2020d).

Em Redes Neurais, comumente emprega-se a camada totalmente conectada (*fully conected layer*), em que cada neurônio se conecta com todos os neurônios da camada subsequente. Redes mais profundas, isto é, com mais camadas, podem aproximar funções mais complexas, reforçando o conceito de Aprendizado Profundo (*Deep Learning*). Por outro lado, é maior também a chance de que a rede sofra um sobreajuste (*overfit*) aos dados de treinamento, aprendendo demais sobre eles e perdendo sua capacidade de generalização diante de dados novos (GÉRON, 2019; STANFORD, n.d.c). Uma forma comum de avaliar o poder de generalização de um modelo é dividir o conjunto de dados entre *conjuntos de treino, validação e teste*. Dessa forma, uma parte do conjunto total de dados que o modelo não conhece, por ser avulsa ao conjunto de treinamento, é utilizada para validar o modelo, ou seja, avaliá-lo em diferentes configurações/estruturas, e testá-lo, ou seja, verificar se o modelo já bem ajustado e treinado de fato se desempenha bem sobre dados totalmente desconhecidos (GÉRON, 2019; STANFORD, 2017c).

Quando se pretende aplicar Redes Neurais no âmbito de imagens, como neste trabalho, a utilização de redes comuns se torna inviável. A grande quantidade de pixeis que uma

imagem pode conter demandaria aprendizado sobre uma quantidade exorbitante de pesos, implicando em excessivo esforço computacional e grandes chances de sobreajuste. A resolução deste problema é atribuída à aplicação das *Redes Neurais Convolucionais* (GÉRON, 2019; GOODFELLOW; BENGIO; COURVILLE, 2016; STANFORD, n.d.a).

### 2.2.1 Redes Neurais Convolucionais

Os estudos de Hubel e Wiesel (vide Seção 2.1) revelaram descobertas interessantes sobre o córtex visual animal: o campo visual completo seria formado pela combinação de estímulos causados por *campos receptivos locais*. A cada neurônio estaria associado um campo receptivo, e neurônios em camadas mais profundas combinam a interpretação de padrões mais simples da camada anterior para interpretar padrões mais complexos. Uma Rede Neural Convolucional (CNN, do inglês *Convolutional Neural Network*), reproduz essas ideias no âmbito computacional (GÉRON, 2019).

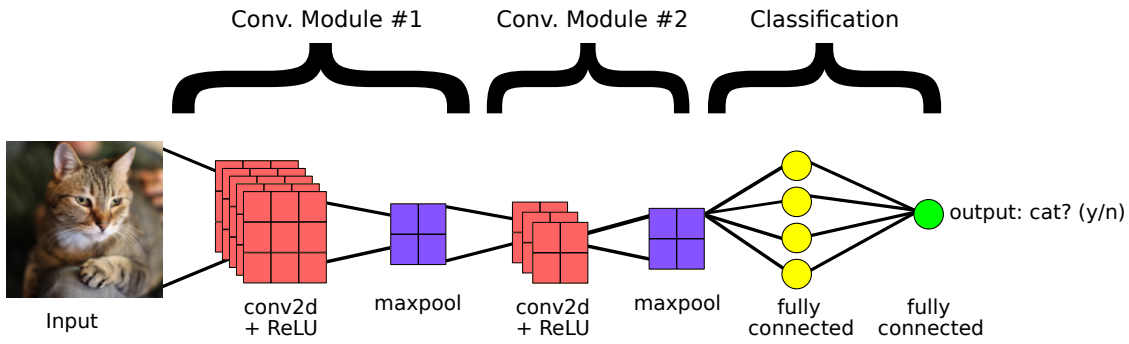
A Figura 5a[3] exemplifica a arquitetura de uma CNN: uma quantidade arbitrária de módulos contendo diferentes tipos de camadas. Os principais tipos de camadas são (GÉRON, 2019; GOOGLE, 2020c; STANFORD, n.d.a):

- **Camada convolucional**: considere a imagem de entrada como um mapa de características (*feature map*). Um filtro convolucional, uma pequena matriz de pesos, age como uma janela deslizante sobre o mapa de características, executando um produto escalar entre os pesos e a região abrangida do mapa (o campo receptivo), resultando em um outro mapa de características a ser aplicado à camada seguinte. A Figura 5b exemplifica a ação de um filtro convolucional $3 \times 3$ sobre um mapa de características $5 \times 5$ de entrada, gerando um mapa $3 \times 3$ de saída (Figura 5c). Geralmente, a função de ativação ReLU é aplicada sobre a saída antes que ela seja provida como entrada à próxima camada;

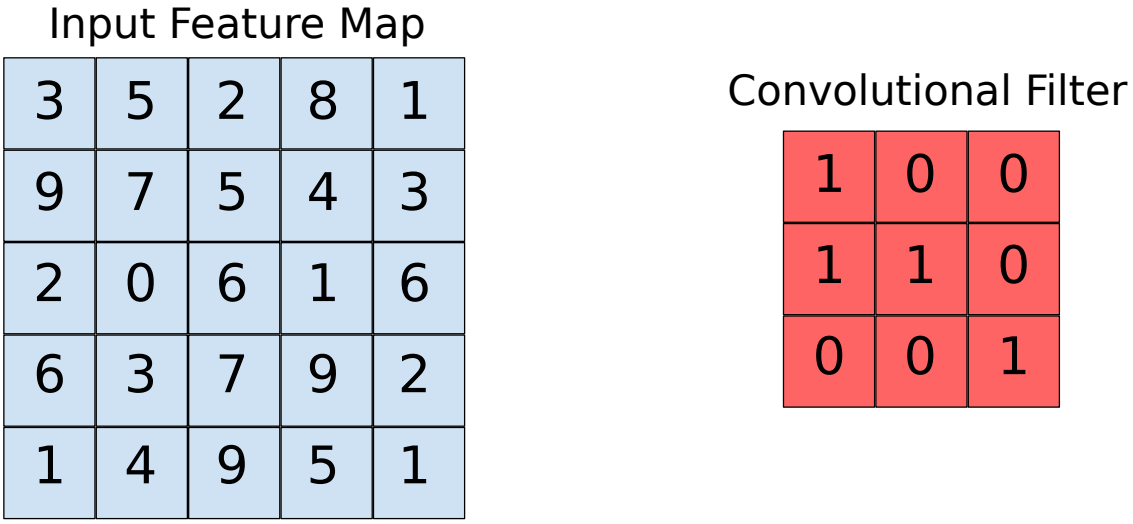
- ***Pooling***: a camada *pooling* visa reduzir as dimensões do mapa de características e, consequentemente, a quantidade de pesos a aprender e o esforço computacional. Novamente, uma janela deslizante atua sobre o mapa, executando uma operação sobre o campo receptivo. Uma operação comumente aplicada é a *max pooling*, cuja saída é igual ao maior valor presente no campo receptivo. A Figura 5d ilustra um exemplo de *max pooling* $2 \times 2$ executado sobre um mapa de características $4 \times 4$, gerando um mapa $2 \times 2$ de saída;

- **Camada totalmente conectada**: estrutura neural já apresentada na Seção 2.2. Depois que as todas camadas convolucionais e *pooling* atuam sobre a imagem de

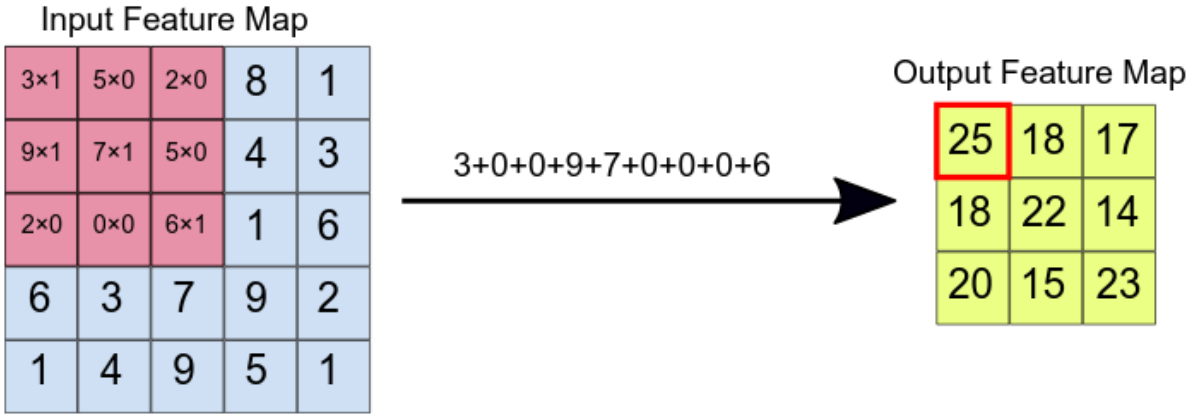
[3] Todas as imagens que compõem a Figura 5 são disponibilizadas por Google (2020c) sob a licença <https://creativecommons.org/licenses/by/4.0/>.




(a)

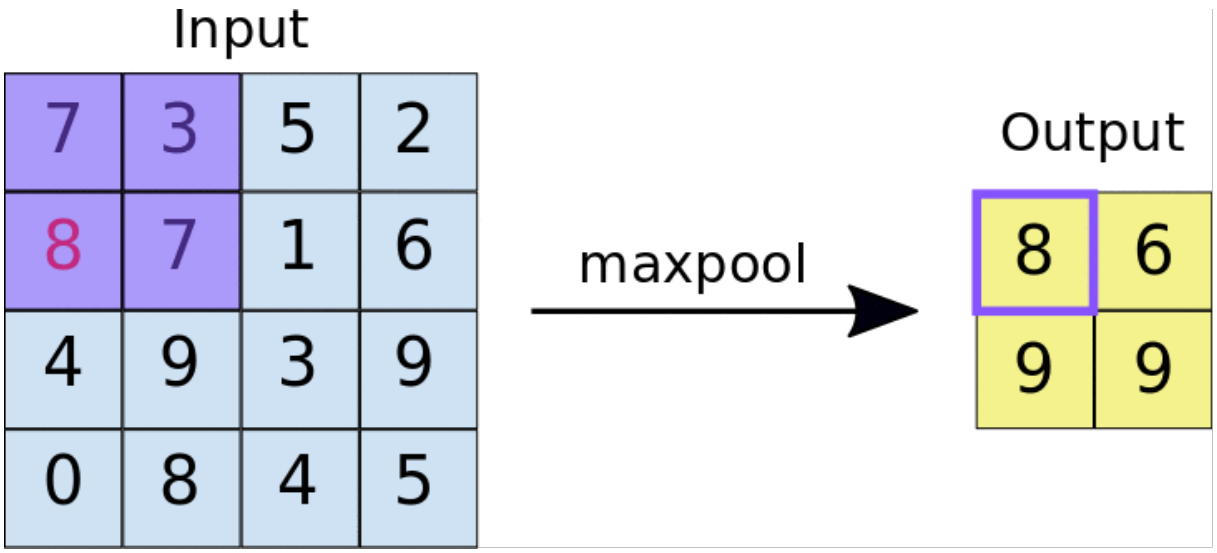



(b)




(c)

entrada, o mapa de características resultante é submetido a esta camada de modo que cada um de seus valores corresponda a uma característica de entrada, gerando a resposta final da rede sobre a imagem. Na Figura 5a, a CNN recebe uma imagem que exibe um gato e a saída da camada totalmente conectada responde à pergunta *esta imagem retrata um gato?*, calculando diferentes pontuações/probabilidades para as respostas *sim* ou *não*.




(d)

Figura 5 – (a) Exemplo de arquitetura de uma CNN. A imagem de entrada (*Input*) é submetida a alguns módulos convolucionais (*Conv. Module*) contendo uma camada convolucional (*conv2d* e uma *max pooling* (*maxpool*). O mapa de características resultante é submetido a uma camada totalmente conectada (*fully conected*) de classificação (*Classification*) cuja saída (*output*) responde à pergunta: "gato? (sim/não)" (*cat? (y/n)*); (b) Exemplo de mapa de características de entrada (*Input Feature Map*) e de filtro convolucional (*Convolutional Filter*); (c) Mapa de características de saída (*Output Feature Map*) gerado pela convolução do filtro sobre o mapa de entrada; (d) *Max pooling* sobre um mapa de características de entrada.

Como a grande quantidade de pesos que uma CNN precisa ajustar pode favorecer o sobreajuste, se mostra viável aplicar ao conjunto de treinamento um aumento de dados (*data augmentation*): novas imagens de treinamento podem ser geradas aplicando-se modificações realistas às imagens originais, como mover, rotacionar, redimensionar, inverter, ajustar brilho/contraste, entre outras. Isso aumenta consideravelmente o conjunto de treinamento e favorece a generalização (GÉRON, 2019; TENSORFLOW, 2021).

### 2.2.2 *ResNet*

Uma vez que uma variante da ResNet (*Residual Network*: Rede Residual) é utilizada neste trabalho, esta seção traz uma breve conceitualização baseada em sua publicação original (HE et al., 2016) e no conteúdo de Géron (2019).

A proposta de uma Rede Residual é bem simples: seja x a entrada de um determinado conjunto de camadas da rede e $f(\mathbf{x})$ a sua saída. Em vez de alimentar o próximo conjunto com $f(\mathbf{x})$, a ResNet emprega $f(\mathbf{x}) + \mathrm{x}$ como entrada, caracterizando um *aprendizado residual*. Com isso, o aprendizado se aproxima da função identidade (já que a entrada passa a fazer parte da saída) e pode se modelar de forma mais rápida e eficiente em direção

à real função que melhor descreve os dados.

Na publicação original, os autores apresentam resultados para configurações com diferentes profundidades. Neste trabalho, é utilizada a arquitetura de 101 camadas (ResNet-101).

## 2.3 Detecção de objetos

Como um dos principais problemas de Visão Computacional, a detecção de objetos é uma das várias tarefas para as quais as CNN's apresentam um satisfatório desempenho em sua execução. Diferentemente da classificação de imagens, que designa classes à cada imagem de entrada como um todo, a detecção de objetos visa identificar todos os objetos-alvo em uma imagem e identificar não apenas a que classe pertencem, mas também as coordenadas de uma caixa delimitadora (*bounding box*) que, como o nome sugere, demarca de forma retangular a região da imagem em que o objeto se encontra. Para refinar suas predições durante o treinamento, a rede precisa conhecer as caixas delimitadoras *rotuladas* de cada objeto, ou seja, a localização real do objeto na imagem e, naturalmente, a classe à qual o objeto pertence (STANFORD, 2017b).

Girshick et al. (2014) introduz a R-CNN, em que 'R' se refere a "Região", que utiliza um algoritmo para propor regiões prováveis de conter o objeto-alvo com base em segmentação e submete essas regiões para classificação por meio de CNN, enquanto também tenta prever as coordenadas da caixa delimitadora do objeto por meio de regressão linear. Girshick (2015) propõe a Fast-RCNN, que reduz em 9× o tempo de treino da R-CNN submetendo a imagem inteira a uma CNN e então trabalhando sobre as regiões propostas, a partir de então chamadas de Regiões de Interesse (RoI, do inglês *Regions of Interest*), projetadas no mapa de características resultante, gerando pequenos vetores de características que alimentam redes totalmente conectadas. Ren et al. (2015) propõe a Faster R-CNN, um algoritmo ainda mais rápido, que reduz o tempo de computação das RoI's fazendo com que uma CNN as compute sobre o mapa de características da imagem completa, descartando a etapa de busca de RoI's na imagem original. Com uma abordagem diferente, (REDMON et al., 2016) introduz a rede YOLO (*You Only Look Once*) que divide a imagem de entrada em subpartes e tenta prever a presença de objetos dentro de diferentes escalas de caixas delimitadoras sobre cada subparte, bem como tenta prever a real posição da caixa delimitadora verdadeira (STANFORD, 2017b).

### 2.3.1 Parâmetros e métricas de avaliação

É importante definir alguns parâmetros e métricas de avaliação utilizados no âmbito de detecção de objetos, inclusive neste trabalho. Segundo Padilla et al. (2021):

- Interseção sobre união (IOU, do inglês *Intersection over union*): razão entre a interseção e a união das áreas de uma caixa delimitadora de predição e uma de rotulação. Basicamente, determina o quanto a predição "se encaixa" na rotulação;
- Confiança: probabilidade de uma predição estar correta segundo os resultados do detector;
- Precisão: taxa de predições corretas entre todas as predições;
- Revocação: taxa de predições corretas entre todas as rotulações;
- *Mean Average Precision* (mAP): variando-se o limiar de confiança para que as predições sejam consideradas corretas, pode-se traçar curvas Precisão × Revocação para cada classe de objeto a ser detectada, baseada nos diferentes níveis de confiança. A área sobre esta curva informa a *Average Precision* (Precisão Média) para a classe. A mAP é a média aritmética das Precisões Médias;
- *F1-score*: média harmônica entre precisão e revocação.

### 2.3.2 *Faster R-CNN*

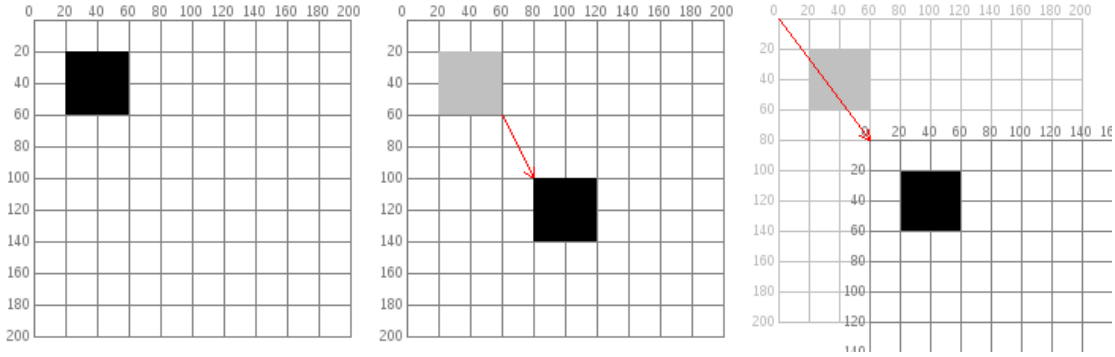
Esta seção detalha mais sobre a rede de detecção de objetos utilizada neste trabalho: a Faster R-CNN, já mencionada na Seção 2.3, de acordo com sua própria publicação de referência (REN et al., 2015).

A Faster R-CNN introduziu o conceito de Redes de Proposição de Regiões (RPN, do inglês *Region Proposal Networks*), que aprendem a determinar as RoI's a serem aplicadas ao algoritmo da Fast R-CNN. Uma vez computado o mapa de características da imagem, uma janela deslizante o percorre, e computa-se a saída (probabilidades de objetos e coordenadas da caixa delimitadora) para regiões do mapa chamadas de âncoras (*anchor boxes*), que adaptam a janela para diferentes escalas e proporções altura-largura.

Na época de sua publicação, a Faster R-CNN havia atingido desempenho estado-da-arte em conhecidos *datasets* de imagens, bem como atingiu o primeiro lugar em importantes competições. Além disso, a rede foi utilizada por Torres et al. (2019) no trabalho no qual este trabalho se baseia.

## 2.4 Computação gráfica e dados sintéticos

Eck (2016) define *computação gráfica* como o uso de computadores para criação e manipulação de imagens. A computação gráfica pode ser aplicada às mais diferentes áreas, estando dominantemente presente em, por exemplo, jogos eletrônicos, desenhos e filmes animados, efeitos visuais, *design* de peças e produtos, geração de imagens médicas e visualização de informações naturalmente não-visuais (MARSCHNER; SHIRLEY, 2018).



(a) (b) (c)

Figura 6 – (a) Retângulo desenhado sobre o sistema de coordenadas da tela; (b) Translação do retângulo modificando-se suas coordenadas; (c) Translação do retângulo por translação do sistema de coordenadas.

As formas e transformações básicas de objetos gráficos consideram a matriz de pixeis que constitui uma janela da tela do computador como um sistema de coordenadas $(x, y)$, em que $x$ é a coordenada horizontal e $y$ a vertical. Conjuntos de coordenadas podem constituir tanto formas simples como linhas, triângulos, retângulos e elipses quanto formas mais complexas (SHIFFMAN, n.d.a; SHIFFMAN, n.d.c). A posição, angulação e tamanho das formas podem ser transformados, variando a exibição de um mesmo objeto na cena virtual. A Figura 6a[4] ilustra um retângulo desenhado entre as coordenadas $(20, 20)$ (canto superior-esquerdo) e $(40, 40)$ (canto inferior-direito), enquanto as Figuras 6b e 6c exemplificam dois modos de transladar o objeto, respectivamente: movendo suas coordenadas diretamente ou movendo o sistema de coordenadas (opção mais vantajosa em caso de necessidade de reprodução do objeto em diferentes disposições) (EISENBERG, n.d.).

De acordo com Shiffman (n.d.b), o espaço de coordenadas também pode ser estendido para o formato tridimensional. A adição de um eixo $z$ que simula profundidade em pixeis viabiliza a representação de formas tridimensionais básicas, como prismas retangulares, e formas mais complexas, compostas por polígonos. Algumas funções gráficas se mostram interessantes para a construção do espaço gráfico tridimensional:

- Textura: imagem aplicada à superfície de uma forma 3D. Por exemplo, uma imagem da superfície da Terra aplicada a uma esfera;
- Iluminação: aplicação de efeitos de luz na cena que simulam efeitos de iluminação reais. Pode ser representada de várias formas, como: a luz ambiente, que se distribui uniformemente pela cena; a luz direcional, que se manifesta mais intensamente sobre superfícies atingidas em determinada direção; e o holofote, luz direcional concentrada

[4] Todas as imagens que compõem a Figura 6 são disponibilizadas por Eisenberg (n.d.) sob a licença <http://creativecommons.org/licenses/by-nc-sa/4.0/>.

por um ângulo de abertura;

- Perspectiva: projeção que exibe em menor escala objetos mais distantes;
- Câmera: ponto de vista da cena. Simulada definindo-se as coordenadas da posição do olho e do ponto para o qual se olha, bem como a direção na qual a câmera será verticalmente alinhada.

Neste trabalho, formas combinadas reproduzem elementos comuns do trânsito que, estando devidamente posicionados e combinados a efeitos de câmera, perspectiva e iluminação, compõem cenas 3D sintéticas de tráfego sinalizado por semáforos. As cenas fazem parte de um conjunto de dados sintéticos destinados a alimentarem uma rede convolucional de detecção de semáforos a ser aplicada em contextos de tráfego do mundo real.

## 2.5 Trabalhos relacionados

Em seu trabalho, Jensen et al. (2015) compara um método de detecção de semáforos baseado em aprendizado com trabalhos até então mais usuais, que se baseavam em características do modelo de um semáforo. Assim como Li et al. (2017), utiliza o algoritmo de ACF (*Aggregate Channel Features*) para detectar semáforos através de uma correspondência de modelo, enquanto trabalhos mais antigos como (GOMEZ et al., 2014) se baseiam em cores e formas do semáforo. Barnes, Maddern e Posner (2015) emprega o uso de SVM e HoG (*Histogram of Oriented Gradients*) como outra forma de promover detecção de semáforos baseada em aprendizado.

Trabalhos mais recentes já demostram que a utilização de consolidados algoritmos detectores de objetos, como YOLO e Faster R-CNN podem ser efetivos na execução da tarefa. Possatti et al. (2019) combina YOLO com regiões de localização geográfica previamente mapeadas de semáforos para detectá-los e inferir seu estado, enquanto Behrendt, Novak e Botros (2017) emprega o algoritmo em regiões específicas da imagem onde a presença de um semáforo é mais provável. Já Pon et al. (2018) modifica o módulo de classificação da Faster R-CNN para classificar objetos entre semáforos e placas de trânsito e então classificar o objeto com base na resposta da classificação anterior. Apesar de bem sucedidos, os métodos que se baseiam em detectores de Aprendizado Profundo comumente requerem uma extensa gama de dados rotulados, além de sofrerem com o desbalanceamento de classes (no âmbito de semáforos, escassez de exemplares em estado amarelo) que prejudica o aprendizado dos detectores.

Algumas ferramentas, como a Microsoft *Visual Object Tagging Tool* (VoTT)[5] e a

[5] https://github.com/Microsoft/VoTT

Scalabel[6], foram propostas a fim de reduzir o esforço laboral na anotação de imagens e vídeos. Esse problema também foi explorado por alguns métodos como os de Wang et al. (2018), que propõe sobrepor regiões não rotuladas de uma imagem sobre imagens já rotuladas e avaliar a qualidade de um detector no contexto montado, e de Oquab et al. (2015) e Sangineto et al. (2018), que propõem rotulação não a nível de objeto mas a nível de imagem e utilizam técnicas para propor a localização do objeto rotulado na imagem. Chen et al. (2018) e Kang et al. (2019) propõem métodos que envolvem aprendizado satisfatório com poucos objetos rotulados ( *Few-shot* [ou *Low-shot*] *learning*). O primeiro combina um regressor de caixa delimitadora unificado para todas as classes em vez de individual por classe, como na Faster R-CNN, com o classificador desta, de modo que o modelo aprenda mais rápido com menos exemplos. Por sua vez, o segundo propõe um detector que, além de gerar mapas de características dos poucos objetos rotulados, também pondera os níveis de relevância dessas características, facilitando a identificação de novos exemplares.

Por sua vez, o desbalanceamento de classes é um problema já antigo (HE; GARCIA, 2009) de aprendizado para o qual trabalhos recentes propõem soluções. Técnicas comuns para lidar com o problema incluem reamostrar os exemplos de determinadas classes reproduzindo exemplos de classes pouco frequentes e/ou descartando exemplos de classes mais frequentes ou atribuir diferentes custos para diferentes classes durante o cálculo da função de erro de treinamento (HUANG et al., 2016). Ainda, Wang, Ramanan e Hebert (2017) demonstra uma forma de lidar com desbalanceamento baseada em aprendizado, transferindo conhecimento sobre as classes mais frequentes para as menos frequentes.

Para lidar com as questões levantadas no âmbito de placas de trânsito, Torres et al. (2019) propõem o treinamento do detector Faster R-CNN por meio de dados rotulados sem coleta de dados do mundo real e sem esforço humano de rotulação sobrepondo aleatoriamente modelos de placas de trânsito sobre imagens de contextos arbitrários. Este trabalho apresenta proposta similar no âmbito de semáforos, porém adicionando um fator de melhoria baseado no aprendizado não apenas sobre os objetos dispostos automaticamente sobre os planos de fundo, mas também sobre o contexto no qual estão inseridos: toda uma contextualização de uma cena de trânsito, incluindo os semáforos como objetos de interesse, é gerada sinteticamente e sobreposta ao plano de fundo.

[6] https://www.scalabel.ai/

# 3 Detecção de semáforos utilizando contexto sintético

O trabalho proposto visa combinar um plano de fundo descrito por uma imagem natural não relacionada à trânsito a um primeiro plano que denota uma cena de trânsito gerada através de técnicas simples e não-realistas de computação gráfica. A cena representa elementos comuns de um contexto de tráfego, como pista, veículos, faixas de pedestre e de sinalização, postes e semáforos. Uma vez definidos, cada plano de fundo é combinado a um primeiro plano de forma a gerar um completo *dataset* capaz de treinar uma CNN. O modelo treinado pode então ser usado para detectar e classificar o estado de semáforos em imagens contendo cenas de tráfego do mundo real. A Figura 7[7] apresenta uma visão geral do método.

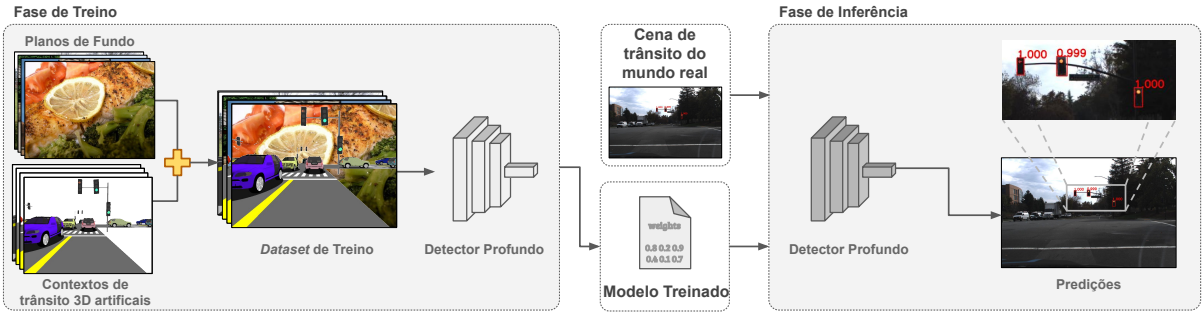



Figura 7 – Visualização geral do método.

A Seção 3.1 apresenta uma descrição das imagens naturais utilizadas como plano de fundo; a Seção 3.2 descreve o processo de geração das cenas de tráfego sintéticas utilizadas como primeiro plano; e, finalmente, a Seção 3.3 descreve como plano de fundo e primeiro plano são pré-processados e combinados de modo a construir o *dataset* de treino.

[7] Esta imagem contém adaptações de imagens dos *datasets* Udacity Driving (GONZALEZ et al., 2020), submetido à licença MIT (<https://github.com/udacity/self-driving-car/blob/master/datasets/LICENSE.md>) e COCO (LIN et al., 2014), adaptáveis e divulgáveis sob a licença <https://creativecommons.org/licenses/by/3.0/>, originalmente disponíveis em:
<http://farm4.staticflickr.com/3212/2897824940_819eb4b0a0_z.jpg>
<http://farm9.staticflickr.com/8448/7979211072_cdbf555f05_z.jpg>
<http://farm4.staticflickr.com/3819/8841141670_c5cf991e43_z.jpg>
<http://farm5.staticflickr.com/4026/4493201538_f4da305527_z.jpg>

## 3.1 Plano de fundo

O conjunto de planos de fundo é composto por um grande volume de imagens que representam uma enorme variedade de cenários e situações, como paisagens naturais, comida, pessoas praticando esportes, etc. A única restrição é que o conjunto não inclua cenas relacionadas à trânsito, de modo a evitar que elementos do plano de fundo conflitem com elementos do primeiro plano durante o processo de aprendizado de rede. A obtenção de um conjunto representativo de planos de fundo é facilitada pela disponibilidade de vastos *datasets* de imagens naturais acessíveis ao público.

## 3.2 Primeiro plano

A ideia do primeiro plano é reproduzir a visão de um motorista de um cruzamento sinalizado por semáforos. Cada imagem de primeiro plano apresenta, sobre um fundo transparente, sua própria disposição aleatória dos seguintes elementos:

- Pista;
- Postes de semáforo;
- Semáforos;
- Veículos;
- Iluminação;
- Câmera.

Utilizando o ambiente *Processing*[8] de computação gráfica, cada elemento da cena é construído e combinado ao demais de acordo com o explicado nas subseções seguintes. Alguns parâmetros utilizados na construção dos elementos foram definidos em função da altura e largura da imagem final do primeiro plano. Nas próximas subseções, considere $A$ como a altura em pixels do primeiro plano final e $L$ como sua largura em pixels. Além disso, por simplificação, algumas variáveis serão apresentadas nas figuras explicativas e podem ser compartilhadas entre diferentes figuras.

### 3.2.1 Pista

A pista representada é composta por dois a quatro trechos, representados por retângulos. Sobre cada trecho, retângulos finos representam faixas de sinalização e/ou de pedestres. Cada trecho apresenta duas vias de sentidos opostos (mãos), com uma quantidade

[8] <https://processing.org/>

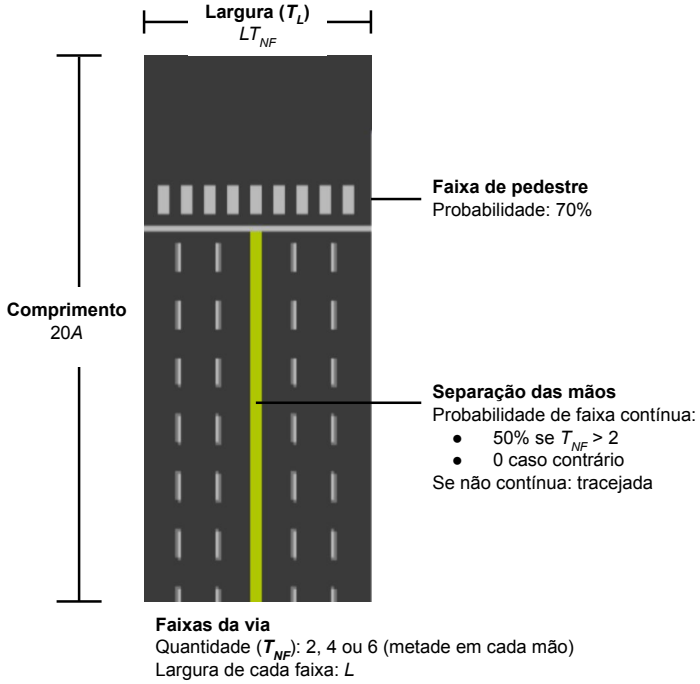



Figura 8 – Exemplo de trecho da pista e parâmetros utilizados para sua construção. Variáveis destacadas: $A$ - altura da imagem final; $L$ - largura da imagem final; $T_L$ - largura do trecho de pista; $T_{NF}$ - quantidade de faixas da via.

definida aleatoriamente, entre uma e três, de faixas em cada mão. A Figura 8 apresenta um exemplo de trecho de pista, indicando os parâmetros adotados para construi-lo e seus possíveis valores.

Dos potenciais quatro trechos gerados em cada cena, um deles, chamado de **trecho sul**, é mandatoriamente representado, uma vez que neste trecho se encontrará a primeira pessoa da cena, representada pela câmera. Juntamente ao trecho sul, pelo menos um entre os três seguintes também é representado na cena: o **trecho oeste**, perpendicular e à esquerda do trecho sul, o **trecho leste**, perpendicular e à direita do trecho sul, e o **trecho norte**, paralelo e em direção oposta ao trecho sul. Todos os trechos são gerados de forma similar ao exibido na Figura 8 e os trechos oeste, norte e leste são rotacionados em um ângulo múltiplo de 90° em torno do centro de sua aresta superior. A Figura 9 mostra os quatro trechos possíveis junto às suas respectivas probabilidades e ângulos de representação.

### 3.2.2 Postes de semáforo

A cada trecho da pista pode estar associado no máximo 1 poste de semáforo (obrigatório para o trecho sul), que pode estar disposto à sua direita ou à sua esquerda. Seu eixo é perpendicular ao plano da pista, e ele pode conter até 1 semáforo e uma extensão horizontal na parte superior contendo pelo menos 1 semáforo. A Figura 10 descreve a parametrização utilizada na renderização dos postes.

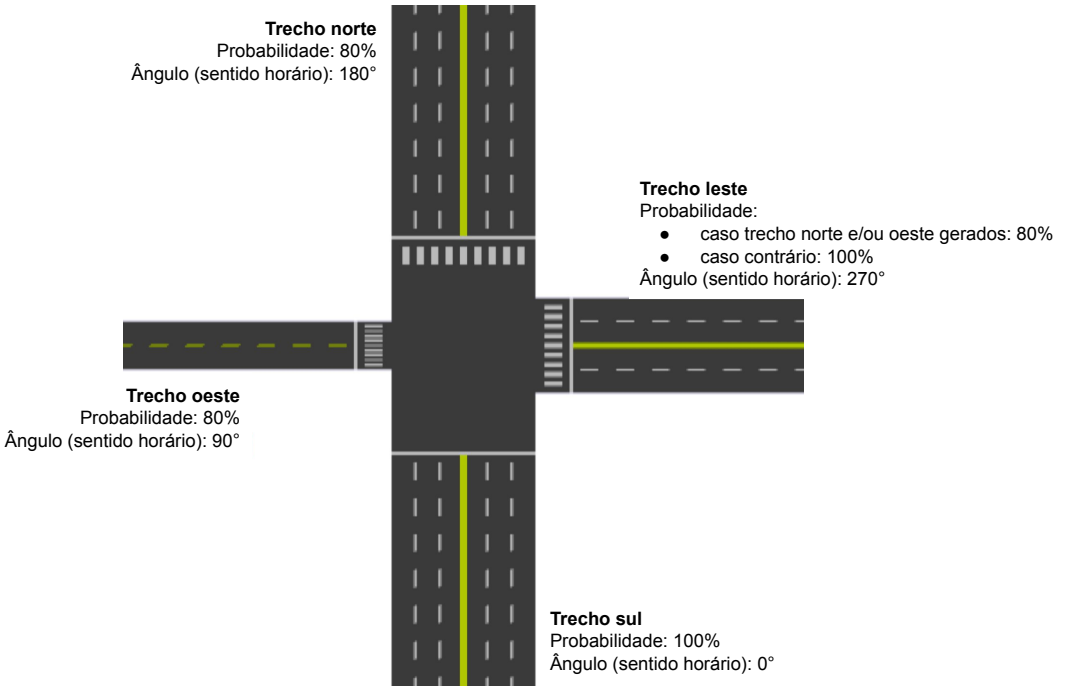



Figura 9 – Exemplos de possíveis trechos da cena e definição de suas probabilidades e ângulos de representação em relação ao trecho sul.

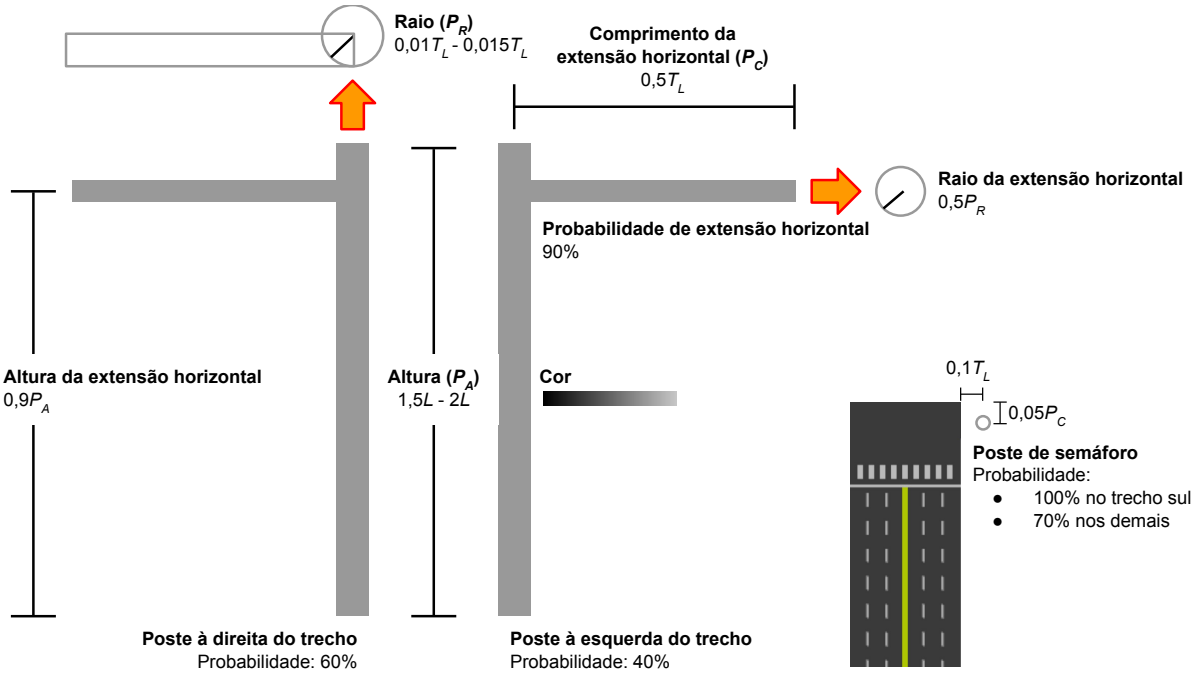



Figura 10 – Parametrização utilizada na construção e disposição dos postes de semáforo. Variáveis destacadas: $L$ - largura da imagem final em pixeis; $P_A$ - altura do poste; $P_C$ - comprimento da extensão horizontal; $P_R$ - raio do poste; $T_L$ - largura do trecho.

### 3.2.3 Semáforos

Os semáforos gerados para a construção das cenas correspondem ao modelo negro vertical de três bulbos. Cada semáforo tem igual probabilidade de ser gerado em estado vermelho, amarelo ou verde. Considere como *bulbo principal* aquele que, quando aceso, define o estado do semáforo: superior para vermelho, central para amarelo e inferior para verde. Um semáforo pode pertencer a qualquer um dos três tipos:

- Semáforo com cronômetro (Figura 11a): além de ter o bulbo principal completamente aceso, segmentos de dígitos são acesos aleatoriamente (não necessariamente representando um dígito conhecido) no bulbo central. Apenas semáforos vermelhos ou verdes podem ser gerados com cronômetro;
- Semáforo comum (Figura 11b): tem o bulbo principal completamente aceso e os demais apagados;
- Semáforo com seta direcional (Figura 11c): o bulbo principal emite luz na forma de uma seta apontando aleatoriamente para a esquerda ou para a direita, enquanto os demais ficam apagados. Apenas semáforos vermelhos ou verdes podem ser gerados com seta direcional.

Essas figuras também destacam os principais elementos gráficos que compõem cada parte dos semáforos:

1. Corpo: caixa escura de dimensões aleatoriamente definidas: altura $S_A \in [280, 320)$ pixeis, comprimento $S_C \in [80, 120)$ pixeis e largura $S_L = 0,2S_A$;

2-5. Bulbos: esferas de raio $S_R = 0,1S_A$ que interceptam a face frontal do semáforo de modo que metade de seu volume fique visível. São centralizadas na horizontal e uniformemente distribuídas na vertical em relação ao corpo do semáforo. As variedades a seguir são exibidas nas figuras:

   2. Bulbo apagado: esfera escura levemente translúcida;
   3. Bulbo aceso: esfera de material emissivo (que emite luz) de cor compatível com o estado do semáforo;
   4. Bulbo com cronômetro: dentro de um bulbo apagado, prismas retangulares simulam segmentos de dois dígitos. Cada segmento pode estar apagado — escuro — ou aceso — da cor do bulbo aceso;
   5. Bulbo com seta: prismas retangulares de material emissivo dentro de bulbos apagados simulam as setas direcionais.

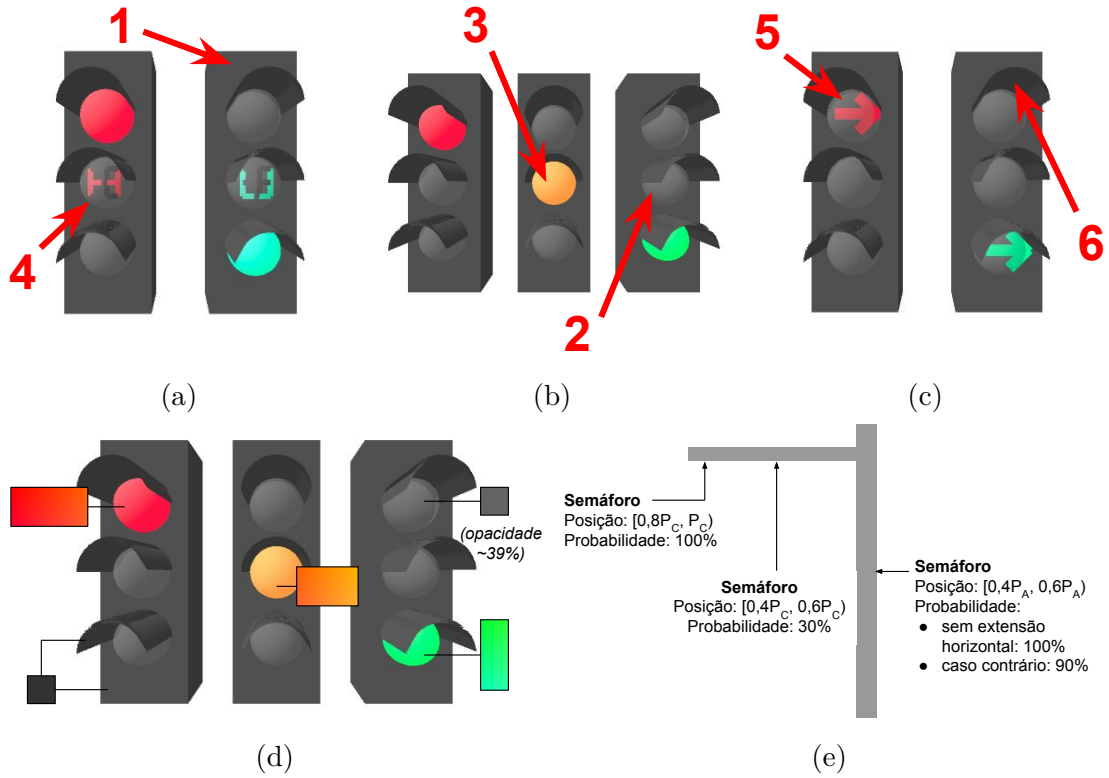



Figura 11 – (a) Exemplos de semáforos com cronômetro; (b) exemplos de semáforos comuns; (c) exemplos de semáforos com seta direcional; (d) cores possíveis para cada elemento dos semáforos; (e) posições possíveis para os semáforos nos postes. 1. Corpo do semáforo; 2. bulbo apagado; 3. bulbo aceso; 4. bulbo com cronômetro; 5. bulbo com seta direcional; 6. cobertura de bulbo. Variáveis destacadas: $P_A$ - altura do poste; $P_C$ - comprimento da extensão horizontal do poste.

6. Cobertura de bulbo: arco de cilindro de 120° e comprimento $S_{CC} = 0,2S_A$ que cobre cada bulbo.

A cor que define o estado de cada semáforo é retirada aleatoriamente de um intervalo de cores definido. Já as cores dos demais elementos do semáforo são uniformes. A Figura 11d mostra as cores que cada elemento do semáforo pode assumir para cada um dos três estados possíveis.

Por fim, a Figura 11e mostra as possíveis disposições dos semáforos nos postes. Em resumo, o poste pode possuir de um a três semáforos, nas posições: (i) região central do poste; (ii) região central da extensão horizontal e (iii) próximo à extremidade da extensão horizontal. O semáforo na posição (i) é garantido caso não haja extensão horizontal. Caso contrário, o único semáforo garantido é o da posição (iii), enquanto cada um dos demais têm uma probabilidade definida de serem inclusos.

[9] <https://free3d.com/3d-model/bmw-x5-1542.html>

(a) (b)

Figura 12 – Modelo 3D de veículo (a) carregado sem textura; (b) após ser colorido aleatoriamente e ter faróis traseiros e sombra adicionados.

### 3.2.4 Veículos

Os veículos dispostos na cena provém de um modelo 3D que se encontrava publicamente disponível[9] na época de execução do projeto. O objeto corresponde a um modelo BMW X5, carregado sem textura, como mostrado na Figura 12a. O objetivo principal é dispor várias cópias do modelo em cores e posições diferentes na cena. Cada cópia tem sua escala aumentada em cinco vezes e tem suas faces coloridas de modo que a lataria apresente uma cor aleatória no espectro RGB e os demais elementos apresentem cores padrão (por exemplo, pneus negros) e um nível de translucidez nos faróis e janelas. Faróis traseiros, modelados como um prisma retangular amarelo de material emissivo dentro de um prisma retangular vermelho translúcido complementam a cópia. Além disso, uma elipse escura simula a sombra projetada pelo veículo. A Figura 12b mostra um exemplo de cópia do modelo colorida em azul.

O posicionamento de veículos em cada trecho da pista se dá conforme o seguinte protocolo:

- cada faixa (incluindo a região do cruzamento) tem uma quantidade aleatória $V_N$ de veículos entre zero e cinco;
- o comprimento da faixa é dividido em $V_N + 1$ partes de modo que cada veículo seja centralizado sobre cada divisão, assegurando equidistância;
- a direção do veículo na faixa é tal que o suposto motorista conduza na mão à sua direita;
- seja $V_{pos}$ a posição na faixa prevista para o veículo, $C_{pos}$ a posição da câmera no plano da pista e $V_D$ a distância entre veículos na faixa. Se $V_{pos}$ está dentro da margem $C_{pos} \pm 0,5V_D$, o veículo não é incluído. Isso garante que um veículo não obstrua a visão da câmera.

### 3.2.5 Iluminação

Cada cena possui uma configuração aleatória de iluminação, aumentando ainda mais a variabilidade. Duas características de iluminação são adicionadas à cena:

- uma luz direcional branca orientada aleatoriamente na vertical e na direção leste-oeste/oeste-leste, mas sempre no sentido norte-sul;
- uma luz ambiente em escala de cinza aleatória.

### 3.2.6 Câmera

A câmera da cena é centralizada em largura sobre uma faixa aleatória da mão direita do trecho sul, à uma distância aleatoriamente definida dentro da margem $[8, 9A, 21A)$ do cruzamento. A altura em relação à pista está dentro da margem $[200, 300)$ pixels. A câmera aponta para o centro do cruzamento.

## 3.3 Geração de dados

A geração de cada imagem do *dataset* final começa pela seleção aleatória de uma imagem de plano de fundo e uma de primeiro plano para serem combinadas. Para garantir maior variabilidade de dados, alguns processos de aumento de dados foram aplicados tanto em cada plano quanto na imagem final, sendo eles: transformação de brilho, ruído de histograma e borramento. As subseções seguintes descrevem cada um destes processos, bem como o processo final de combinação dos planos. Os intervalos de valores descritos para cada parâmetro foram escolhidos empiricamente (em maior parte, mantidos os mesmos de Torres et al. (2019)). Para auxiliar a inspeção visual dos aumentos e a determinação dos intervalos de valores dos parâmetros, as dimensões dos planos de fundo e imagens de primeiro plano são reescaladas para $1280 \times 960$ pixeis (as dimensões originais, individualmente descritas ao decorrer da Seção 4, são tais que esse redimensionamento não altere as proporções da imagem).

### 3.3.1 Transformação de brilho

A cada pixel de cada canal do plano de fundo é adicionado um mesmo valor real $A_{PF} \in [-120, 120)$. Cada resultado é multiplicado por um valor $M_{PF} \in [0,75; 1,25)$. Ao primeiro plano se busca conferir um destaque discretamente maior, adicionado-se a cada pixel de cada canal um valor $A_{PP} = A_{PF} + 40$ e multiplicando os resultados por $M_{PP} = M_{PF}$.

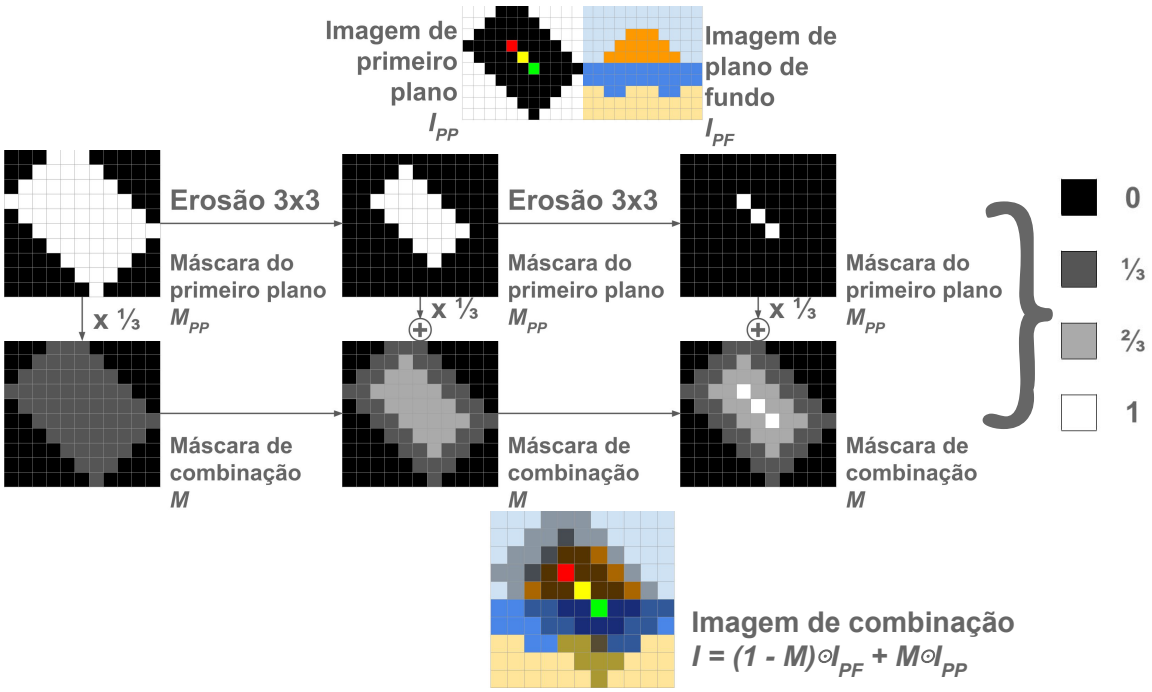



Figura 13 – Processo de combinação de plano de fundo e primeiro plano.

### 3.3.2 Ruído de histograma

A cada pixel de cada canal do primeiro plano é adicionado um valor inteiro aleatório $R_{PP} \in [-15, 15)$. Isso perturba levemente a uniformidade de cor que um objeto gerado sinteticamente possui ao longo de seu corpo, provendo maior realismo.

### 3.3.3 Borramento

O realismo do primeiro plano também é intensificado pela aplicação de um filtro Gaussiano de borramento com um desvio padrão real aleatório $DB_{PP} \in [0, 3)$. Dessa forma, a transição entre elementos sintéticos adjacentes de diferentes cores deixa de ser brusca e se torna mais suave. Após o processo de combinação dos planos, descrito na próxima subseção, filtragem similar (selecionando novamente o valor de $DB_{PP}$) é aplicada na imagem completa, suavizando a transição entre primeiro plano e plano de fundo.

### 3.3.4 Combinação dos planos

A combinação dos planos é feita de modo que o primeiro plano seja sobreposto ao plano de fundo e os limites entre os planos sejam suavizados. Considere: (i) $I_{PP}$ a imagem correspondente ao primeiro plano; (ii) $I_{PF}$ a imagem que corresponde ao plano de fundo; (iii) $M_{PP}$ uma máscara, que consiste em uma matriz $L \times A$ que emprega 1's nas posições correspondentes aos pixeis não-transparentes do primeiro plano e 0's nas demais; (iv) $M$ a máscara correspondente à imagem combinada final, tal que, inicialmente, $M = M_{PP}/3$; e (v) $I$ a imagem combinada final. A Figura 13 ilustra o processo de combinação.

Basicamente, $M_{PP}$ passa por dois processos subsequentes de erosão, utilizando um *kernel* quadrado $3 \times 3$. Após cada um deles, $\frac{1}{3}$ do resultado é somado a $M$. Então, a imagem final corresponde a $I = (1 - M) \odot I_{PF} + M \odot I_{PP}$, de modo que os pixeis que correspondem aos limites entre os planos resultem de uma contribuição ponderada de cada um deles.

## 3.4 Rotulação dos dados

Como mencionado nas Seções 3.2.2 e 3.2.3, todos os trechos podem conter postes de semáforo, e cada poste deve conter pelo menos um semáforo. Naturalmente, apenas o poste associado ao trecho sul apresenta semáforos com a face frontal predominantemente voltada para a câmera. Por isso, apenas semáforos nesta condição (sobre o poste do trecho sul) são rotulados.

Cada caixa de rotulação descreve tanto o estado do semáforo quanto suas as coordenadas 2D na imagem final. O estado, como já mencionado, é definido aleatoriamente no momento de geração do modelo do semáforo. Por sua vez, as coordenadas da caixa de rotulação são determinadas conforme a seguir:

- Tomam-se as coordenadas 3D dos quatro vértices da face frontal do semáforo;
- Calculam-se as posições 2D dessas coordenadas na imagem final, obtendo-se um conjunto $C_{2D}$ de quatro coordenadas $xy_{posição} = (x_{posição}, y_{posição})$, em que $x_{posição}$ é o valor horizontal da coordenada e $y_{posição}$ o valor vertical: $C_{2D} = \{xy_{superior-esquerda}, xy_{superior-direita}, xy_{inferior-direita}, xy_{inferior-esquerda}\}$;
- Considere $x_{esq}$ o valor de $x$ mais à esquerda, $x_{dir}$ o valor de $x$ mais à direita, $y_{sup}$ o valor de $y$ mais acima e $y_{inf}$ o valor de $y$ mais abaixo. O conjunto $C_{CA}$ de coordenadas da caixa de rotulação é tal que a caixa seja o menor retângulo (sem inclinação) que contém $C_{2D}$: $C_{CA} = \{(x_{esq}, y_{sup}), (x_{dir}, y_{sup}), (x_{dir}, y_{inf}), (x_{esq}, y_{inf})\}$.

Dada a natureza da geração da cena, existem duas ocasiões principais em que rotulações correspondentes a semáforos podem ser total ou parcialmente oclusas:

- Quando o semáforo é gerado fora do limite visível da imagem;
- Quando veículos sobrepõem os semáforos.

No segundo caso, a sobreposição é verificada observando-se a interseção entre a caixa delimitadora do semáforo e caixas delimitadoras calculadas de forma similar para cada veículo do trecho sul, considerando a largura e altura do veículo a partir do centro

de seu volume. Em ambos os casos, o semáforo não é incluído na cena caso a área de sua caixa delimitadora fosse sofrer pelo menos 50% de oclusão.

# 4 Metodologia Experimental

Diferentes experimentos foram realizados com o objetivo de avaliar o desempenho do método proposto em relação a detectores treinados com retratos realistas de trânsito e estudar alguns dos fatores determinantes para a eficácia do modelo sintético.

A Seção 4.1 descreve os *datasets* montados e/ou aproveitados para a execução dos experimentos; a Seção 4.2 resume o ferramental e parametrização experimentais utilizados; a Seção 4.3 explica as métricas e procedimentos empregados na avaliação de desempenho dos modelos, incluindo a validação-de-caixa; a Seção 4.4 descreve os experimentos realizados; e, por fim, a Seção 4.5 expõe as características dos recursos computacionais que viabilizaram a execução dos experimentos.

## 4.1 *Datasets*

Dentre os tipos de *datasets* utilizados, estão: os ***datasets* de planos de fundo**, conjuntos de dados públicos utilizados para a obtenção de planos de fundo; os ***datasets* de treino e validação**, montados para preparar os modelos experimentados e validar seu desempenho; e os ***datasets* de teste e validação-de-caixa** utilizados para testar os modelos em cenas de trânsito do mundo real após submetê-lo ao processo de *validação-de-caixa*, conceito que será explicado posteriormente. A Figura 14 ilustra os *datasets* utilizados, as relações entre eles e sua função entre conjunto de treino, validação, validação-de-caixa ou teste.

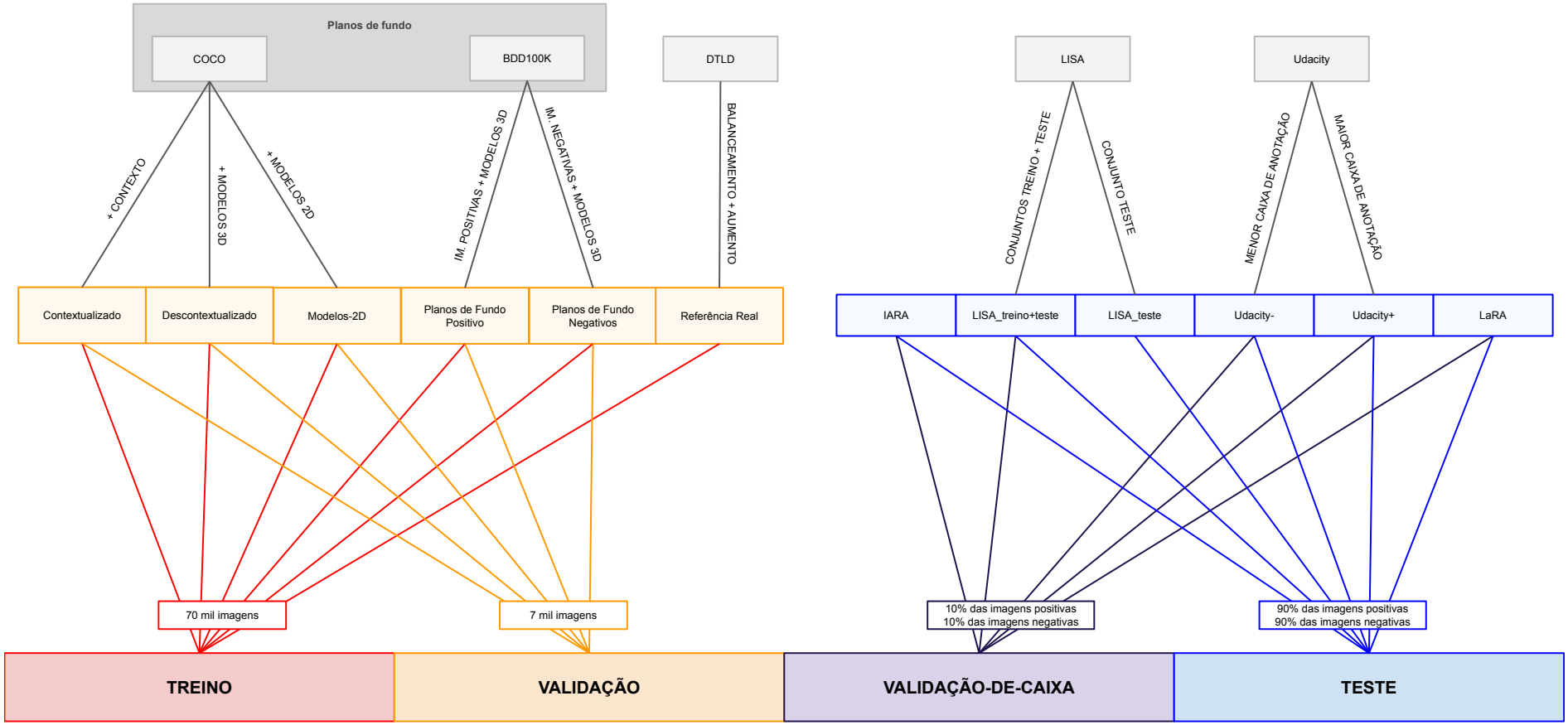



Figura 14 – Ilustração dos *datasets* utilizados e suas funções.

### 4.1.1 *Datasets* de planos de fundo

As subseções seguintes descrevem *datasets* disponíveis publicamente utilizados para a obtenção de planos de fundo. O *dataset Microsoft Common Objects in Context* (COCO) foi usado como a fonte de planos de fundo não relacionados à trânsito usados para a construção do modelo proposto, já o *dataset Berkeley DeepDrive* (BDD100K) serviu para a obtenção de planos de fundo exibindo cenas de trânsito, para estudo do efeito de se utilizar planos de fundo dentro do contexto de treino.

#### 4.1.1.1 *Microsoft Common Objects in Context* (COCO)

Os planos de fundo não relacionados à trânsito para execução da proposta foram retirados da versão de 2017 do *dataset* COCO (LIN et al., 2014). Este *dataset* é composto por 328 mil imagens com rotulações de 91 categorias de elementos comuns. Imagens com pelo menos uma dimensão menor que 120 pixeis ou com rotulações de "semáforo", "bicicleta", "carro", "ônibus", "moto", "caminhão" e "placa de pare" foram rejeitadas, de modo a evitar contexto de trânsito e imagens muito pequenas entre os planos de fundo. Das imagens restantes, 30 mil foram aleatoriamente selecionadas para compor os planos de fundo do *dataset* de treino e outras 7 mil para o *dataset* de validação. Cada imagem foi redimensionada (quando necessário) de modo a possuir pelo menos 640 pixeis de largura e 480 de altura mantendo a proporção original e, então, teve seus $640 \times 480$ pixeis centrais extraídos e tomados como o plano de fundo final.

#### 4.1.1.2 *Berkeley DeepDrive* (BDD100K)

O *dataset* BDD100K (YU et al., 2018) é composto por 100 mil vídeos de 40 segundos cada referentes a mais de 50 mil percursos realizados no trânsito de diferentes cidades dos Estados Unidos. Para cada vídeo, os elementos de trânsito do quadro referente ao décimo segundo de vídeo são rotulados, resultando em um *dataset* de 100 mil imagens de $1280 \times 720$ pixeis.

Imagens que retratem períodos do dia não correspondentes ao período diurno ou ao nascer ou pôr-do-sol foram desconsideradas. As 47.341 imagens restantes foi redimensionada, sem prejuízo a suas proporções, de modo a possuir 960 pixeis de altura, e então tiveram todos os pixeis que excedem os $1280 \times 960$ pixeis centrais removidos. O conjunto de imagens resultante foi dividido em um subconjunto de 23.850 imagens positivas (com pelo menos um semáforo rotulado) e outro de 23.941 imagens negativas (sem qualquer semáforo

[10] Figuras 15, 16 e 17: COCO; Licença CC-BY 3.0 (<https://creativecommons.org/licenses/by/3.0/>); Fonte original: <http://farm8.staticflickr.com/7219/7405154504_810b2af29b_z.jpg>;
Figuras 18 e 19: BDD100K; Licença BSD 3-Clause (<https://github.com/ucbdrive/bdd100k/blob/master/LICENSE>);
Figura 20: DTLD; Licença para uso e divulgação em publicações científicas.

rotulado). 7 mil imagens de cada subconjunto foram selecionadas aleatoriamente para a criação de um subconjunto positivo e um negativo de planos de fundo para validação.

### 4.1.2 *Datasets* de treino e validação

As próximas subseções descrevem os *datasets* de treino e validação construídos para a execução dos experimentos[10]. Os *datasets* denominados como Contextualizado, Descontextualizado e Modelos-2D combinam os planos de fundo não relacionados à trânsito do *dataset* COCO com imagens de primeiro plano sintéticas. Os *datasets* Planos de Fundo Positivos e Planos de Fundo Negativos empregam planos de fundo que retratam cenas de trânsito e imagens de primeiro sintéticas. Já o *dataset* Referência Real tem como base o *dataset* público *DriveU Traffic Light Dataset* (DTLD) composto por imagens de trânsito do mundo real com rotulações de semáforos, e serve como um comparativo de desempenho para com o método proposto.

#### 4.1.2.1 Contextualizado (Proposto)

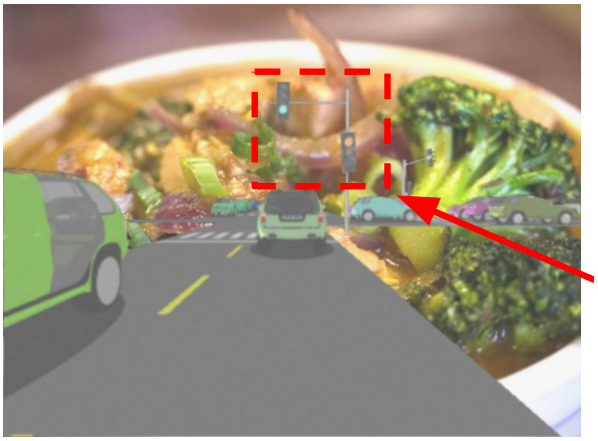


Figura 15 – Imagem do *dataset* Contextualizado.

O *dataset* Contextualizado é também referido como "Proposto", uma vez que se provém do método proposto. Para o conjunto de treino, um total de 20 mil imagens de primeiro plano contendo contexto de trânsito sintético, com resolução de $640 \times 480$ pixeis, foi gerado conforme descrito na Seção 3.2. Então, 70 mil combinações aleatórias entre as 20 mil imagens de primeiro plano e os 30 mil planos de fundo do *dataset* COCO foram geradas conforme os procedimentos explicados na Seção 3.3.

O conjunto de validação foi gerado de forma análoga, porém com combinações entre os 7 mil planos de fundo reservados para validação e 7 mil imagens de primeiro plano à parte das usadas no conjunto de treino. Cada plano de fundo e cada imagem de primeiro plano no conjunto de validação ocorrem estritamente uma vez.

#### 4.1.2.2 Descontextualizado

Este *dataset* é análogo ao Contextualizado, descrevendo exatamente as mesmas combinações de planos, com a diferença de que as imagens de primeiro plano exibem apenas os semáforos da cena de trânsito correspondente, e não todo o contexto de tráfego em si.

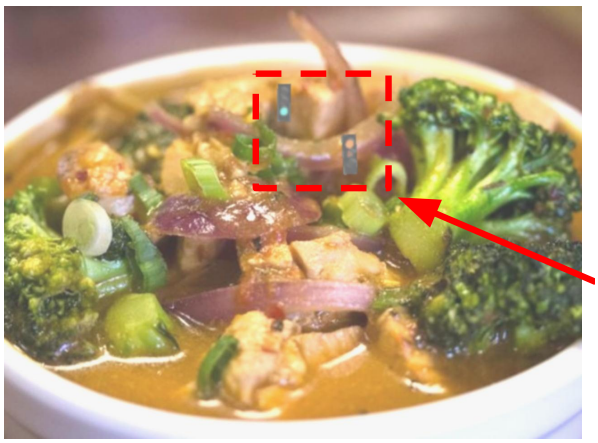


Figura 16 – Imagem do *dataset* Descontextualizado equivalente à Figura 15.

#### 4.1.2.3 Modelos-2D

Figura 17 – Imagem do *dataset* Modelos-2D equivalente à Figura 16.

Este *dataset* é análogo ao Descontextualizado, porém substitui os semáforos tridimensionais das cenas por modelos 2D, como (TORRES et al., 2019) procedeu com placas de trânsito. Esses modelos foram gerados de forma análoga aos tridimensionais em termos de dimensões, cores e tipos, porém substituindo prismas retangulares por retângulos, esferas por círculos, etc.

Cada modelo 2D sofre as seguintes transformações:

- Perspectiva: as laterais horizontal e vertical que estarão mais distantes do centro da imagem final são menores que suas laterais opostas;
- Rotação: o modelo é rotacionado de um ângulo aleatoriamente escolhido dentro do intervalo $\pm 3,6°$;
- Redimensionamento: o resultado das operações anteriores é redimensionado para caber na caixa delimitadora correspondente do *dataset* Descontextualizado.

Por fim, cada modelo 2D é posicionado sobre uma imagem transparente que é sobreposta ao plano de fundo, de modo que a disposição dos semáforos na imagem final

corresponda à de seus equivalentes no conjunto Descontextualizado.

#### 4.1.2.4 Planos de Fundo Positivos

Este *dataset* substitui os planos de fundo do Descontextualizado pelas imagens do subconjunto positivo do BDD100K. Dessa forma, obtém-se um *dataset* com semáforos sintéticos sobre planos de fundo contendo cenas de trânsito reais que incluem semáforos reais. Como estes fazem parte do plano de fundo, apenas os semáforos sintéticos são rotulados.

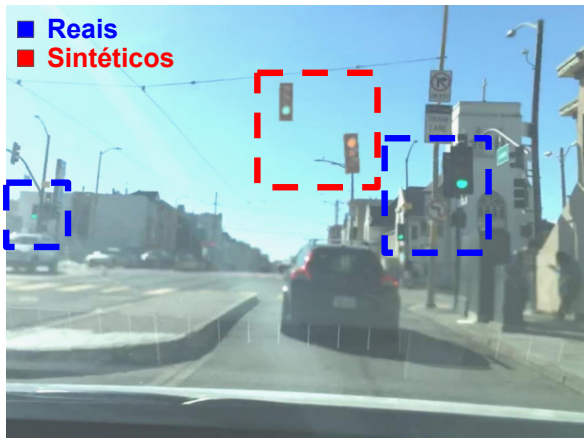



Figura 18 – Imagem do *dataset* Planos de Fundo Positivos equivalente à Figura 16.

#### 4.1.2.5 Planos de Fundo Negativos

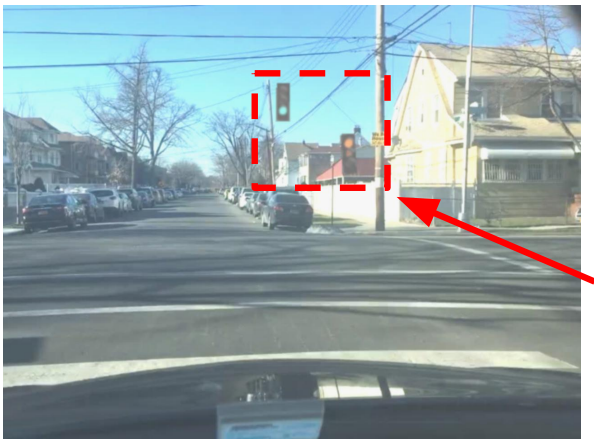


Figura 19 – Imagem do *dataset* Planos de Fundo Negativos equivalente à Figura 16.

Como o nome sugere, este *dataset* é análogo ao Planos de Fundo Positivos, porém substituindo os planos de fundo positivos pelos negativos. Assim, semáforos sintéticos rotulados sobrepõem cenas de trânsito do mundo real que não apresentem semáforos originalmente.

| Característica | Semáforos escolhidos |
|---|---|
| Orientação da face | Vista frontal |
| Relevância[12] e oclusão | Todos os níveis |
| Orientação axial | Vertical |
| Quantidade de bulbos | 3 |
| Estado | Vermelho, verde e amarelo |
| Pictograma[13] | Bulbo completamente iluminado ou com seta |

Tabela 1 – Perfis de semáforo disponíveis no *dataset* DTLD escolhidos segundo suas características.

#### 4.1.2.6 Referência Real: *DriveU Traffic Light Dataset* (DTLD)

O *dataset* DTLD[11] (FREGIN et al., 2018) contém mais de 43 mil imagens apresentando cenas reais de tráfego em 11 cidades da Alemanha, contando com mais de 230 mil semáforos rotulados. As imagens, de resolução original de $2048 \times 1024$ pixeis, tiveram uma área de $1280 \times 960$ pixeis, superior na vertical e centralizada na horizontal, extraída e redimensionada para $640 \times 480$ pixeis sem prejudicar suas proporções. Então, as imagens foram filtradas de acordo com características dos semáforos, presentes nas rotulações, que eram de interesse para o treino do modelo. A Tabela 1 descreve as características rotuladas e os critérios de escolha dos semáforos com base nelas.

Figura 20 – Imagem do *dataset* DTLD.

Após a filtragem, restaram ao *dataset* um total de 33.374 imagens e aproximadamente 26 mil, 48 mil e 2.500 semáforos em estado vermelho, verde e amarelo, respectivamente. Visto que o conjunto resultante é altamente desbalanceado e que os demais *datasets* foram montados com um total de 70 mil imagens, o seguinte processo foi aplicado:

- Os seguintes três subconjuntos de imagens (com um nível de interseção entre os dois

[11] <https://www.uni-ulm.de/en/in/driveu/projects/driveu-traffic-light-dataset/>
[12] Influência do semáforo sobre a via em que a primeira pessoa do *dataset* está dirigindo.
[13] "Imagem" impressa no bulbo: seta, pedestre, bulbo completamente iluminado, etc.

últimos) foram considerados:

  - imagens com pelo menos 1 semáforo amarelo;
  - imagens com pelo menos 1 semáforo verde e nenhum amarelo;
  - imagens com pelo menos 1 semáforo vermelho e nenhum amarelo.

- um novo conjunto de imagens foi proposto;
- uma imagem de cada subconjunto era alternadamente atribuída ao novo conjunto;
- o processo foi interrompido ao se atingir um total de 70 mil imagens dentro do novo conjunto.

Este processo resultou no *dataset* final de treino, contendo 70 mil imagens e aproximadamente 56 mil, 59 mil e 46 mil semáforos vermelho, verde e amarelo, respectivamente. O *dataset* resultante foi submetido aos mesmos procedimentos de aumento aplicados aos demais *datasets*.

Excepcionalmente, este *dataset* não possui um conjunto de validação específico. Em vez disso, o modelo Referência Real foi validado sobre os conjuntos de validação-de-caixa explicados na Seção 4.1.3. Cada um destes conjuntos corresponde a uma porção de um *dataset* de teste, de modo que validar o modelo sobre eles enviesa o modelo para ser mais eficiente sobre o conjunto de teste correspondente. Assim, os demais *datasets* são comparados com o melhor caso do Referência Real para cada conjunto de teste.

### 4.1.3 *Datasets* de teste e validação-de-caixa

Os *datasets* utilizados para teste descrevem diferentes conjuntos de imagens de cenas reais de tráfego, de modo que possa ser avaliada a aplicação dos modelos no mundo real[14]. Cada um deles teve 10% de suas imagens positivas e 10% das negativas extraídas para a construção do respectivo conjunto de validação-de-caixa.

**Validação-de-caixa:** alguns dos *datasets* de teste apresentam caixas delimitadoras pouco acuradas, isto é, de área consideravelmente menor ou maior que a área do semáforo que ela compreende, como exemplificado na Figura 21[15]. Isso pode levar a erros na interpretação dos resultados de desempenho dos modelos, visto que um semáforo que teve sua caixa delimitadora de detecção muito precisa pode apresentar uma interseção com a caixa

[14] Figura 23: LISA; Licença CC-BY-NC-SA 4.0 (<https://creativecommons.org/licenses/by-nc-sa/4.0/>);
Figura 24: Udacity; Licença MIT (<https://github.com/udacity/self-driving-car/blob/master/datasets/LICENSE.md>);
Figura 25: LaRA; Licença livre para pesquisa.

Figura 21 – Exemplo de caixas delimitadoras rotuladas de dimensões muito diferentes das dos semáforos.

rotulada insuficiente para que se considere uma detecção correta como deveria. A ideia por trás dos conjuntos de validação-de-caixa é encontrar uma proporção de suas caixas delimitadoras originais que leve a melhores resultados, de modo que essa proporção seja aplicada ao conjunto de teste correspondente durante o teste. Note que esse procedimento visa tão somente uma avaliação mais justa dos modelos e não a obtenção de um parâmetro ótimo, o que justifica este conjunto ser derivado do conjunto de teste e não do de treino.

Os *datasets* usados para validação-de-caixa e teste foram o *Intelligent Autonomous Robotic Automobile* (IARA), um *dataset* interno, e três *datasets* publicamente disponíveis: *Laboratory for Intelligent & Safe Automobiles* (LISA), *Udacity Driving Dataset* e *La Route Automatisée* (LaRA). Estes conjuntos de dados foram adaptados para a derivação de *datasets* customizados.

[15] Imagem adaptada do *dataset* LISA (JENSEN et al., 2016; PHILIPSEN et al., 2015), adaptável, divulgável e redistribuída sob a licença <https://creativecommons.org/licenses/by-nc-sa/4.0/>.

#### 4.1.3.1 *Intelligent Autonomous Robotic Automobile* (IARA)

Figura 22 – Imagem do *dataset* IARA.

Este *dataset* apresenta imagens do tráfego na cidade de Vitória - ES, gravadas com o auxílio da câmera presente na plataforma IARA, veículo autônomo desenvolvido pelo Laboratório de Computação de Alto Desempenho (LCAD) da Universidade Federal do Espírito Santo (UFES) (POSSATTI et al., 2019; BADUE et al., 2020). O *dataset* possui 3.997 imagens de resolução $1280 \times 960$ pixeis, das quais 2.695 são positivas, e um total de 5.002 rotulações de semáforos, distribuídas como 1.813 vermelhos, 2.923 verdes e 266 amarelos.

#### 4.1.3.2 LISA_treino+teste e LISA_teste: *Laboratory for Intelligent & Safe Automobiles* (LISA)

O *dataset* LISA (JENSEN et al., 2016; PHILIPSEN et al., 2015) apresenta cenas de cenas de trânsito das vias de San Diego - Califórnia, EUA, registradas em resolução de $1280 \times 960$ pixeis nos períodos tanto diurno quando noturno. O *dataset* é originalmente dividido em um conjunto de treino composto por 13 seções diurnas e cinco noturnas e um conjunto de teste dividido em duas sequências diurnas e 2 noturnas.

Figura 23 – Imagem do *dataset* LISA.

Neste trabalho, apenas as imagens correspondentes ao período diurno foram utilizadas, com suas dimensões reduzidas à metade. Semáforos originalmente rotulados como *stop* ou *stopLeft* foram considerados vermelhos, como *go* ou *goLeft* verdes e como *warning* ou *warningLeft* amarelos. O conjunto de treino resultante possui 14.034 imagens, sendo 12.775 positivas, e aproximadamente 22 mil semáforos vermelhos, 15 mil verdes e 1.000 amarelos. Já o conjunto de teste possui 10.954 imagens, sendo 7.473 positivas, e aproximadamente

10 mil, 8 mil e 450 semáforos vermelhos, verdes e amarelos, respectivamente.

Dois *datasets* foram derivados do *dataset* LISA para este trabalho:

- LISA_treino+teste: corresponde a todas as imagens consideradas, tanto as do conjunto original de treino quanto do de teste;
- LISA_teste: corresponde somente às imagens originalmente pertencentes ao conjunto de teste.

#### 4.1.3.3 Udacity- e Udacity+: *Udacity Driving Dataset*

Figura 24 – Imagem do *dataset* Udacity.

Em seu repositório público[16], a Udacity[17] oferece dois *datasets* apresentando cenas do trânsito na região de Mountain View - Californa, EUA, com diversos elementos de trânsito rotulados. No entanto, apenas o *Dataset* 2 apresenta rotulação de semáforo, sendo, por isso, o único utilizado neste trabalho. Este conjunto é composto por 15 mil imagens de 1920 × 1200 pixeis de resolução coletadas em período diurno.

As rotulações correspondentes a semáforos oclusos foram descartadas, resultando em um total de mais de 14 mil semáforos rotulados distribuídos entre as 4.474 imagens positivas do conjunto. No entanto, cerca de 3 mil semáforos foram rotulados múltiplas vezes, o que motivou a derivação de dois *datasets* a partir do *dataset* original:

- Udacity-: corresponde ao *dataset* original, porém considerando apenas a caixa delimitadora de menor área de cada conjunto de caixas sobrepostas;
- Udacity+: análogo ao Udacity-, porém considerando a caixa delimitadora de maior área.

No total, cada um dos *datasets* derivados apresenta aproximadamente 6.800 rotulações de semáforos vermelhos, 4.200 de semáforos verdes e 200 de semáforos amarelos.

[16] <https://github.com/udacity/self-driving-car/tree/master/annotations>
[17] <https://www.udacity.com/>

#### 4.1.3.4 *La Route Automatisée* (LaRA)

O *dataset* LaRA (CHARETTE, 2013) é composto por quadros de um vídeo gravado dentro do trânsito da cidade de Paris, França. Neste trabalho, ele foi utilizado em sua forma original: 11.179 imagens, sendo 5.932 positivas; resolução de 640 × 480 pixeis; 5.280 semáforos vermelhos, originalmente rotulados como *stop*; 3.381 semáforos verdes, originalmente rotulados como *go*; e 58 semáforos amarelos, originalmente rotulados como *warning*.

Figura 25 – Imagem do *dataset* LaRA.

## 4.2 Configuração experimental

O projeto, assim como o trabalho de (TORRES et al., 2019), teve como base uma implementação em *Tensorflow* (ABADI et al., 2016)[18] da consolidada arquitetura *Faster R-CNN* (REN et al., 2015) para detecção profunda de objetos, suportada pela também consolidada arquitetura *ResNet-101* (HE et al., 2016) como extrator de características. Maior conhecimento sobre essas arquiteturas pode ser conferido nas Seções 2.2.2 e 2.3.2.

Com exceção de um único caso, explanado mais à frente, os modelos foram treinados por 70 mil iterações. Os principais ajustes empiricamente feitos à configuração original da implementação são descritos na Tabela 2.

## 4.3 Métricas e procedimentos de avaliação

Para a avaliação do desempenho dos modelos, foram adotadas as métricas *mean Average Precision* (mAP) e *F1-score*. Foram consideradas como corretas as caixas de detecção com um índice de Interseção sobre união (IOU) igual ou superior a 0,5. Secundariamente, uma das análises apresentadas na Seção 5 também é feita sobre as métricas de precisão e revocação. Uma revisão do significado destas métricas pode ser feita na Seção 2.3.1.

O mAP é calculado considerando todas as caixas de detecção obtidas durante o processo de inferência, sendo também empregado na escolha da melhor proporção de caixa delimitadora durante a etapa de validação-de-caixa. Já o *F1-score* é calculado, durante a

[18] <https://github.com/endernewton/tf-faster-rcnn>

| **Parâmetro** | **Valor** | **Descrição** |
|---|---|---|
| Escala das âncoras | $\{2; 4; 8; 16; 32\}$ | Multiplicadores da área de referência das âncoras |
| Sobreposição mínima de RoI | 0 | RoI é considerada plano de fundo se nenhuma interseção com caixas delimitadoras rotuladas |
| Tamanho do lote | 1 | 1 imagem por iteração |
| Taxa de aprendizado | $10^{-3}$ / $10^{-4}$ | $10^{-3}$ para as primeiras 50 mil iterações, depois reduzida para 10% de seu valor original. |
| Escalas de imagem: treino | $\{480; 960\}$ | Reescala aleatória da menor dimensão. O tamanho original ($1280 \times 960$) é mantido ou reduzido à metade. |
| Escalas de imagem: teste | $\{960\}$ | Menor dimensão das imagens de teste são redimensionadas para este valor, preservando a proporção. |

Tabela 2 – Ajustes feitos à configuração original da implementação

etapa de validação, considerando caixas de detecção com limiares de confiança a partir de diferentes valores, na busca de um limiar ótimo a ser adotado para os testes.

### 4.3.1 Validação

O processo de validação visa a obtenção de um limiar de confiança que maximize o *F1-score* do modelo sobre o conjunto de validação. Posteriormente, esse limiar ótimo é utilizado para teste do modelo sobre as imagens de trânsito do mundo real.

- **Procedimento:** Para um limiar de confiança $c$ de 0,01 a 1,0, em passos de 0,01:
  - filtram-se as caixas de detecção com confiança igual ou superior a $c$;
  - as métricas são computadas considerando as caixas delimitadoras filtradas;
  - se o *F1-score* resultante supera o obtido no passo anterior, o valor de $c$ é guardado.
- **Saída:** limiar de confiança $c$ que gera o maior *F1-score*.

### 4.3.2 Validação-de-caixa

A validação-de-caixa ocorre de forma similar à validação comum, porém executada diretamente sobre porções dos *datasets* de teste, gerando variações das caixas delimitadoras rotuladas que mantém seu centro e proporções originais enquanto comprime/expande suas áreas. A variação que gera o maior mAP é empregada durante o teste do modelo sobre aquele mesmo *dataset*, permitindo uma avaliação mais justa dos modelos treinados com caixas delimitadoras ajustadas ao semáforos sobre *datasets* com caixas rotuladas menos fiéis às áreas dos semáforos.

- **Procedimento:** Para um fator $f$ de 0,4 a 1,9, em passos de 0,1:
  - Para um limiar de confiança $c$ de 0,01 a 1,0, em passos de 0,01:
    - multiplicam-se as áreas de todas as caixas delimitadoras por $f$, mantendo seu centro e sua proporção altura/largura;
    - filtram-se as caixas de detecção com confiança igual ou superior ao limiar $c$;
    - as métricas são computadas considerando as caixas delimitadoras filtradas e modificadas;
    - o mAP obtido é guardado.
- **Saída:** fator $f$ que leve ao maior mAP. Em caso de empate, o valor de $f$ mais próximo de 1,0 é considerado.

### 4.3.3 Teste

Em posse do limiar de confiança ótimo e dos fatores multiplicativos mais apropriados para cada *dataset* de teste, o desempenho do modelo pode ser devidamente testado sobre eles, aplicando-se esses parâmetros.

- **Procedimento:**
  - multiplicam-se as áreas de todas as caixas delimitadoras pelo fator $f$ obtido na etapa de validação-de-caixa, mantendo seu centro e sua proporção altura/largura;
  - filtram-se as caixas de detecção com confiança igual ou superior ao limiar $c$ obtido na etapa de validação;
  - as métricas são computadas considerando as caixas delimitadoras filtradas e modificadas;
  - os valores obtidos para as métricas de avaliação são guardados.
- **Saída:** resultados numéricos para as métricas de avaliação.

## 4.4 Experimentos e análises

Esta seção descreve os experimentos realizados e as análises realizadas sobre eles. Isso inclui o treinamento, validação, validação-de-caixa e teste de modelos usando cada um dos *datasets* descritos na Seção 4.1.2 e de combinações entre *datasets* sintéticos e do mundo real. Objetiva-se comparar diferentes grupos de modelos a fim de se estudar seu desempenho em relação a características específicas dos *datasets*, como a presença de contexto de trânsito, o uso de modelos de semáforos mais ou menos realistas, o domínio contextual do plano de fundo, e a natureza sintética, real ou híbrida do *dataset*.

Por simplicidade, ao decorrer do texto será utilizada a terminologia *modelo* $X$, em que $X$ é o nome de um *dataset*, para denotar um *modelo treinado utilizando o* dataset $X$.

### 4.4.1 Análise do impacto do contexto

Esta análise diz respeito à comparação de desempenho dos modelos Contextualizado e Descontextualizado. Em outras palavras, pretende-se avaliar as diferenças de eficácia no uso de *datasets* idênticos em termos de planos de fundo e distribuição de semáforos sintéticos, porém apresentando ou não contexto de trânsito no primeiro plano.

### 4.4.2 Modelos 2D *versus* modelos 3D

Comparação de desempenho entre os modelos Descontextualizado e Modelos-2D. Visa o estudo do impacto de se utilizar modelos 3D mais realistas de semáforo em lugar de modelos 2D simples sobrepondo o plano de fundo.

### 4.4.3 Análise do domínio dos planos de fundo

Visa avaliar se empregar planos de fundo relacionados à trânsito em vez de planos de fundo de outros domínios de fato prejudica o modelo. Foram comparados os desempenhos do modelo Descontextualizado com o do (i) Planos de Fundo Negativos, de modo a verificar se o domínio de trânsito ao fundo tem algum impacto sobre o desempenho em caso de não apresentar semáforos, e o do (ii) Planos de Fundo Positivos, para analisar se semáforos reais tomados como plano de fundo prejudicam o aprendizado sobre os semáforos sintéticos que compõem o primeiro plano.

### 4.4.4 Uso do método proposto como aumento de dados

A ideia deste estudo é analisar se o *dataset* proposto pode ser utilizado para melhorar o desempenho de um modelo treinado com dados reais. Três análises foram realizadas:

- **Contextualizado + Referência Real - 70 mil iterações**: os *datasets* Contextualizado e Referência Real foram misturados, formando um único *dataset* com 140 mil imagens. Apesar de ter o dobro do tamanho, este *dataset* foi treinado com 70 mil iterações (1 por imagem, ignorando um grupo aleatório de 70 mil imagens), quantidade usada para treinar cada um dos *dataset* que compõem este híbrido. O modelo resultante será referido como **Contexto+Real70**;
- **Contextualizado + Referência Real - 140 mil iterações**: modelo resultante de uma iteração sobre cada uma das 140 mil imagens do *dataset* híbrido descrito no tópico anterior. Este modelo é denotado como **Contexto+Real140**;
- **Ajuste fino - real sobre sintético**: o modelo Contextualizado passou por um ajuste fino de mais 70 mil iterações sobre o *dataset* Referência Real. O modelo final será denotado como **Contexto->Real**.

A validação de cada um destes modelos híbridos foi realizada sobre o conjunto de validação Contextualizado.

### 4.4.5 Dados reais *versus* sintéticos

Corresponde ao estudo da diferença de desempenho entre o método proposto e um modelo treinado com dados do mundo real, isto é, entre os modelos Contextualizado e Referência Real.

Vale lembrar o que foi introduzido na Seção 4.1.2.6: a validação do modelo Referência Real, excepcionalmente, não é feita sobre um conjunto próprio de validação, e sim sobre cada um dos conjuntos de validação-de-caixa. Como estes são diretamente provenientes dos conjuntos de teste, isso implica numa tendência em encontrar o melhor limiar de confiança possível para cada conjunto de teste, de modo a enviesar para melhor os resultados obtidos com o modelo Referência Real. Dessa forma, o desempenho do método proposto pode ser comparado com o melhor caso de seu concorrente.

## 4.5 Recursos computacionais

Tanto os processos de treino quanto os de inferência foram executados em uma máquina contendo CPU Intel® Core™ i7-4770 (3,40 GHz), 16 GB de memória RAM e uma GPU NVIDIA® TITAN Xp™ com 12 GB de memória. O treino dura aproximadamente 0,35 segundos por iteração e a inferência em uma imagem dura cerca de 0,13 segundos.

# 5 Resultados e discussão

A Figura 26 mostra os resultados de mAP e *F1-score* de teste obtidos para cada um dos *datasets* customizados de teste.

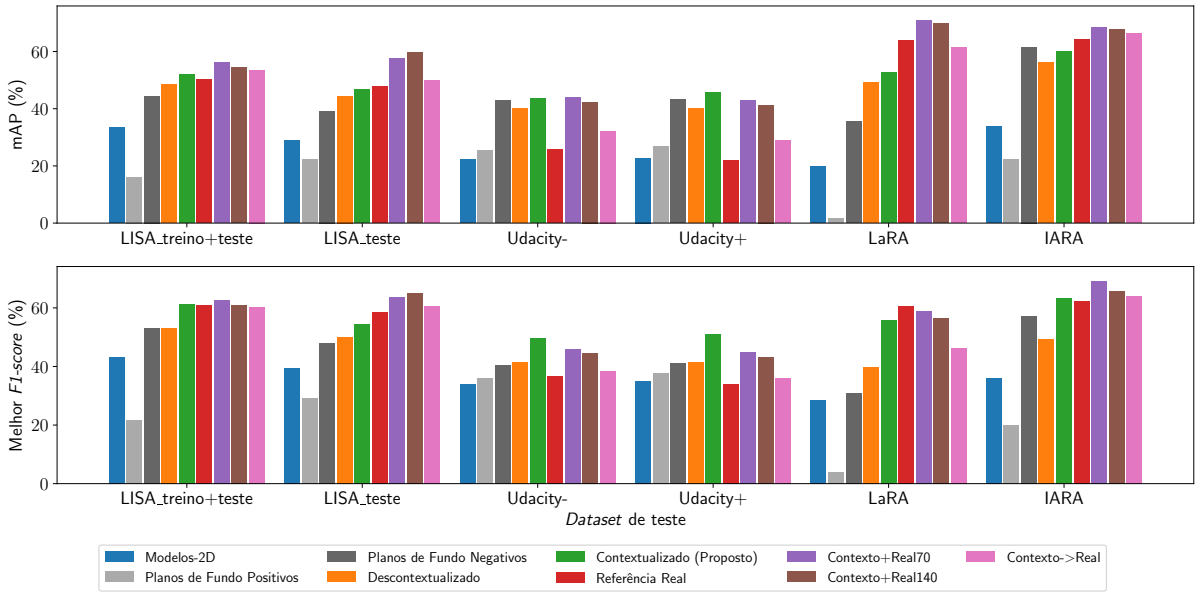



Figura 26 – Resultados de mAP e *F1-score* para cada *dataset* de teste.

As subseções a seguir apresentam os resultados e interpretações obtidas para cada um dos experimentos e análises apresentados na Seção 4.4. Cada subseção apresentará uma variação da Figura 26 que destaca apenas os resultados de interesse, para facilitar a visualização.

## 5.1 Análise do impacto do contexto

Esta subseção compara os resultados obtidos para os modelos Contextualizado, em verde na Figura 27, e Descontextualizado, em laranja.

Como se pode observar, o modelo Contextualizado supera o Descontextualizado para absolutamente todos os *datasets* de teste e para ambas as métricas. O ganho em mAP com a presença de contexto varia de 2,22 (LISA_teste, 44,46 a 46,68%) a 5,58 p.p. (Udacity+, 40,16 a 45,74%), enquanto o *F1-score* aumenta de 4,36 (LISA_teste, 49.95 a 54.31%) a significantes 16,18 p.p. (LaRA, 39,60 a 55,78%).

Os resultados de precisão e revocação propriamente ditos sugerem que os aumentos em mAP e *F1-score* devem estar associados a um aumento de precisão, enquanto a revocação se conserva. A precisão sofre um aumento considerável, de 10,55 (Udacity-,

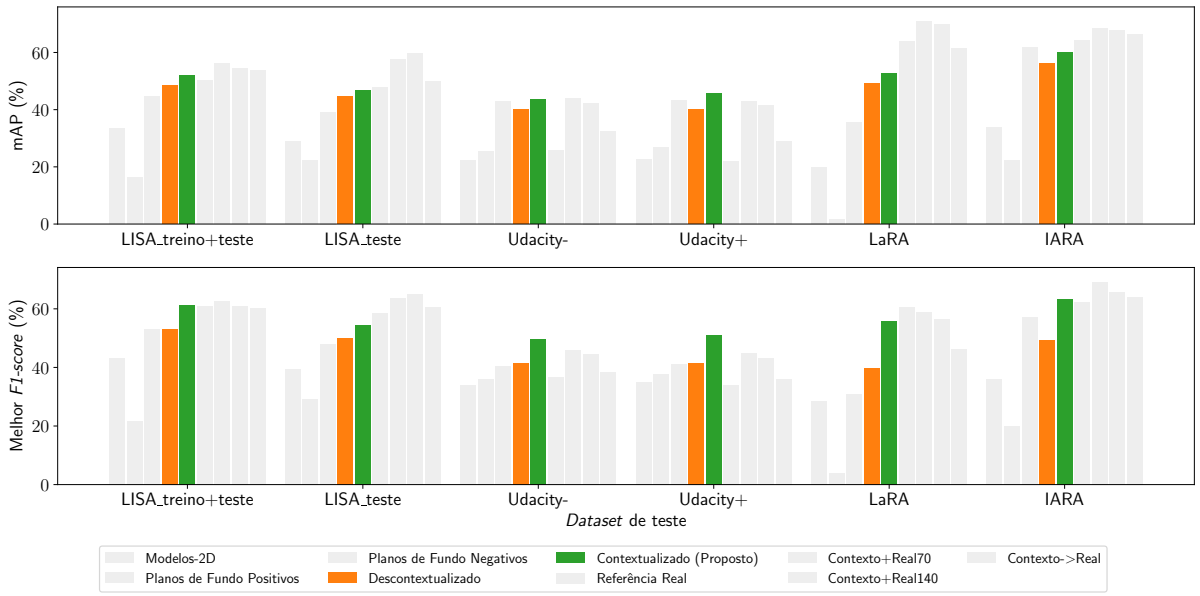



Figura 27 – Resultados: Contextualizado x Descontextualizado.

31,59 a 42,14%) a 25,48 p.p. (LaRA, 30,50 a 55,98%), enquanto a variação na revocação se limita a -1,89 (LISA_treino+teste, 64,27 a 62,38%) a 2,37 (Udacity+, 60,98 a 63,35%), não incluso o resultado para o *dataset* LISA_teste. Este, excepcionalmente, apresentou uma variação considerável na revocação (-9,29 p.p.) que é, no entanto, compensada por um aumento ainda mais significativo na precisão (+15,74 p.p.).

Em geral, estes resultados sugerem que a presença de contexto diminui a quantidade de falsos positivos enquanto exerce pouca influência sobre a quantidade de falsos negativos. Isso significa que se torna mais incomum a ocorrência de detecção de elementos quaisquer da cena como semáforos, limitando as detecções a semáforos de fato, ao mesmo tempo que não prejudica a detecção dos semáforos que o modelo treinado sem contexto já seria capaz de detectar.

## 5.2 Modelos 2D *versus* modelos 3D

Esta subseção compara os resultados obtidos para os modelos Descontextualizado, em laranja na Figura 28, e Modelos-2D, em azul.

É notável que o modelo Descontextualizado supera consideravelmente seu concorrente sem exceção de *dataset* ou métrica. Em mAP, o modelo Descontextualizado produz de 15,01 (LISA_treino+teste) a 29,61 p.p. (LaRA) a mais que o Modelos-2D, enquanto em *F1-score* a evolução compreende um intervalo de 6,74 (Udacity+) a 13,26 p.p. (IARA).

Esta análise esclarece que o emprego de modelos 3D mais realistas supera o de modelos 2D simples para detecção. Este resultado era esperado, dado que características realistas dificilmente alcançáveis pelos modelos 2D, como noção de profundidade em

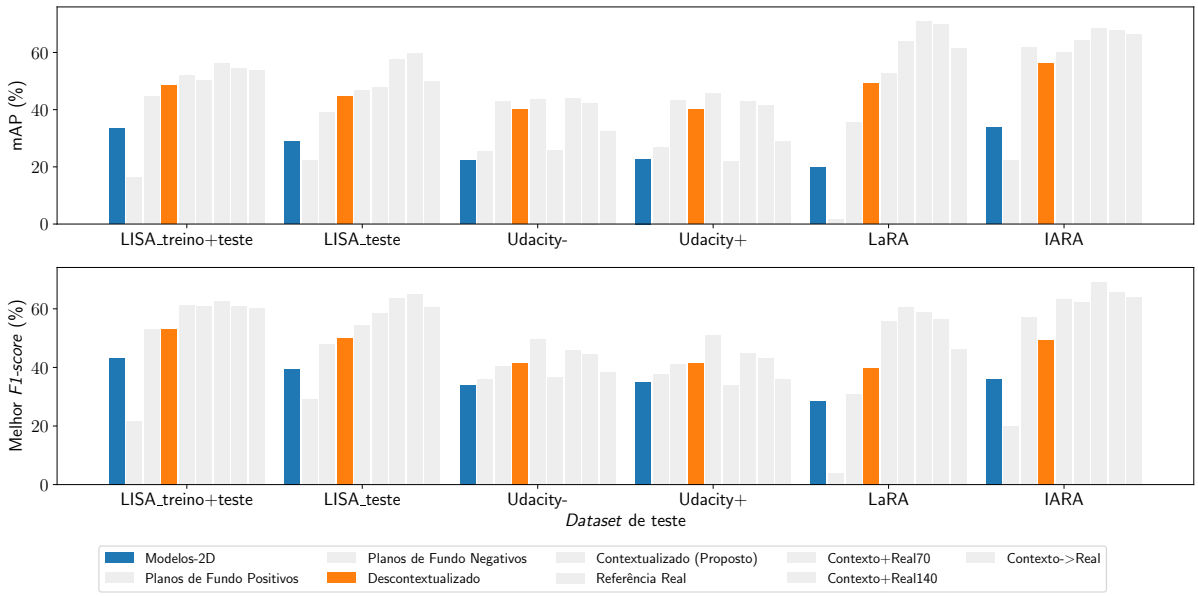



Figura 28 – Resultados: Modelos-2D x Descontextualizado.

perspectiva e efeitos da iluminação externa, aproximam os modelos 3D dos semáforos reais exibidos pelos conjuntos de teste.

## 5.3 Análise do domínio dos planos de fundo

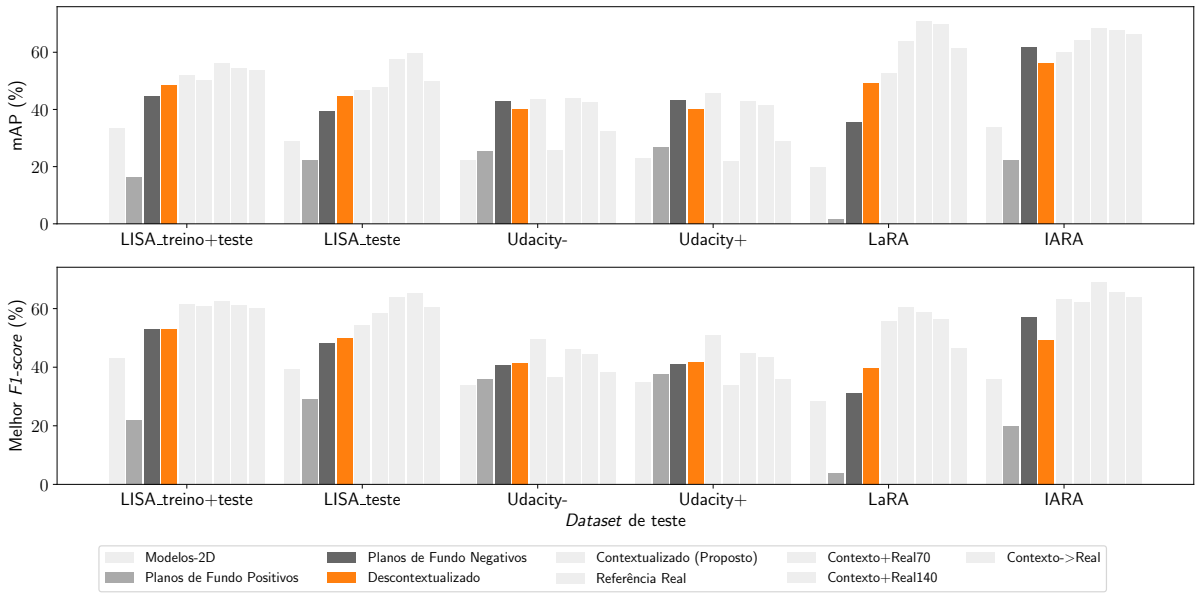



Figura 29 – Resultados: Planos de Fundo Positivos x Planos de Fundo Negativos x Descontextualizado.

Esta subseção compara os resultados obtidos para os modelos Descontextualizado, em laranja na Figura 29, Planos de Fundo Negativos, em cinza escuro, e Planos de Fundo Positivos, em cinza mediano.

Para os *datasets* LISA_teste, LISA_treino+teste, Udacity- e Udacity+, a diferença de desempenho do modelo Planos de Fundo Negativos para o Descontextualizado é desprezível em magnitude (2,73 (Udacity-) a 5,27 p.p. (LISA_teste) em mAP; 0,05 (LISA_treino+teste) a 1,82 p.p. (LISA_teste) em *F1-score*). Para os dois *datasets* restantes, a diferença é um pouco mais acentuada, porém em direções opostas: para o *dataset* LaRA, o modelo Descontextualizado é superior em 13,69 p.p. em mAP e 8,56 p.p. em *F1-score*, enquanto para o *dataset* IARA é o modelo Planos de Fundo Negativos que apresenta desempenho superior, com vantagem de 5,46 p.p. de mAP e 7,90 p.p. de *F1-score*.

Esses resultados demonstram que utilizar planos de fundo relacionados a trânsito mas sem a presença de semáforos tende a ser tão efetivo quanto utilizar planos de fundo de outros domínios (como o método proposto), podendo apresentar apenas diferenças imprevisíveis e pouco acentuadas de desempenho.

Já quando se compara o modelo Planos de Fundo Positivos com o modelo Descontextualizado (e com sua contraparte negativa, que já se mostrou comparável ao Descontextualizado), observa-se que a presença de semáforos como parte do plano de fundo deteriora os resultados. Os números mostram que o Planos de Fundo Positivos é inferior ao Descontextualizado em uma faixa de 13,42 (Udacity+) a significantes 47,73 p.p. (LaRA) em mAP, considerando todos os *datasets*, e em uma faixa de 20,78 (LISA_teste) a 35,63 p.p. (LaRA) em *F1-score* desconsiderando os *datasets* Udacity+ e Udacity-, casos excepcionais em que a diferença foi pequena (e ainda assim favorável ao Descontextualizado).

Essa análise sugere que a presença do objeto que se pretende detectar como parte do plano de fundo prejudica o aprendizado da rede, dada a dificuldade em se discernir quando um mesmo objeto (semáforo, neste caso) deve ser detectado e quando deve ser tomado como parte do plano de fundo.

Em conclusão, o desempenho do modelo não é impactado pelo domínio do plano de fundo, mas sim o fato de este apresentar ou não amostras do objeto de detecção.

## 5.4 Uso do método proposto como aumento de dados

Esta subseção compara entre si os resultados obtidos para os modelos híbridos Contexto+Real70 (em roxo na Figura 30), Contexto+Real140 (em marrom) e Contexto->Real (em rosa). Além disso, o desempenho destes modelos também é comparado ao do modelo proposto (Contextualizado, em verde) e ao da referência do mundo real (Referência Real, em vermelho).

Entre os modelos Contexto+Real70 e Contexto+Real140, nota-se que dobrar o número de iterações não surtiu efeito significativo sobre o desempenho, demonstrando que 70 mil iterações se fazem suficientes. Por outro lado, observa-se que o ajuste fino é muito

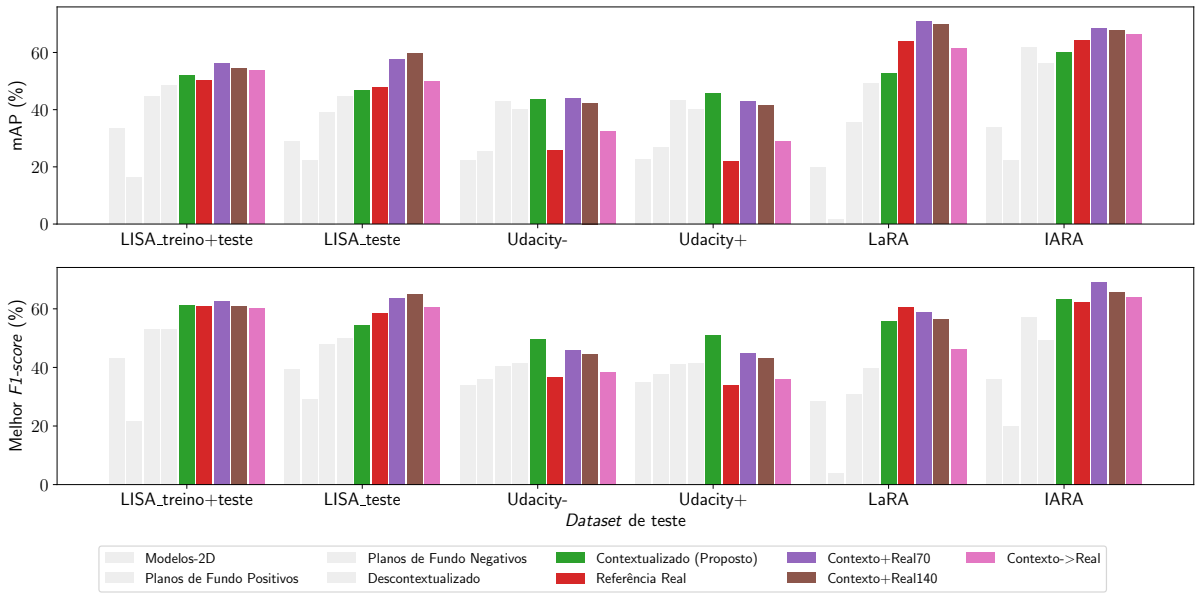



Figura 30 – Resultados: Contextualizado x Referência Real x Contexto+Real70 x Contexto+Real140 x Contexto->Real.

menos promissor que misturar os dados, sendo superado por ambos os modelos com dados misturados (Contexto+Real70 e Contexto+Real140) em absolutamente todos os casos. Entre os *datasets* LISA_teste, Udacity-, Udacity+ e LaRA, o ajuste fino é inferior em 9,49 a 13,93 p.p. em mAP e em 4,5 a 12,4 p.p. em *F1-score* em relação ao modelo misturado mais efetivo de cada caso.

Ao se comparar o desempenho dos modelos misturados com o do Referência Real, nota-se que houve, em geral, um ganho. O maior ganho ocorre para o *dataset* Udacity+, sendo superior a 21 p.p. em mAP e 10 p.p. em *F1-score*. O único caso em que houve piora foi para o *F1-score* do *dataset* LaRA, e ainda assim a diferença é limitada a 4,3 p.p.. O ajuste fino, por outro lado, se mostra geralmente comparável ao Referência Real. Como o ajuste foi feito utilizando com o próprio *dataset* Referência Real, isso indica que o aprendizado posterior de dados reais se sobressai ao aprendizado anterior de dados sintéticos.

Em geral, os modelos híbridos também se mostraram mais efetivos que o modelo Contextualizado, exceto para os *datasets* Udacity+ e Udacity-. Para estes casos, porém, a diferença de desempenho entre o modelo Contextualizado e o híbrido mais efetivo (Contexto+Real70) é limitada a 4,43 p.p. em mAP e 6,06 p.p. em *F1-score* (Udacity+).

Os resultados discutidos revelam que a utilização do método proposto como aumento de dados é promissora. Porém, é preferível que modelos sejam treinados com dados misturados do que através de ajuste fino.

## 5.5 Dados reais *versus* sintéticos

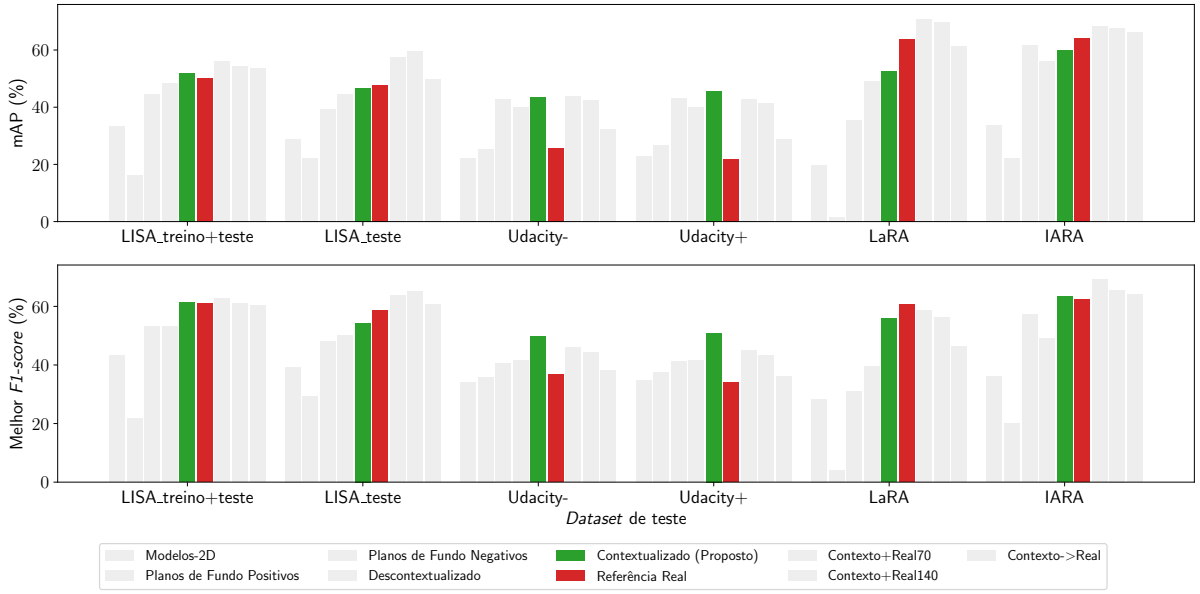



Figura 31 – Resultados: Contextualizado x Referência Real.

Finalmente, esta seção apresenta a importante análise que compara o modelo sintético proposto (Contextualizado, em verde na Figura 31) a um modelo treinado com dados do mundo real (Referência Real, em vermelho).

As diferenças de desempenho entre os modelos Contextualizado e Referência Real parecem variar consideravelmente entre os *datasets* de teste. Para os *datasets* Udacity+, Udacity- e LISA_treino+teste, o modelo Contextualizado superou o Referência Real, com uma diferença que varia entre de 1,70 a 23,93 p.p. de mAP e 0,56 a 16,90 p.p. de *F1-score*. Por outro lado, o modelo Referência Real supera o Contextualizado para os *datasets* LISA_teste e LaRA, com diferenças de desempenho de 1,20 a 11,29 p.p. de mAP e 4,31 a 4,87 p.p. de *F1-score*. Por fim, para o *dataset* IARA as métricas divergem entre si: o modelo Contextualizado é inferior em cerca de 4,20 p.p. de mAP e superior em cerca de 1,06 p.p.

Considerando todos os resultados exibidos na Figura 31, calcula-se que o modelo Contextualizado produza, em média, $50,08\% \pm 5,99\%$ de mAP contra $45,61\% \pm 18,26\%$ do modelo Referência Real. Para o *F1-score*, esse valores são de respectivamente $55,93\% \pm 5,50\%$ contra $52,21\% \pm 13,1\%$. Esses resultados são compatíveis com o estado-da-arte, visto que os trabalhos relacionados recentes (POSSATTI et al., 2019; PON et al., 2018; KIM; PARK; JUNG, 2018) reportam resultados aproximados de mAP entre 38 e 55% em cenários intra-*dataset*, que tendem a apresentar melhor desempenho dada a semelhança entre os dados de treino e os de teste, enquanto os obtidos neste trabalho refletem um estudo naturalmente inter-*dataset* e, consequentemente, mais desafiador. De todos esses fatos, pode-se concluir que:

- o método proposto pode competir com modelos treinados com dados do mundo real sem os esforços de coletas e rotulação;
- tanto o método proposto quanto sua contraparte não-sintética atingem desempenho aceitável quando comparados ao estado-da-arte;
- dada a relativa dificuldade de comparação do método proposto com outros cenários inter-*dataset* por serem menos passíveis de serem encontrados na literatura, o modelo Referência Real se apresenta como uma alternativa aceitável de comparação.

A Figura 32 mostra um exemplo visual do desempenho do método proposto sobre uma imagem do *dataset Udacity*. As caixas vermelho-claras representam as caixas de rotulação originais da imagem, que notavelmente não são muito bem ajustadas às áreas reais dos semáforos. Por sua vez, as caixas ciano representam as predições obtidas pelo modelo que, ao contrário das caixas de rotulação, se ajustam muito precisamente aos semáforos, como desejável. Por fim, as caixas vermelho-escuras representam as predições consideradas para fins de avaliação após aplicação do fator multiplicativo obtido com o conjunto de validação-de-caixa, aproximando as predições originais das rotulações e, consequentemente, melhorando os resultados.

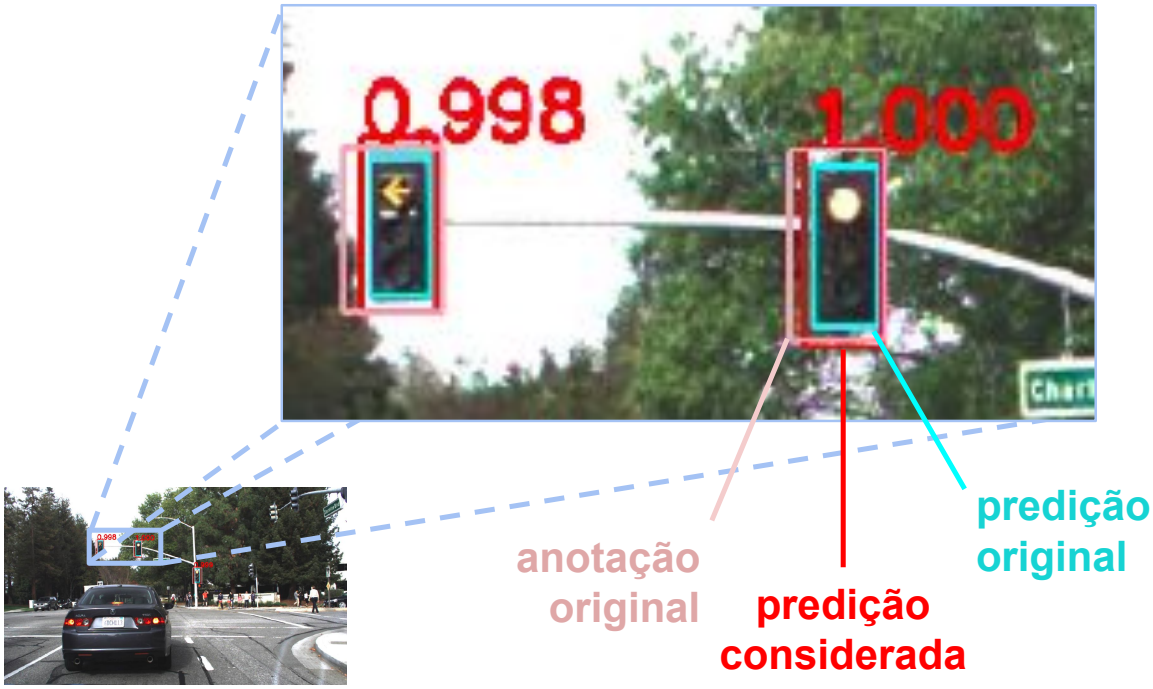



Figura 32 – Exemplo de resultado visual aplicado a uma imagem do *Udacity Driving Dataset*. As caixas delimitadoras vermelho-claras correspondem às rotulações originais, as caixas ciano correspondem às predições obtidas e as caixas vermelho-escuras correspondem às caixas de predição consideradas após a aplicação do melhor fator multiplicativo.

[19] <https://www.youtube.com/watch?v=DkgOyWxwMvE>

No canal do LCAD no YouTube, é possível conferir em formato de vídeo[19] uma comparação de desempenho entre o modelo proposto Contextualizado e o Referência Real.

# 6 Conclusão

A importância de detectar e reconhecer elementos de trânsito, em particular semáforos, para o êxito da direção autônoma contrasta com as dificuldades intrínsecas de se construir um modelo capaz de executar essa tarefa: a aquisição e rotulação de imagens retratando cenas de trânsito do mundo real é altamente laboriosa e sujeita à desbalanceamento de amostras de elementos de interesse mais improváveis de se capturar, como de semáforos em estado amarelo. Neste trabalho, foi demonstrado que um modelo satisfatório de detecção de semáforos pode ser obtido treinado-se com cenas de trânsito sintéticas aplicadas sobre planos de fundo arbitrários, oferecendo como vantagens (i) poupar o esforço de coleta de dados do mundo real, (ii) permitir rotulação automática das caixas delimitadoras durante o processo de geração da cena sintética e (iii) balancear as classes de interesse (neste caso os estados dos semáforos) conforme desejável.

Certo nível de desempenho na detecção de semáforos pode ser obtido apenas aplicando-se semáforos artificiais à planos de fundo arbitrários. Conferir valor semântico a esses semáforos adicionando-se contexto sintético à imagem torna os resultados ainda melhores, garantindo um valor médio aproximado de 50% de mAP e 56% de *F1-score* na detecção de semáforos em cenas do mundo real. Estes valores superam os obtidos com um modelo de referência treinado com imagens do mundo real em cerca de 4 p.p. cada.

Tomando como exemplo os próprios *datasets* de teste utilizados neste trabalho, pode-se observar que detecção de semáforos é um problema para o qual já existe uma boa quantidade de dados disponível para uso. No entanto, o conceito de um conjunto de dados sintético favorece a expansão dos padrões de semáforo para corresponder a padrões não abrangidos pelos *datasets* existentes, apenas implementando representações artificiais desses padrões. Em outras palavras, os *datasets* sintéticos podem ser remodelados de infinitas formas para corresponder tão bem quanto possível ao problema para o qual serão designados. Em sua forma original, o *dataset* DTLD, utilizado como Referência Real, mostra exemplos de semáforos horizontais ou com mais/menos de três bulbos presentes em cidades da Alemanha. Esses e outros padrões de semáforos poderiam ser incluídos no *dataset* proposto. Por outro lado, vale observar que a criação de cenas mais cada vez mais detalhadas e/ou realistas demanda maior esforço, contrariando a proposta inicial do projeto. Cabe ao projetista o balanceamento adequado entre a representação adequada do problema e a minimização do esforço em sua geração.

# Referências


ABADI, M. et al. Tensorflow: A system for large-scale machine learning. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. [S.l.: s.n.], 2016. p. 265–283. Citado na página 66.

BADUE, C. et al. Self-driving cars: A survey. *Expert Systems with Applications*, Elsevier, p. 113816, 2020. Citado 2 vezes nas páginas 25 e 64.

BARNES, D.; MADDERN, W.; POSNER, I. Exploiting 3d semantic scene priors for online traffic light interpretation. In: IEEE. *2015 IEEE Intelligent Vehicles Symposium (IV)*. [S.l.], 2015. p. 573–578. Citado na página 41.

BEHRENDT, K.; NOVAK, L.; BOTROS, R. A deep learning approach to traffic lights: Detection, tracking, and classification. In: IEEE. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2017. p. 1370–1377. Citado na página 41.

BERRIEL, R. F. et al. Ego-lane analysis system (elas): Dataset and algorithms. *Image and Vision Computing*, Elsevier, v. 68, p. 64–75, 2017. Citado na página 25.

BERRIEL, R. F. et al. Automatic large-scale data acquisition via crowdsourcing for crosswalk classification: A deep learning approach. *Computers & Graphics*, Elsevier, v. 68, p. 32–42, 2017. Citado na página 25.

BERRIEL, R. F. et al. Heading direction estimation using deep learning with automatic large-scale data acquisition. In: IEEE. *2018 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2018. p. 1–8. Citado na página 25.

CHARETTE, R. d. *Traffic Lights Recognition (TLR) public benchmarks*. 2013. <http://www.lara.prd.fr/benchmarks/trafficlightsrecognition>. Citado 2 vezes nas páginas 25 e 66.

CHEN, H. et al. Lstd: A low-shot transfer detector for object detection. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2018. v. 32, n. 1. Citado na página 42.

ECK, D. J. *Introduction to Computer Graphics*. [S.l.]: David J. Eck, 2016. Citado na página 39.

EISENBERG, J. D. *2D Transformations*. [S.l.]: Processing, n.d. <https://processing.org/tutorials/transform2d/>. Citado na página 40.

FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, Wiley Online Library, v. 7, n. 2, p. 179–188, 1936. Citado na página 31.

FREGIN, A. et al. The driveu traffic light dataset: Introduction and comparison with existing datasets. In: IEEE. *2018 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2018. p. 3376–3383. Citado 2 vezes nas páginas 25 e 61.

GÉRON, A. *Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow*. [S.l.]: Alta Books, 2019. Citado 6 vezes nas páginas 29, 31, 33, 34, 35 e 37.

GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1440–1448. Citado na página 38.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 580–587. Citado na página 38.

GOMEZ, A. E. et al. Traffic lights detection and state estimation using hidden markov models. In: IEEE. *2014 IEEE Intelligent Vehicles Symposium Proceedings*. [S.l.], 2014. p. 750–755. Citado na página 41.

GONZALEZ, E. et al. *Annotated Driving Dataset*. 2020. <https://github.com/udacity/self-driving-car/tree/master/annotations>. Citado na página 43.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 35.

GOOGLE. *Feature Crosses: Encoding Nonlinearity*. [S.l.]: Google Developers, 2020. <https://developers.google.com/machine-learning/crash-course/feature-crosses/encoding-nonlinearity>. Citado na página 33.

GOOGLE. *ML Practicum: Image Classification*. [S.l.]: Google Developers, 2020. <https://developers.google.com/machine-learning/practica/image-classification>. Citado na página 29.

GOOGLE. *ML Practicum: Image Classification: Introducing Convolutional Neural Networks*. [S.l.]: Google Developers, 2020. <https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks>. Citado na página 35.

GOOGLE. *Neural Networks: Structure*. [S.l.]: Google Developers, 2020. <https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/anatomy>. Citado na página 34.

HE, H.; GARCIA, E. A. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, Ieee, v. 21, n. 9, p. 1263–1284, 2009. Citado na página 42.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado 2 vezes nas páginas 37 e 66.

HUANG, C. et al. Learning deep representation for imbalanced classification. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 5375–5384. Citado na página 42.

HUBEL, D. H. Single unit activity in striate cortex of unrestrained cats. *The Journal of physiology*, Wiley Online Library, v. 147, n. 2, p. 226–238, 1959. Citado na página 31.

HUBEL, D. H.; WIESEL, T. N. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, Wiley Online Library, v. 148, n. 3, p. 574–591, 1959. Citado na página 31.

HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey

striate cortex. *The Journal of physiology*, Wiley Online Library, v. 195, n. 1, p. 215–243, 1968. Citado na página 31.

IIHS, H. *Red light running*. 1996–2021. https://www.iihs.org/topics/red-light-running. Citado na página 25.

JANG, C. et al. Multiple exposure images based traffic light recognition. In: IEEE. *2014 IEEE Intelligent Vehicles Symposium Proceedings*. [S.l.], 2014. p. 1313–1318. Citado na página 25.

JENSEN, M. B.; NASROLLAHI, K.; MOESLUND, T. B. Evaluating state-of-the-art object detector on challenging traffic light data. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2017. p. 9–15. Citado na página 25.

JENSEN, M. B. et al. Traffic light detection at night: Comparison of a learning-based detector and three model-based detectors. In: SPRINGER. *International Symposium on Visual Computing*. [S.l.], 2015. p. 774–783. Citado na página 41.

JENSEN, M. B. et al. Vision for looking at traffic lights: Issues, survey, and perspectives. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 17, n. 7, p. 1800–1815, 2016. Citado 2 vezes nas páginas 63 e 64.

KANG, B. et al. Few-shot object detection via feature reweighting. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 8420–8429. Citado na página 42.

KIM, H.-K.; PARK, J. H.; JUNG, H.-Y. An efficient color space for deep-learning based traffic light recognition. *Journal of Advanced Transportation*, Hindawi, v. 2018, 2018. Citado na página 76.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, v. 25, p. 1097–1105, 2012. Citado na página 31.

LI, X. et al. Traffic light recognition for complex scene with fusion detections. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 19, n. 1, p. 199–208, 2017. Citado na página 41.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. *European conference on computer vision*. [S.l.], 2014. p. 740–755. Citado 2 vezes nas páginas 43 e 57.

LYRIO, L. J. et al. Image-based mapping, global localization and position tracking using vg-ram weightless neural networks. In: IEEE. *2015 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2015. p. 3603–3610. Citado na página 25.

MARSCHNER, S.; SHIRLEY, P. *Fundamentals of computer graphics*. [S.l.]: CRC Press, 2018. Citado na página 39.

MUTZ, F. et al. Large-scale mapping in complex field scenarios using an autonomous car. *Expert Systems with Applications*, Elsevier, v. 46, p. 439–462, 2016. Citado na página 25.

OQUAB, M. et al. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 685–694. Citado na página 42.

OUYANG, W.; WANG, X. Joint deep learning for pedestrian detection. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2013. p. 2056–2063. Citado na página 25.

PADILLA, R. et al. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, Multidisciplinary Digital Publishing Institute, v. 10, n. 3, p. 279, 2021. Citado na página 38.

PHILIPSEN, M. P. et al. Traffic light detection: A learning algorithm and evaluations on challenging dataset. In: IEEE. *intelligent transportation systems (ITSC), 2015 IEEE 18th international conference on*. [S.l.], 2015. p. 2341–2345. Citado 2 vezes nas páginas 63 e 64.

PON, A. et al. A hierarchical deep architecture and mini-batch selection method for joint traffic sign and light detection. In: IEEE. *2018 15th Conference on Computer and Robot Vision (CRV)*. [S.l.], 2018. p. 102–109. Citado 2 vezes nas páginas 41 e 76.

POSSATTI, L. C. et al. Traffic light recognition using deep learning and prior maps for autonomous cars. In: IEEE. *2019 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2019. p. 1–8. Citado 3 vezes nas páginas 41, 64 e 76.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 779–788. Citado 2 vezes nas páginas 25 e 38.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. Citado 4 vezes nas páginas 25, 38, 39 e 66.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015. Citado na página 31.

SANGINETO, E. et al. Self paced deep learning for weakly supervised object detection. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 41, n. 3, p. 712–725, 2018. Citado na página 42.

SARCINELLI, R. et al. Handling pedestrians in self-driving cars using image tracking and alternative path generation with frenét frames. *Computers & Graphics*, Elsevier, v. 84, p. 173–184, 2019. Citado na página 25.

SHIFFMAN, D. *Coordinate System and Shapes*. [S.l.]: Processing, n.d. <https://processing.org/tutorials/drawing/>. Citado na página 40.

SHIFFMAN, D. *P3D*. [S.l.]: Processing, n.d. <https://processing.org/tutorials/p3d/>. Citado na página 40.

SHIFFMAN, D. *PShape*. [S.l.]: Processing, n.d. <https://processing.org/tutorials/pshape/>. Citado na página 40.

STANFORD. *Lecture 1 | Introduction to Convolutional Neural Networks for Visual Recognition*. [S.l.]: YouTube, 2017. <https://www.youtube.com/watch?v=vT1JzLTH4G4&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv>. Citado 2 vezes nas páginas 29 e 31.

STANFORD. *Lecture 11 | Detection and Segmentation*. [S.l.]: YouTube, 2017. <https://www.youtube.com/watch?v=nDPWywWRIRo&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=11>. Citado na página 38.

STANFORD. *Lecture 2 | Image Classification*. [S.l.]: YouTube, 2017. <https://www.youtube.com/watch?v=OoUX-nOEjG0&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=21>. Citado 3 vezes nas páginas 29, 31 e 34.

STANFORD. *Convolutional Neural Networks (CNNs / ConvNets)*. [S.l.]: CS231n: Convolutional Neural Networks for Visual Recognition, n.d. <https://cs231n.github.io/convolutional-networks/>. Citado na página 35.

STANFORD. *Image Classification*. [S.l.]: CS231n: Convolutional Neural Networks for Visual Recognition, n.d. <https://cs231n.github.io/classification/>. Citado na página 29.

STANFORD. *Quick intro*. [S.l.]: CS231n: Convolutional Neural Networks for Visual Recognition, n.d. <https://cs231n.github.io/neural-networks-1/>. Citado na página 34.

SZELISKI, R. *Computer vision: algorithms and applications*. [S.l.]: Springer Science & Business Media, 2010. Citado 2 vezes nas páginas 29 e 31.

TENSORFLOW. *Aumento de dados*. [S.l.]: TensorFlow, 2021. <https://www.tensorflow.org/tutorials/images/data_augmentation>. Citado na página 37.

TORRES, L. T. et al. Effortless deep training for traffic sign detection using templates and arbitrary natural images. In: IEEE. *2019 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2019. p. 1–7. Citado 7 vezes nas páginas 25, 26, 39, 42, 50, 59 e 66.

WANG, K. et al. Towards human-machine cooperation: Self-supervised sample mining for object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2018. p. 1605–1613. Citado na página 42.

WANG, Y.-X.; RAMANAN, D.; HEBERT, M. Learning to model the tail. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. [S.l.: s.n.], 2017. p. 7032–7042. Citado na página 42.

YU, F. et al. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. *arXiv preprint arXiv:1805.04687*, 2018. Citado 2 vezes nas páginas 25 e 57.

# Apêndices

# APÊNDICE A – Acesso ao projeto

Boa parte do conteúdo referente ao projeto se encontra disponível para acesso público no seguinte endereço:

- <https://github.com/Jpvmello/traffic-light-detection-synthetic-context>

O repositório inclui:

- código para geração de *datasets* sintéticos dados os conjuntos de planos de fundo e de imagens de primeiro plano;
- os modelos treinados descritos neste trabalho;
- os conjuntos de imagens de primeiro plano utilizados para o treinamento dos modelos aqui descritos;
- o *dataset* proprietário IARA;
- código para geração de imagens de primeiro plano.

O *dataset* proposto não foi disponibilizado pronto devido à restrições de licença de distribuição de algumas imagens do *dataset* COCO.

# APÊNDICE B – Resultados numéricos

A Tabela 3 mostra numericamente os resultados de mAP e *F1-score* exibidos na Figura 26 e os valores por modelo do melhor multiplicador de caixa e do limiar de confiança que leva ao melhor *F1-score*. A tabela traduz o material suplementar do artigo publicado, disponível sob a licença CC-BY 4.0[20], corrigindo alguns valores repetidos.

[20] <http://creativecommons.org/licenses/by/4.0/>

| ***Dataset*** **de teste** | **Modelo treinado** | **Melhor multip. de caixa** | **mAP (%)** | **Melhor limiar de confiança** | **Melhor *F1-score* (%)** |
|---|---|---|---|---|---|
| | Contextualizado | 1,3 | 52,03 | 0,86 | 61,43 |
| | Descontextualizado | 1,4 | 48,33 | 0,93 | 53,16 |
| | Modelos-2D | 1,3 | 33,32 | 0,96 | 43,21 |
| | Planos de Fundo Positivos | 1,0 | 16,14 | 0,95 | 21,80 |
| LISA_treino+teste | Planos de Fundo Negativos | 1,4 | 44,46 | 0,80 | 53,11 |
| | Referência Real | 1,8 | 50,33 | 0,92 | 60,87 |
| | Contexto+Real70 | 1,4 | 56,25 | 0,92 | 62,66 |
| | Contexto+Real140 | 1,4 | 54,50 | 0,88 | 61,10 |
| | Contexto->Real | 1,8 | 53,55 | 0,84 | 60,20 |
| | Contextualizado | 1,3 | 46,68 | 0,86 | 54,31 |
| | Descontextualizado | 1,4 | 44,46 | 0,93 | 49,95 |
| | Modelos-2D | 1,3 | 29,02 | 0,96 | 39,33 |
| | Planos de Fundo Positivos | 1,0 | 22,16 | 0,95 | 29,17 |
| LISA_teste | Planos de Fundo Negativos | 1,4 | 39,19 | 0,80 | 48,13 |
| | Referência Real | 1,8 | 47,88 | 0,92 | 58,62 |
| | Contexto+Real70 | 1,4 | 57,57 | 0,92 | 63,81 |
| | Contexto+Real140 | 1,4 | 59,52 | 0,88 | 65,17 |
| | Contexto->Real | 1,8 | 49,81 | 0,84 | 60,67 |
| | Contextualizado | 1,6 | 43,58 | 0,86 | 49,75 |
| | Descontextualizado | 1,8 | 40,11 | 0,93 | 41,43 |
| | Modelos-2D | 1,7 | 22,26 | 0,96 | 33,97 |
| | Planos de Fundo Positivos | 1,6 | 25,37 | 0,95 | 35,92 |
| Udacity- | Planos de Fundo Negativos | 1,7 | 42,84 | 0,80 | 40,58 |
| | Referência Real | 1,9 | 25,66 | 0,96 | 36,77 |
| | Contexto+Real70 | 1,9 | 43,91 | 0,92 | 46,08 |
| | Contexto+Real140 | 1,9 | 42,34 | 0,88 | 44,47 |
| | Contexto->Real | 1,9 | 32,21 | 0,84 | 38,32 |
| | Contextualizado | 1,8 | 45,74 | 0,86 | 50,94 |
| | Descontextualizado | 1,9 | 40,16 | 0,93 | 41,61 |
| | Modelos-2D | 1,9 | 22,75 | 0,96 | 34,87 |
| | Planos de Fundo Positivos | 1,9 | 26,74 | 0,95 | 37,62 |
| Udacity+ | Planos de Fundo Negativos | 1,9 | 43,31 | 0,80 | 41,07 |
| | Referência Real | 1,9 | 21,81 | 0,96 | 34,04 |
| | Contexto+Real70 | 1,9 | 42,85 | 0,92 | 44,88 |
| | Contexto+Real140 | 1,9 | 41,31 | 0,88 | 43,30 |
| | Contexto->Real | 1,9 | 28,92 | 0,84 | 36,04 |
| | Contextualizado | 0,6 | 52,55 | 0,86 | 55,78 |
| | Descontextualizado | 0,8 | 49,27 | 0,93 | 39,60 |
| | Modelos-2D | 1,0 | 19,66 | 0,96 | 28,35 |
| | Planos de Fundo Positivos | 1,0 | 1,54 | 0,95 | 3,97 |
| LaRA | Planos de Fundo Negativos | 0,8 | 35,58 | 0,80 | 31,04 |
| | Referência Real | 1,0 | 63,84 | 0,92 | 60,65 |
| | Contexto+Real70 | 0,7 | 70,93 | 0,92 | 58,77 |
| | Contexto+Real140 | 0,7 | 69,92 | 0,88 | 56,35 |
| | Contexto->Real | 0,8 | 61,44 | 0,84 | 46,37 |
| | Contextualizado | 0,8 | 59,92 | 0,86 | 63,36 |
| | Descontextualizado | 1,0 | 56,13 | 0,93 | 49,24 |
| | Modelos-2D | 1,1 | 33,66 | 0,96 | 35,98 |
| | Planos de Fundo Positivos | 1,1 | 22,18 | 0,95 | 20,02 |
| IARA | Planos de Fundo Negativos | 1,0 | 61,59 | 0,80 | 57,14 |
| | Referência Real | 1,1 | 64,12 | 0,84 | 62,30 |
| | Contexto+Real70 | 1,0 | 68,29 | 0,92 | 69,12 |
| | Contexto+Real140 | 1,0 | 67,61 | 0,88 | 65,64 |
| | Contexto->Real | 1,3 | 66,27 | 0,84 | 64,08 |

Tabela 3 – Resultados de teste para todos os *datasets* de teste e modelos treinados.