Bruno Marques

Avaliação de algoritmos baseados em Deep Learning para localizar placas veiculares brasileiras em ambientes complexos

Brasil

2018

Bruno Marques

Avaliação de algoritmos baseados em Deep Learning para localizar placas veiculares brasileiras em ambientes complexos

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal Rural de Pernambuco

Departamento de Computação

Bacharelado em Ciência da Computação

Orientador: Valmir Macario

Brasil

2018

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas da UFRPE
Biblioteca Central, Recife-PE, Brasil

M357a Marques, Bruno Henrique Pereira.
Avaliação de algoritmos baseados em Deep Learning para Localizar placas veiculares brasileiras em ambientes complexos / Bruno Henrique Pereira Marques. - Recife, 2018.
60 f.: il.

Orientador(a): Valmir Macário Filho.
Trabalho de Conclusão de Curso (Graduação) – Universidade Federal Rural de Pernambuco, Departamento de Computação, Recife, BR-PE, 2019.
Inclui referências.

1. Inteligência artificial 2. Aprendizagem profunda 3. YOLO 4. Rede neural convolucional 5. Otimização de hiperparâmetros 6. Placas veiculares I. Macário Filho, Valmir, orient. II. Título

CDD 004

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por Bruno Henrique Pereira Marques às 11 horas do dia 15 de janeiro de 2019, no Auditório do CEAGRI-02 – Sala 07, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado " **Avaliação de algoritmos baseados em Deep Learning para localizar placas veiculares brasileiras em ambientes complexos** ", orientado por Valmir Macario Filho e aprovado pela seguinte banca examinadora:

_____

Valmir Macario Filho

DC/UFRPE

_____

Filipe Rolim Cordeiro
DC/UFRPE

"Entrega o teu caminho ao Senhor; confia nele, e Ele tudo fará."
Salmos 37:5

Agradecimentos

Agradeço a Deus por ter me abençoado com saúde e força para enfrentar as dificuldades. Pelas pessoas de bem que tem cruzado o meu caminho. Agradeço a Ele pelas oportunidades, vitórias e conquistas alcançadas durante minha vida.

Agradeço à minha família, meus pais Vera e Marcos, minha irmã Andressa e meu cunhado Diego, pela educação e ensinamentos de vida, pelo apoio e pela compreensão do convívio muitas vezes poupado para os meus estudos e trabalhos. Agradeço a meus primos, tios e avós, também pela força que me deram nesses anos de caminhada.

Aos meus professores, por toda experiência e conhecimento passados. Em especial ao meu orientador Prof. Valmir Macário pela confiança no meu esforço, por todas as oportunidades e motivação que me deu. Da mesma forma, agradeço aos professores César França, Kellyton Brito e Vanilson Burégio, os quais potencializaram a minha experiência durante o projeto de extensão na universidade.

Aos meus amigos, pela ajuda e fraternidade durante toda esta jornada. Especialmente aos da turma de Ciência da Computação de 2013.2. Meus eternos agradecimentos a Danielly Queiroz. Agradeço pela colaboração na construção do banco de imagens deste trabalho. Por ter tornado esta caminhada mais leve, com amor e luz. Pelos momentos de carinho e descontração, essenciais para meu descanso físico e mental. Ao meu amigo Sérgio Chevtchenko, muito obrigado pelas instruções e força dadas a mim e a este trabalho.

E a Universidade Federal Rural de Pernambuco, aos funcionários que zelam e fazem da universidade um local familiar e de crescimento humano.

Resumo

Com o aumento da quantidade de veículos particulares, nota-se o crescimento do número de violações das leis de trânsito, roubo de veículos e assim, se faz necessário uma melhor gestão e fiscalização do tráfego. Um veículo e seu proprietário são reconhecidos através da placa veicular (PV) única e obrigatória, e para que sejam fiscalizados e terem dados extraídos com maior eficiência, é recomendável a utilização de sistemas automatizados para detecção e reconhecimento de placas veiculares. Este trabalho apresenta um estudo e avaliação de algoritmos baseados em Aprendizagem Profunda para localizar PV brasileiras em ambientes complexos. Para a realização dos experimentos, foi criado um banco de imagens de PV brasileiras baseada em problemas como imagens com resolução, qualidade, iluminação e perspectiva de cena diferentes. Foram utilizados os algoritmos de Aprendizagem Profunda YOLOv2 e YOLOv3, o qual ainda não foi estudado para o melhor do nosso conhecimento, neste contexto. Além disso, foi utilizado o algoritmo *Tree-structured Parzen Estimator* (TPE) para realizar a otimização de hiperparâmetros e maximizar o desempenho das redes neurais convolucionais selecionadas. Para a avaliação, foram utilizadas as métricas de desempenho: tempo de predição, *Intersection over Union* (IoU) e taxa de confiança. O resultado dos experimentos mostrou que o YOLOv3 apresentou melhor desempenho, obtendo 99.3% de detecção das placas veiculares.

Palavras-chave: Inteligência Artificial, Aprendizagem Profunda, YOLO, Rede Neural Convolucional, Otimização de Hiperparâmetros, Placas Veiculares.

Abstract

With the increase in the number of private vehicles, we can observe the increase in the number of violations of traffic laws, theft of vehicles and, thus, a better management and traffic control is necessary. A vehicle and its owner are recognized through the unique and required vehicle license plate (LP), and to be inspected and extracted data with greater efficiency, it is recommended to use automated systems for detecting and recognizing vehicle license plates. This work introduce a study and evaluation of algorithms based on Deep Learning to locate Brazilian LPs in complex environments. For the achievement of the experiments, a bank of images of Brazilian LPs was created based on problems like images with different resolution, quality, lighting and perspective of scene. Were used the Deep Learning algorithms YOLOv2 and YOLOv3, which has not yet been studied to the best of our knowledge. In addition, the Tree-structured Parzen Estimator (TPE) algorithm was used to optimize hyperparameters and maximize the performance of selected convolutional neural networks. For the evaluation, the performance metrics were used: prediction time, Intersection over Union (IoU) and confidence rate. The experiments result demonstrate that YOLOv3 presented better performance, obtaining 99.3% of vehicle license plate detection.

Key-words: Artificial Intelligence, Deep Learning, YOLO, Convolutional Neural Networks, Hyperparameter Optimization, License Plate.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

No Brasil, diante de um conjunto de problemas de segurança e transporte público, a necessidade de utilizar veículos particulares é crescente (O Globo, 2018). Dado o aumento do número de veículos particulares (G1 Autoesporte, 2018), acompanha-se o crescimento do número de violações das leis de trânsito, que remete a necessidade de um melhoramento na fiscalização de vias e uma melhor gestão em áreas de acesso restrito. Para a indústria, a análise de dados sobre automóveis de clientes que frequentam um estabelecimento pode ser utilizada para auxiliar decisões estratégicas (RIBEIRO et al., 2017). Além disso, os dados mais recentes mostram que a cada minuto um veículo é roubado ou furtado no país (Folha SP, 2017). Sendo assim, um monitoramento de placas veiculares automatizado pode ser mais eficiente na detecção destas placas, entre outras questões (XIE et al., 2018). Uma placa veicular (PV) é única e obrigatória para todos os veículos que circulam no Brasil, elas representam dados do veículo como: marca, modelo, ano, cor, dados do proprietário, entre outros. Até a construção deste trabalho, as placas veiculares no país possuem um padrão de 3 letras e 4 números em tamanho destacado, a sigla do estado e o nome do município em menor tamanho, conforme a Figura 1. Além disso, existem combinações de cores de plano de fundo e dos caracteres para destacar a categoria a qual o veículo pertence. Para a fiscalização ou extração de dados que atenda a tantas combinações e a crescente frota com rapidez e eficiência, é recomendável a utilização de sistemas automatizados para detecção e reconhecimento de placas veiculares (SADRPV).

Várias propostas de SADRPV, baseadas em diversas operações de processamento de imagens, tais como, análise de regiões com janelas deslizantes (PRATES et al., 2014), operações morfológicas (FONSECA-GALINDO et al., 2016) e detecção de bordas (SAGHA-RICHI; SHAKERI, 2016), podem ser encontradas na literatura. Estas operações são importantes para reduzir detalhes na imagem que dificultam o funcionamento de SADRPV como por exemplo, uma forte luz que reflete na lataria de um veículo e incide diretamente na lente da câmera, pode gerar um ofuscamento prejudicando a localização e detecção da placa. Outro tipo de problema ocorre quando a imagem possui baixa resolução ou apresenta ruídos consideráveis. Estes foram apenas exemplos entre casos ilimitados e complexos. Sendo assim, propor um método baseado em processamento de imagens e que atenda a muitas possibilidades de casos, é relativamente difícil (XIE et al., 2018).

Figura 1 – Placas veiculares brasileiras. a) Particulares; b) Transportes; c) República; d) Autoescola; e) Coleção; f) Oficiais; g) Experiência; e h) Diplomatas.

Fonte – AutoIdeias.

1.1 Problema de Pesquisa

O problema de pesquisa deste trabalho baseia-se em como localizar placas veiculares brasileiras em ambientes complexos. Um ambiente é dito como complexo quando uma base de dados possui imagens com cenários diferentes. Já um ambiente é simples quando uma base de dados possui imagens com cenários repetidos, ou seja, extraídas através de uma câmera fixa em uma determinada posição (Figura 2). Dessa forma, é preciso generalizar a análise de características para que a diversidade de cenários não afete o desempenho da localização de PV.

Figura 2 – a) Ambientes complexos; e b) Ambientes simples.

Fonte – 2a: Internet. 2b: UFOP-Gate (PEIXOTO et al., 2015).

Algoritmos baseados em aprendizagem profunda vêm sendo aplicado em trabalhos recentes para executar a tarefa de detecção de placas veiculares, como: Faster R-CNN (REN et al., 2015), Single Shot Multibox Detector - SSD (LIU et al., 2016) e RNC com o YOLOv2 (LAROCA et al., 2018). Esses algoritmos foram aplicados separadamente para detectar padrões em países diferentes, incluindo o Brasil, e obtiveram bons níveis de

generalização e resultados. Sendo assim, é preciso avaliar os algoritmos estudados para saber quais são os promissores em detectar PV com o padrão brasileiro.

A tarefa de localizar PV em imagens possui obstáculos que a torna desafiadora, visto que são passíveis de problemas, como: ângulo de perspectiva da cena, condições de iluminação, distância entre a placa e a câmera, nível de ruído, objetos que podem ser confundidos com placas veiculares, entre outros que influenciam o desempenho (FONSECA-GALINDO et al., 2016). A Figura 3 apresenta três exemplos desses problemas.

Figura 3 – a) Sombreamento; b) Informações ruidosas; e c) Claridade.

Fonte – Coletadas na internet.

Neste trabalho foi montado um banco de imagens com base nos obstáculos citados anteriormente. Todas as imagens foram coletadas em diversos sites, principalmente aquelas destacadas pelo Google Images (Google, 2018), com o objetivo de obter-se maior variedade de resolução e qualidade da imagem. Neste trabalho foram utilizados dois algoritmos baseados em Aprendizagem Profunda contidos no Yolo: i) a versão 2, avaliada em trabalhos recentes na literatura(LAROCA et al., 2018); e ii) a versão 3, publicada recentemente e que ainda não foi analisada até o melhor do nosso conhecimento. Além disso, foi executado o algoritmo *Tree-structured Parzen Estimator* (TPE) para a otimização de hiperparâmetros, afim de explorar várias combinações de parâmetros no espaço de busca onde os resultados podem ser maximizados. A otimização de hiperparâmetros para esse problema também não foi realizada por nenhum outro trabalho, no melhor do nosso conhecimento. Por fim, para as localizações das placas veiculares, foi avaliado o desempenho dos algoritmos em termos de i) tempo de predição; ii) taxa de confiança; e iii) *Intersection over Union* (IoU).

1.2 Objetivos

1.2.1 Objetivo Geral

- Avaliar algoritmos baseados em Aprendizagem Profunda para localizar placas veiculares brasileiras em ambientes complexos.

1.2.2 Objetivos Específicos

1. Estudar os algoritmos de Aprendizagem Profunda de detecção do *framework* Yolo.
2. Obter métricas e avaliá-las em termos de qualidade na localização da placa e o tempo de predição.

1.3 Organização do Trabalho

Este trabalho está organizado com a seguinte estrutura: o Capítulo 2 apresenta os fundamentos básicos. Problemas que sistemas automatizados para detecção e reconhecimento de placas veiculares (SADRPV) enfrentam; como são as propostas dos métodos clássicos, as quais se baseiam em processamento de imagens; como são as propostas mais recentes apresentadas na literatura; também é mostrada uma introdução a Redes Neurais Convolucionais; e o funcionamento do algoritmo YOLO.

O Capítulo 3 contém uma revisão da literatura, com trabalhos relacionados ao tema aqui abordado. Estes trabalhos foram essenciais para o estudo sobre a evolução das tecnologias utilizadas em SADRPV, para a definição da metodologia utilizada e para a definição das métricas de desempenho.

O Capítulo 4 traz todo o conteúdo sobre a localização de placas veiculares. Sobre como este trabalho foi desenvolvido para chegar ao resultado esperado, enfrentando diversos obstáculos presentes nas imagens.

No Capítulo 5 descreve a metodologia seguida para a realização do começo ao fim deste trabalho. Como o banco de imagens foi criado, quais as especificações técnicas do computador utilizado e a definição para a execução e avaliação dos experimentos.

No Capítulo 6 são apresentados os resultados dos experimentos. Uma avaliação também é mostrada sobre as métricas de desempenho coletadas em termos do tempo de predição em segundos, IoU e taxa de confiança dos algoritmos.

O Capítulo 7 contém a conclusão deste trabalho uma visão para trabalhos futuros.

2 Conceitos Básicos

2.1 O Problema de Localizar Placa

Diante dos desafios que os sistemas automatizados para detecção e reconhecimento de placas veiculares (SADRPV) enfrentam, encontram-se metodologias com base em técnicas clássicas, as quais aplicam filtros específicos de processamento de imagem, afim de destacar características dos objetos presentes na cena para, em seguida, executar uma determinada análise onde a placa deverá ser devidamente localizada. Combinando-se às técnicas, é possível aplicar descritores, que também são responsáveis por destacar e quantificar características dos elementos que compõem a imagem. Além disso, encontra-se técnicas de segmentação de imagem, onde é possível realizar operações específicas em pequenas regiões da imagem ou separá-las para uma segunda etapa de processamento.

A aplicação de técnicas de pré-processamento nas imagens tem objetivo de reduzir detalhes na imagem que dificultam a localização de objetos (YAN-LI, 2015). Alguns trabalhos utilizam essas técnicas para melhorar as condições da cena da imagem para localizar placas (S.JYOTHI et al., 2017) (WANG; GAO; YANG, 2009) (CHEN et al., 2013). Existem várias técnicas que podem ser combinadas para melhorar ainda mais o destaque de objetos como: nitidez, brilho, contraste, binarização, suavização, redução de ruídos, inversão de cores, etc. Para melhor entendimento, a Figura 4 apresenta duas imagens: a imagem da esquerda (4a) mostra a entrada original, ou seja, sem nenhum pré-processamento aplicado. Já a imagem da direita (4b) mostra como a aplicação da técnica que altera o brilho e contraste clareando e melhorando os contornos da placa.

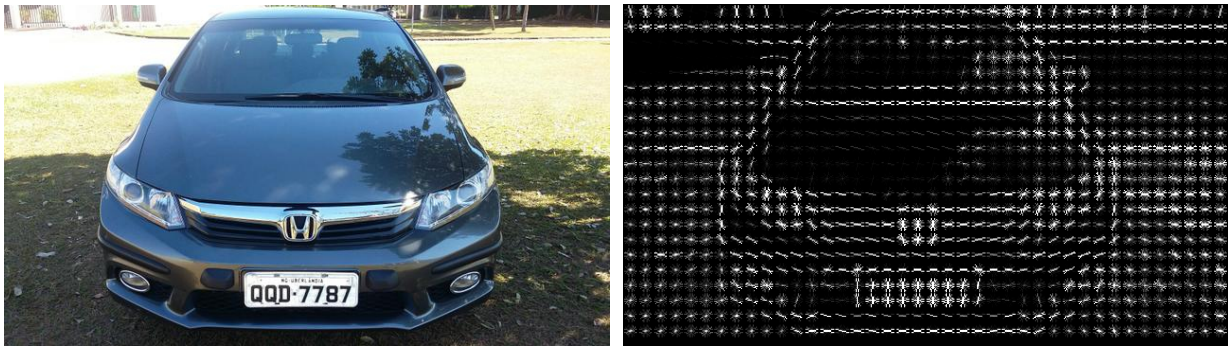
Figura 4 – Exemplo de filtros.

(a) Sem pré-processamento (b) Melhoria no brilho e contraste

Fonte – Do autor.

Além dos filtros, os descritores também são utilizados para, não apenas destacar, mas também para descrever as características dos objetos que pertencem a cena da imagem. Os descritores geraram um conjunto de valores a partir de uma imagem. Esses valores podem representar medidas estatísticas, de formas ou texturas relacionados ao conteúdo da imagem. A coleta desses dados é uma parte fundamental para diferenciar um objeto de outro bem como realizar o reconhecimento. Um exemplo de descritor é o Histograma de Gradientes Orientados (HOG), o qual foi bastante utilizado em trabalhos para a localização de placas veiculares (PRATES et al., 2014) (ASTAWA et al., 2017) (GOU et al., 2014). HOG analisa a imagem em pequenas células para calcular individualmente a intensidade e a orientação da alternância de cores. Para cada pixel de uma imagem, são analisados os pixels vizinhos diretamente conectados, com objetivo de determinar quão escuro é o pixel em comparação aos seus vizinhos. Dessa forma, a orientação da alternância de cores em relação a todos os pixels da imagem é mapeada. A Figura 5 apresenta a imagem original (5a) e a imagem resultante da aplicação do HOG (5b). O trabalho de Prates *et al.* (PRATES et al., 2014) conseguiu atingir boa acurácia de cerca de 95%, utilizando HOG e *Support Vector Machine* (SVM) para a detecção de placas. Dessa forma, alinhando-se técnicas de processamento de imagens e algoritmos de aprendizagem de máquina, consegue-se obter bons resultados.

Figura 5 – Aplicação do HOG.



(a) Imagem original (b) HOG aplicado

Fonte – Do autor.

Uma das técnicas de segmentação encontrada na literatura, é a operação com janelas deslizantes (PRATES et al., 2014) (ANAGNOSTOPOULOS et al., 2005) (WANG; LIN; HORNG, 2011). Janela deslizante é uma região na imagem a qual será processada até que a imagem original seja completamente analisada. Normalmente essas janelas percorrem a imagem começando do canto superior esquerdo, até o final, ou seja, canto inferior direito. Geralmente o tamanho das janelas deslizantes são definidos de acordo com o objeto a ser detectado, como as placas veiculares possuem formatos retangulares, o tamanho recomendado é de 128 x 48 pixels ou proporcional (ASTAWA et al., 2017). Como mostra a Figura 6, a janela destacada em vermelho, percorre por toda a imagem realizando algum processamento e análise de características. Em verde, temos a localização da placa veicular.

Figura 6 – Janela deslizante.

Fonte – Do autor.

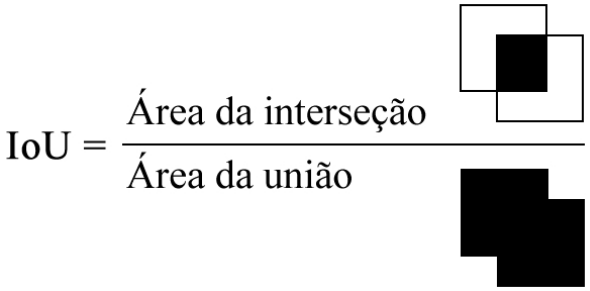
Com a evolução dos algoritmos de inteligência artificial e da acessibilidade para maiores recursos computacionais, nos últimos anos têm se pesquisado sobre a aplicação de algoritmos baseados em aprendizagem profunda, tais como: Faster R-CNN (REN et al., 2015), Single Shot Multibox Detector - SSD (LIU et al., 2016) e RNC com o YOLOv2 (LAROCA et al., 2018). Esses algoritmos, devido a grande capacidade de generalização, não necessitam da implementação manual de etapas para pré-processamento de imagens e extração de características, visto que realizam essas etapas internamente. De acordo com o trabalho de Xie *et al.* (XIE et al., 2018), utilizando o Yolo, conseguiu obter ótimos resultados alterando sua estrutura para detectar placas que variam de a rotação entre -30 a 30 graus, chegando a 93% de acurácia com 60% de IoU (*Intersection over Union*). IoU é uma das métricas bastante utilizada para medir o desempenho das predições dos algoritmos. Para melhor entendimento, a seção 2.2 apresenta seu funcionamento.

2.2 Métricas de desempenho

Algoritmos para detecção de objetos têm sido utilizado em vários contextos. Seja para o contexto de reconhecimento facial, medicinal ou rastreamento de veículos, por exemplo, esses algoritmos precisam fornecer resultados satisfatórios para serem viáveis. Esta seção descreve duas métricas das mais utilizadas, que calculam o desempenho em relação aos resultados dos treinamentos e predições dos algoritmos.

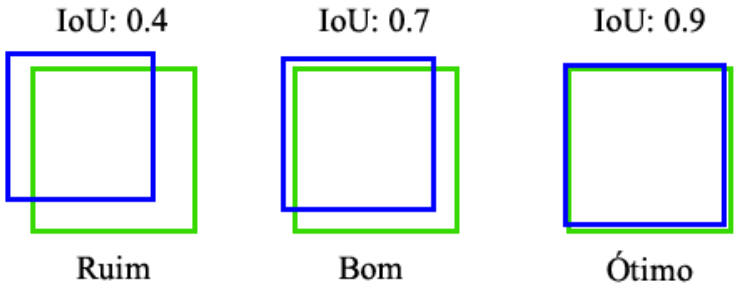
Intersection over Union, ou simplesmente IoU, é uma métrica utilizada para definir quanto uma predição da localização de um objeto na imagem coincide com a sua verdadeira localização (ROSEBROCK, 2016). A Figura 7 mostra que para obter o IoU, é necessário dividir a interseção pela união entre a área da região correta do objeto e a área resultante da predição. Com valores de porcentagem, quanto maior for o valor do IoU, mais preciso será a seleção de um objeto, porém pode restringir o algoritmo. Já um valor baixo, faz com que o algoritmo tenha uma margem de aceitação alta, porém não fica com tanta precisão. A Figura 8 mostra a relação dos valores do IoU.

Figura 7 – Cálculo do IoU



Fonte – Adaptado de Rosebrock (2016).

Figura 8 – IoU. Em verde é a localização exata de um objeto na imagem. Em Azul é a localização estimada de acordo com o IoU.



Fonte – Adaptado de Rosebrock (2016).

Outra métrica comumente utilizada, porém para o treinamento de algoritmos, é a *Mean Average Precision*, ou mAP. Para calcular mAP, é necessário entender como se calcula *Average Precision* (AP). AP é o resultado da razão entre a precisão e *recall* coletados durante o treinamento de um algoritmo. A Figura 9 apresenta a equação para se obter a precisão e *recall*, respectivamente. Ao final do treinamento, depois de obter-se um conjunto de APs, o mAP é calculado através da média aritmética desse conjunto, de forma que, na escala de porcentagem, quanto maior for o valor de mAP melhor é o resultado.

Figura 9 – Cálculo da precisão e *recall*.

$$p = \frac{\text{Verdadeiro Positivo}}{\text{Verdadeiro Positivo} + \text{Falso Positivo}} \qquad r = \frac{\text{Verdadeiro Positivo}}{\text{Verdadeiro Positivo} + \text{Falso Negativo}}$$

(a) Precisão (b) *Recall*

Fonte – Do autor com base em Henderson e Ferrari (2017).

2.3 Rede Neural Convolucional

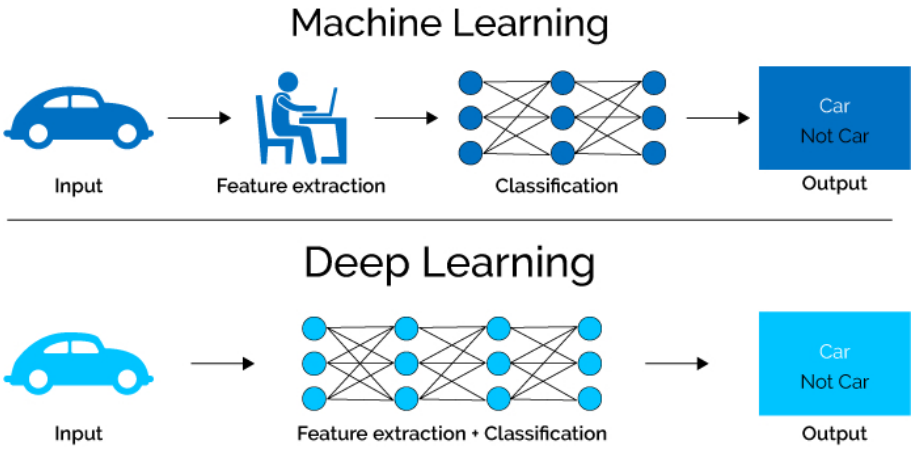
Dentro do conjunto de técnicas relacionadas a aprendizagem de máquina, existe o subconjunto de algoritmos baseados em Aprendizagem Profunda (CHEVTCHENKO, 2018). A popularidade desses algoritmos cresceram consideravelmente com o avanço das tecnologias para processamento gráfico (GPU) pois, com mais recursos, o tempo de treinamento desses algoritmos é menor (SCHMIDHUBER, 2015). Dessa forma, a acurácia das predições aumentaram e isso fez os algoritmos baseados em Aprendizagem Profunda superarem técnicas tradicionais de Aprendizagem de Máquina em problemas de grande relevância (SCHMIDHUBER, 2015).

Redes Neurais Convolucionais (CNNs) são técnicas que pertencem ao subconjunto de algoritmos baseados em Aprendizagem Profunda. Esses algoritmos se destacam pela

possibilidade de instanciar muitas camadas, afim de compreender um objeto de entrada em vários níveis de representação. Cada um dos níveis é responsável por transformar o objeto de entrada em uma representação mais genérica de si mesma (BENGIO, 2011). Em outras palavras, um algoritmo baseado em aprendizagem profunda extrai automaticamente características de alto nível necessárias para a classificação.

As CNNs possuem duas propriedades de grande importância e que as tornam atrativas: i) facilidade de treinamento; e ii) grande capacidade de generalização (SCHMIDHUBER, 2015). Dessa forma, esses algoritmos não dependem de uma etapa de implementação para extração de características em relação ao problema abordado (CHEVTCHENKO, 2018), como por exemplo, uma CNN utilizada para detectar veículos pode ser reutilizada para detectar animais sem que seja necessário readaptar a etapa de extração de características das classes dos animais. A Figura 10 apresenta as etapas de algoritmos convencionais em comparação com as etapas de algoritmos baseados em aprendizagem profunda, como as CNNs.

Figura 10 – Etapas de algoritmos convencionais e etapas de algoritmos baseados em aprendizagem profunda.

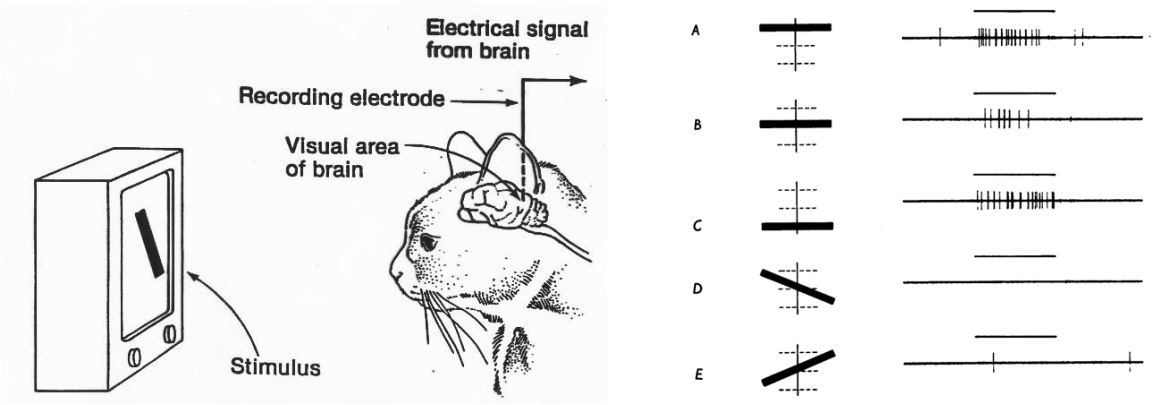


Fonte – Medium - George Seif (2018).

2.3.1 Inspiração Biológica

Em 1961, HUBEL e WIESEL estudaram o comportamento dos neurônios no córtex visual de um gato em relação a formas e orientações do que o animal vê. Nos experimentos, HUBEL e WIESEL descobriram que os neurônios mais próximos da retina são responsáveis pela captação de pontos de luz. Em seguida, essas informações são encaminhadas para o córtex visual e redistribuídas para outras camadas. Uma parte do experimento foi apresentar um bloco preto com formato retangular a um gato, afim de estimular os neurônios. Como mostra a Figura 11, a posição do bloco retangular influencia a forma que os neurônios são estimulados.

Figura 11 – Ilustração do comportamento dos neurônios no córtex visual de um gato.

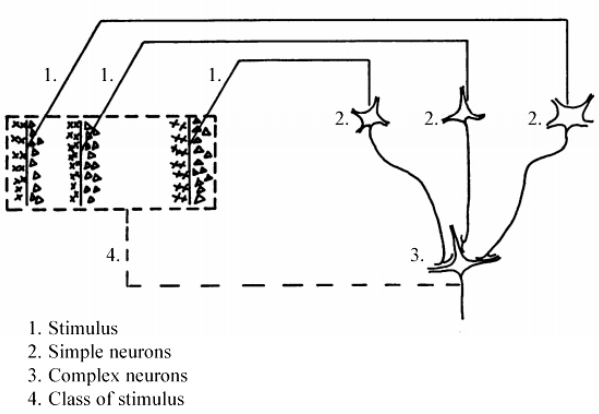


(a) Estímulo com bloco retangular (b) Posições do bloco e a formação dos neurônios

Fonte – HUBEL e WIESEL (1961).

Para representar a formação desses neurônios no córtex visual, HUBEL e WIESEL propuseram um esquema apresentado na Figura 12. Os neurônios sensíveis a pontos de luz são conectados a um neurônio mais complexo no córtex visual, dessa forma, ele pode reagir ou representar o mesmo objeto em posições invariantes a escala, translação, rotação e transformações.

Figura 12 – Esquema para a formação de neurônios no córtex visual.



Fonte – Adaptado de HUBEL e WIESEL (1961).

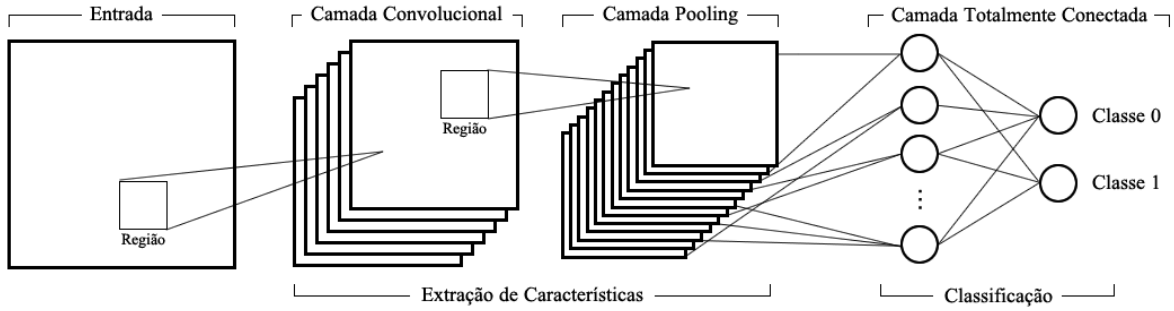
Mecanismos de neurônios biológicos ainda inspiram pesquisadores que buscam formas de generalizar e representar melhor os dados. As redes neurais convolucionais, inspiradas no esquema do córtex visual apresentado acima, têm mostrado ótimas respostas em várias aplicações das inúmeras áreas de pesquisa (CHEVTCHENKO, 2018). Da mesma forma que o córtex visual, as redes neurais convolucionais utilizam neurônios dos primeiros níveis para extração de características e as redirecionam para níveis mais complexos, dessa forma, elava-se a capacidade de generalização dos dados.

2.3.2 Principais Componentes

As Redes Neurais Convolucionais (CNNs) utilizam um sistema de hierarquia ser capaz de reconhecer uma imagem, onde o conjunto dos *pixels* representam arestas; o conjunto destas arestas representam padrões; que por sua vez representam objetos e assim, compõem uma cena (AREL; ROSE; KARNOWSKI, 2010).

Segundo O'SHEA e NASH (2015), a arquitetura de uma CNN, em geral, é baseada em três tipos de camada. A primeira é a camada Convolucional, onde é realizada a operação de convolução. Nesta camada são processadas pequenas regiões da imagem, replicando a operação até completá-la. Em seguida, os dados são passados para a camada *Pooling*. Essa camada gera uma versão com menor resolução do que a Convolucional, aplicando uma função de ativação em várias posições no interior de uma região. Dessa forma, regiões específicas de um objeto na imagem podem receber maior peso. Por fim, a terceira camada é a Totalmente Conectada. O objetivo é combinar as entradas de todas as operações para realizar a classificação dos objetos presentes na imagem.

Figura 13 – Arquitetura básica geralmente encontrada nas CNNs.



Fonte – Do autor com base em O'SHEA e NASH (2015).

Na Figura 13, tem-se a imagem de entrada processada de forma que cada região que compõe a imagem seja entrada da Camada Convolucional. Da mesma forma, cada imagem gerada nesta etapa servirá de entrada para a Camada *Pooling*, onde consequentemente são geradas imagens com menores resoluções. Sendo assim, a extração e geração de um mapa de características são concluídas, passando para a etapa de classificação com a Camada Totalmente Conectada.

A camada Convolucional contém filtros que são aplicados em todas as regiões da imagem de entrada, onde, para cada um desses filtros, um neurônio está conectado a um subconjunto de neurônios na camada anterior. O objetivo dos filtros é gerar características a partir de uma região da imagem, chamada de Campo Receptivo Local (CRL). As operações com filtros são executadas para detectar bordas, aplicar nitidez e suavização, mantendo-se a matriz de pixels da imagem e alterando os valores da matriz do filtro, isso faz com que

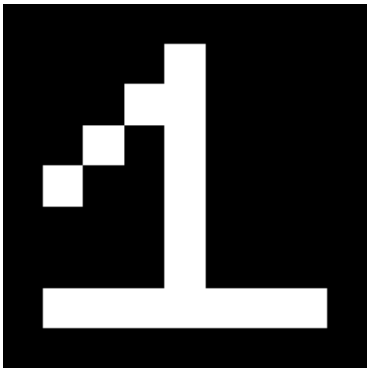
diferentes filtros possam destacar várias características em uma imagem (KARN, 2016). A Figura 14 apresenta alguns exemplos de filtros. Já a Figura 15, exemplifica uma imagem de entrada (15a) e sua matriz de pixels (15b).

Figura 14 – Exemplos de filtros.

Operação	**Filtro**	**Resultado**
Identidade	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Detecção de borda	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Nitidez	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Suavização	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Fonte – Do autor com base em KARN (2016).

Figura 15 – Imagem e sua matriz de pixels.



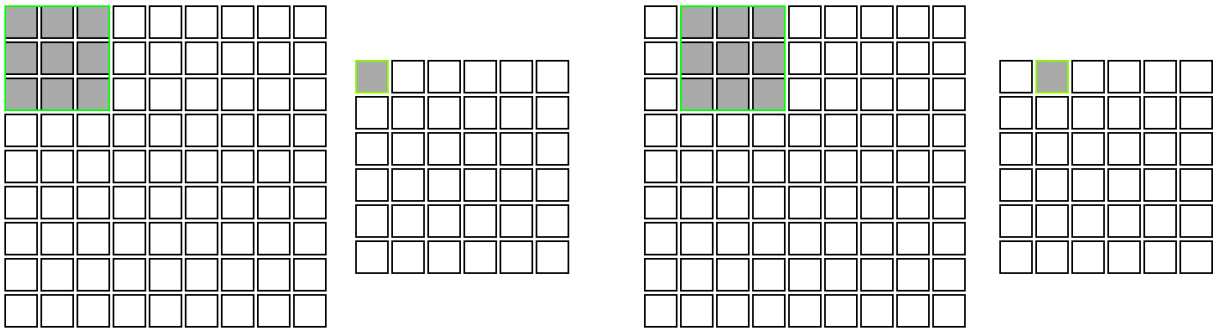
0	0	0	0	0	0	0	0	0
0	0	0	0	255	0	0	0	0
0	0	0	255	255	0	0	0	0
0	0	255	0	255	0	0	0	0
0	255	0	0	255	0	0	0	0
0	0	0	0	255	0	0	0	0
0	0	0	0	255	0	0	0	0
0	255	255	255	255	255	255	255	0
0	0	0	0	0	0	0	0	0

(a) Imagem (b) Matriz de pixels

Fonte – Do autor.

Cada região da imagem processada por um filtro, é denominada de Campo Receptivo Local (CRL). Um pixel de saída é o resultado de uma operação dos pixels de entrada com este CRL. Vale ressaltar que todos os CRL são filtrados com os mesmos pesos (AREL; ROSE; KARNOWSKI, 2010). A Figura 16 apresenta a aplicação de um fitro em um CRL. Ainda na camada Convolucional, existem pesos compartilhados entre os neurônios afim de indicar aos filtros padrões frequentes em qualquer região da imagem de entrada. É através dos pesos compartilhados que as CNNs conseguem detectar diferentes representações de um padrão (HAFEMANN, 2014).

Figura 16 – Exemplo da aplicação de um filtro em Campo Receptivo Local (3x3 pixels) contido em uma imagem (9 x 9 pixels).



(a) (b)

Fonte – Do autor com base em O'SHEA e NASH (2015).

A camada *Pooling* visa reduzir gradativamente a dimensão das representações da CNN, dessa forma, diminui ainda mais o número de parâmetros e principalmente a complexidade computacional do modelo (GOODFELLOW; BENGIO; COURVILLE,

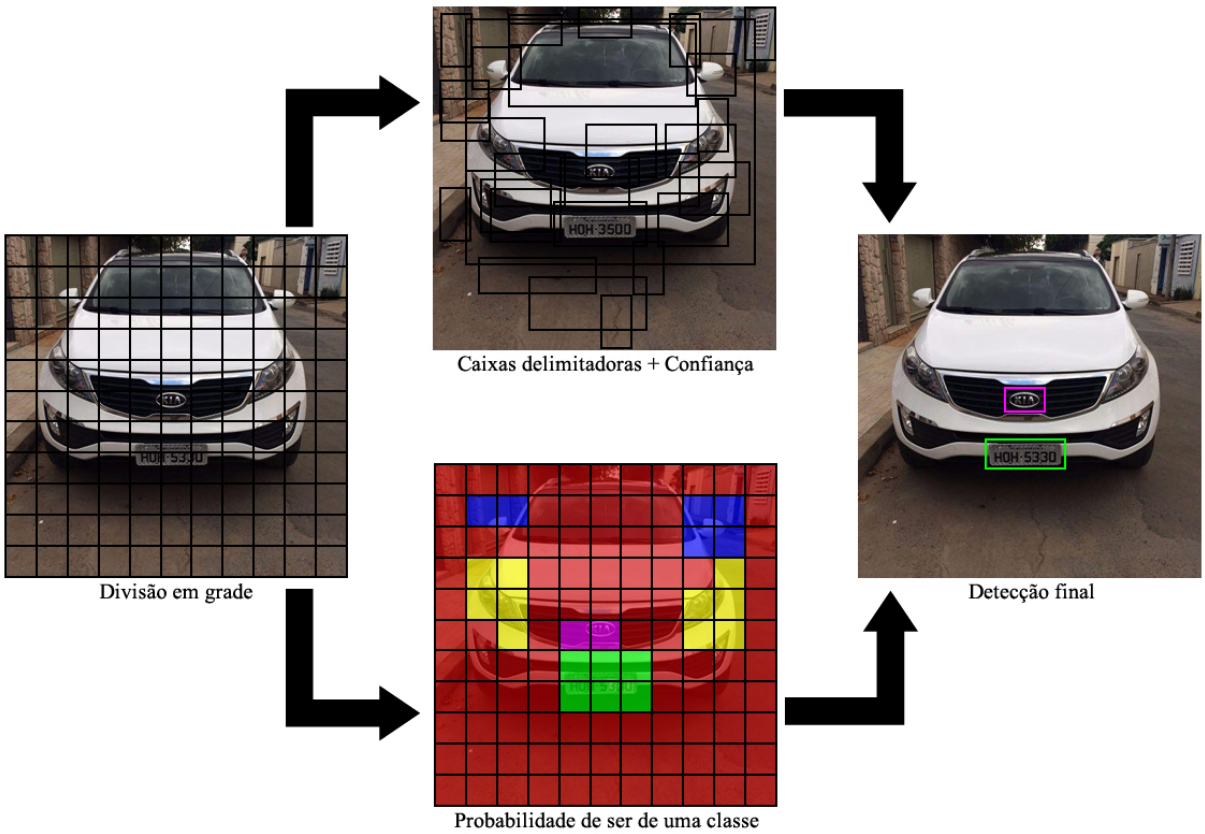
2016). Esta camada atua sobre cada mapa de características gerada anteriormente. A redução da dimensionalidade geralmente ocorre usando a função de máximo (Max). Na maioria das CNNs, essa redução encontra-se em camadas de *Max-pooling*, sendo capaz de redimensionar o mapa de características para 25% do tamanho original (O'SHEA; NASH, 2015).

A camada Totalmente Conectada está presente nas redes neurais tradicionais (HAFEMANN, 2014). Ela contém conexões com todos os neurônios da camada anterior e estão conectados a todos os neurônios da camada seguinte. A saída das camadas Convolucionais e *Pooling* representam características de alto nível da imagem de entrada, sendo assim, o objetivo da camada Totalmente Conectada é utilizar essas características para a classificação da imagem de acordo com as classes do conjunto de treinamento (KARN, 2016).

2.4 YOLO

You Only Look Once (YOLO) é um algoritmo para detecção de objetos voltado para processamento em tempo real, publicado em 2016 (REDMON et al., 2016). Segundo os autores Redmon et al. (2016), o algoritmo baseia-se na arquitetura das Redes Neurais Convolucionais, e possui grande capacidade de generalização, dessa forma, faz com o que se tenha bons resultados para detecção de vários objetos. Para otimizar o processamento e aumentar a quantidade de *frames* por segundo (FPS), sua arquitetura possui uma rede convolucional para prever simultaneamente várias regiões da imagem, considerando as probabilidades de ser um elemento de determinada classe. A Figura 17 mostra o modelo que o YOLO divide uma imagem em grade $S\ x\ S$ e para cada célula, destaca B caixas delimitadoras atribuindo um valor de confiança e a probabilidade do objeto ser de uma determinada classe.

Figura 17 – O Modelo.



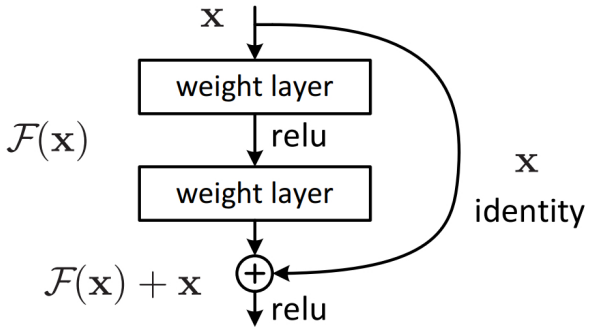
Fonte – Do autor com base em Redmon et al. (2016).

Em 2017, foi lançada a versão 2 do YOLO com melhorias de desempenho, passando a utilizar um modelo totalmente convolucional. Além disso, o YOLOv2 também passou a prever a altura e largura das caixas delimitadoras, tornando a detecção invariante às dimensões do objeto. Segundo Redmon e Farhadi, considerando imagens com 416 x 416 pixels, o YOLOv2 atingiu 67 FPS, 22 FPS a mais em relação a primeira versão do algoritmo, sendo 13,4% superior em termos de acurácia. Junto com versão 2, foi publicado um modelo pré-treinado chamado de YOLO9000, contendo 9 mil categorias de diferentes objetos, prontas para prever e detectar classes que não tenham dados rotulados. O YOLOv2 foi utilizado por vários trabalhos encontrados na literatura, com objetivo de detectar placas veiculares (LAROCA et al., 2018) (SILVA; JUNG, 2018) (HSU et al., 2017). Com o bom desempenho já anunciado por Redmon e Farhadi, todos esses trabalhos comprovaram ótimos resultados ao detectar placas veiculares.

Este ano (2018b), o YOLO foi atualizado para a versão 3. Além de melhorias de desempenho, a recente versão apresenta um novo método para extração de características. Segundo Redmon e Farhadi, a nova rede neural convolucional é uma proposta híbrida entre a rede utilizada no YOLOv2, e a Darknet-19 (REDMON; FARHADI, 2018a). A proposta apresenta o Darknet-53, algoritmo que utiliza camadas de filtros 3 x 3 e 1 x 1,

além de conexões de atalho encontradas nas *Residual Networks* ResNets (HE et al., 2016), para a extração de características, substituindo a Darknet-19. A Figura 18 apresenta um diagrama de como as conexões de atalho funcionam: realizam o mapeamento da identidade da matriz de imagem, de forma que suas saídas são adicionadas diretamente às saídas de outras camadas.

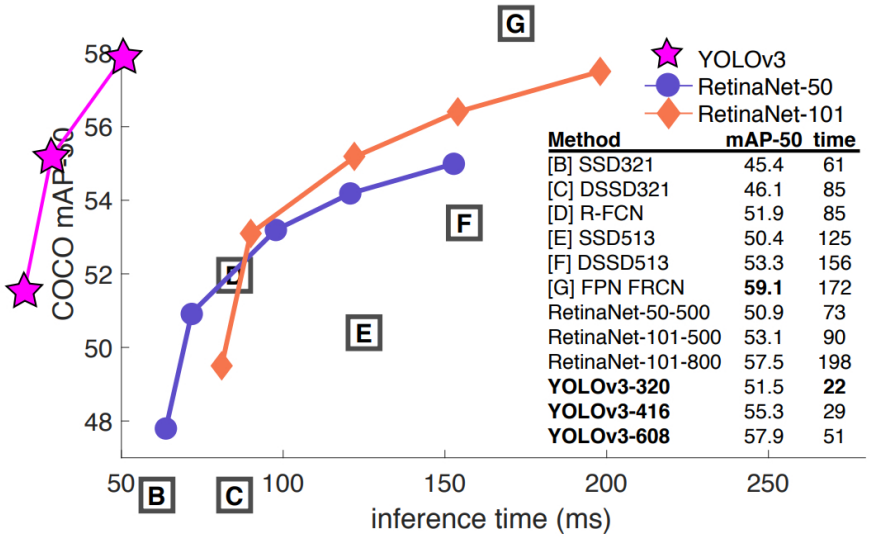
Figura 18 – Conexões de atalho.



Fonte – He et al. (2016).

Em termos de desempenho, o YOLOv3 se destaca pela velocidade de detecção. De acordo com o trabalho Redmon e Farhadi (2018b), além do aumento na capacidade de detectar objetos pequenos, a versão 3 chega a ser cerca de três vezes mais rápida em relação ao algoritmo Single Shot Multibox Detector - SSD. Dessa forma, o YOLOv3 continua sendo especialista na execução em tempo real. A Figura 19 apresenta um gráfico de acurácia x velocidade. Pode-se perceber o quão rápido é o tempo de execução possui o YOLOv3, considerando IoU de 50%.

Figura 19 – Comparação desempenho da YOLOv3 em termos de acurácia x velocidade.



Fonte – Redmon e Farhadi (2018b).

2.4.1 Hiperparâmetros

O Yolo contém hiperparâmetros que podem ser ajustados para melhorar a detecção de objetos de acordo com o problema abordado. A taxa de aprendizagem é um hiperparâmetro que define a rapidez com que a rede neural atualiza seus parâmetros no processo de aprendizagem. Um baixo valor para a taxa de aprendizagem aumenta o tempo do processo, mas obtém resultados mais fieis. Por outro lado, um valor alto diminui o tempo do processo de aprendizagem, mas não apresenta resultados com tanta fidelidade. O hiperparâmetro que define o número de épcoas, é usado para definir quantas iterações o algoritmo irá executar a operação de treinamento por completo. Da mesma forma, é necessário escolher o valor do hiperpâmetro que define a quantidade de exemplos treinados por rodada de uma época, conhecido como *batch size*. Este hiperparâmetro faz com que um subconjunto do conjunto de imagens de treinamento seja processado por vez. O *batch size* deve ser ajustado levando em consideração o espaço de memória disponível no computador. Outro hiperparâmetro presente no YOLO é o Limiar *Non-Maximum Suppression* (NMS). O NMS é utilizado na detecção de objetos em cenários em que se há possibilidade de existir vários objetos na cena, ou seja, através de um limiar, é possível controlar diferentes regiões candidatas para que um objeto não seja detectado mais de uma vez. A Figura 20 apresenta a comparação entre uma detecção sem e outra com NMS aplicado.

Figura 20 – *Non-Maximum Suppression*

(a) Sem NMS (b) Com NMS

Fonte – Do autor com base em Quora (2018)

Existem mais hiperparâmetros que podem ser configurados para melhorar o resultado da detecção. Alterar o valor da penalidade de uma predição não-objeto erroneamente,

faz com que a rede neural receba uma determinada penalidade por não classificar uma região que possui um objeto. Além disso, existe um hiperparâmetro que define um valor de penalidade para a predição de objetos errados. Este valor indica quanto uma rede neural será penalizada ao classificar de forma errada, uma região que não possui objeto. Ainda sim, o valor da penalidade de uma predição de um objeto com coordenadas erradas é o hiperparâmetro que define quão o algoritmo será penalizado em errar as coordenadas na predição de um objeto.

2.5 Otimização de Hiperparâmetros

Otimização de hiperparâmetros é uma técnica que visa otimizar o resultado de uma função combinando um conjunto de variáveis e valores pré-definidos, o qual é chamado de espaço de configurações (BERGSTRA et al., 2011). Esses espaços de configurações podem ser estruturados de várias formas. Uma das formas é a estrutura em árvore, onde algumas variáveis podem ser definidas como folhas, enquanto outras podem ser definidas como nós. Um algoritmo de otimização de hiperparâmetros deve otimizar, não somente valores de variáveis (sendo discretas, ordinais ou contínuas), mas também deve selecionar quais variáveis irão ser otimizadas. De acordo com Bergstra, Yamins e Cox (2012), muitos algoritmos de visão computacional dependem de hiperparâmetros, como por exemplo, a quantidade de filtros, níveis de quantização e taxa de aprendizagem. Essas escolhas podem influenciar diretamente no desempenho do sistema.

O TPE (BERGSTRA; YAMINS; COX, 2013) é um algoritmo de otimização de hiperparâmetro que atua substituindo nós escolhidos seguindo o modelo de Processos Gaussianos (GP) (RASMUSSEN; WILLIAMS, 2006). A partir de um conjunto de valores e variáveis a serem experimentados, em cada iteração, o TPE aplica uma função do modelo GP ($l(x)$) ao conjunto dos menores valores já obtidos em iterações anteriores, ou seja, os melhores encontrados até o momento. Além disso, outro GP ($g(x)$) é aplicado aos demais valores do conjunto. Por fim, o TPE escolhe o valor x que maximiza o resultado da operação $l(x)/g(x)$. A condição de parada é quando o TPE atingir o número máximo de iterações ou quando não houver mais variações em seus resultados. Sendo assim, a utilização do TPE para o problema de otimização de hiperparâmetros faz com que se consiga experimentar várias combinações de valores e variáveis de algoritmos, como redes neurais convolucionais, afim de saber quais são os valores que fornecerão os melhores resultados. Como por exemplo, o TPE poderia experimentar e combinar valores em termos de taxa de aprendizagem, número de épocas, quantidade de exemplos treinados por etapa do treinamento, etc; para potencializar o treinamento do algoritmo.

3 Trabalhos Relacionados

As placas veiculares (PV) são de grande importância para o rastreamento e identificação dos veículos. Elas representam dados do veículo como: marca, modelo, ano, cor, dados do proprietário, entre outros. Com isso, é possível realizar a fiscalização de motoristas infratores, veículos roubados, bem como o controle de acesso à áreas restritas (RIBEIRO et al., 2017). Na literatura é possível encontrar diversos métodos e estudos que buscam resolver de alguma forma o problema de localização de PV em imagens. Isso resulta em experimentos de algoritmos com padrões de placas diferentes em vários países. Sendo assim, foi realizada uma revisão bibliográfica para conhecer quais técnicas estão sendo utilizadas para a atividade de localização de PV. Em um período de cerca de 10 anos, foram analisadas diversas técnicas abordadas, e verificou-se para onde a tendência das pesquisas estão direcionadas em termos de técnicas e algoritmos como: processamento de imagens, aprendizagem de máquina e aprendizagem profunda.

Al-Ghaili et al. (2013) apresentaram uma proposta para localizar placas veiculares com objetivo de ser executado em tempo real. Um dos desafios é localizar placas em imagens de baixa resolução, extraídas através de um *webcam*. Para amenizar a alta taxa de ruído, o trabalho apresenta ótimas máscaras morfológicas. Por utilizar um *webcam*, as imagens de entrada possuem resolução fixa (352x288), restringindo o funcionamento do algoritmo. Além disso, o carro precisa estar em uma distância específica e entre um ângulo (mais ou menos 20 graus de rotação) da câmera. A proposta não considerou os padrões de placas brasileiras, não sendo possível analisar os resultados para este contexto.

Yazdian et al. (2014) propuseram um método robusto para o problema de reconhecimento de placas veiculares de Ontário com base na técnica de compensação de iluminação, em conjunto com um algoritmo de reconhecimento de caracteres. A estratégica adotada foi utilizar técnicas de compensação de iluminação para normalizar luzes, reflexos e sombras presentes no ambiente que podem interferir na extração de características. De acordo com o trabalho, não é possível encontrar mais detalhes sobre as imagens utilizadas na base de dados, apenas a resolução (320x240) e 30 frames por segundo foram apresentados. Dessa forma, não é possível verificar resultados de testes com mais variedades de iluminação, ângulos e *backgrounds* das imagens.

Prates et al. (2014) desenvolveram um método com objetivo de reconhecer placas com padrões de diferentes países. Para isto, utilizaram a técnica de janela deslizante avaliando o Histograma de Gradientes Orientados (HOG) em cada região extraída. HOG, de acordo com a literatura, é uma das técnicas que apresenta um bom funcionamento para classificação de objetos em imagens (ZHANG et al., 2008). A proposta apresentou bons

resultados no desafio de detecção de placas utilizando HOG em conjunto com o algoritmo SVM, com capacidade para detectar placas com diversos padrões, inclusive de padrões brasileiros, como as placas particulares e para transportes. Contudo, o método foi avaliado com um baixo número de imagens que compõe a base (377 imagens), e que apenas 20% dessas imagens foram utilizadas para testes; além das imagens terem resoluções fixas, novas imagens de entradas com diferentes resoluções e condições de iluminação podem dificultar a localização da placa, o que pode reduzir a efetividade.

Cavalcanti et al. (2015) apresentaram um método de reconhecimento de placas veiculares para a gestão de acesso à áreas restritas. O objetivo é reconhecer placas veiculares através de vídeos de câmeras localizadas em estabelecimentos. O método apresentado utiliza 4 técnicas de processamento de imagens: transformada de Hough, operador de gradiente, *Canny* e limiarização para reduzir problemas ou elementos nas imagens que podem diminuir a eficiência do algoritmo. O experimento levou em consideração apenas imagens de câmeras estáticas, ou seja, câmeras fixas em determinados locais do estabelecimento e com resolução fixa. Além disso, o tamanho da placa e dos caracteres também são estáticos, ou seja, inalterados para todas as imagens de entrada. Dessa forma, esse trabalho apresenta uma boa combinação de técnicas de processamento, porém, em imagens com resoluções, posições e condições de iluminação diferentes dos valores fixados, podem ter a detecção da placa afetada.

Sagharichi e Shakeri (2016) propuseram um método para localização e reconhecimento de caracteres numéricos em placas de automóveis. Uma estratégia para que a execução do algoritmo seja em tempo real, é remover o *background* através da subtração de imagens, afim de obter a interseção dos pixels. Esta estratégia reduz significantemente a quantidade de ruídos e o espaço de busca na imagem, elevando a acurácia. Em contrapartida, é necessária a extração de um quadro do vídeo, que não contenha veículo na imagem, para que a remoção do *background* seja feita. Além disso, é preciso que a câmera seja fixa, de modo que o quadro de remoção sempre coincida com o restante dos quadros do vídeo. Sendo assim, o trabalho não levou em consideração a execução da proposta em ambientes complexos, mas sim, em ambientes simples - cenários que se repetem ao longo das imagens contidas na base de dados.

Gonçalves et al. (2016) apresentaram uma proposta baseada em processamento de imagens e aprendizagem de máquina. O trabalho utiliza o Histograma de Gradientes Orientados (HOG) e SVM para localizar e reconhecer a placa em tempo real. Aliado a isso, a utilização da técnica de redundância dos dados forneceu um aumento da acurácia. A proposta tenta localizar o carro na imagem reduzindo o espaço de busca para a detecção da placa. O trabalho consegue ser executado em tempo real, localizando mais de uma placa na mesma imagem. O padrão de placas adotado foi o brasileiro, porém não foi especificado se a base contém outros subpadrões, como por exemplo, as placas de transporte. O

método apresentado utiliza redundância de dados, o que pode requerer maiores recursos computacionais para que seja possível ser executado em tempo real. O Trabalho também utiliza imagens de alta resolução fixada em (1920x1080) e câmera estática, o que pode comprometer a acurácia quando executado em uma base com diferentes resoluções e maior variação de ambientes.

Yuan et al. (2016) apresentaram uma proposta, com foco na eficiência, para localizar placas veiculares em imagens com ambientes complexos. Para que o algoritmo fosse executado em tempo real, reduziram as dimensões da imagem e, com base em experimentos de desempenho, utilizaram o filtro Sobel para detecção de bordas. Por fim, foi utilizado o algoritmo SVM para selecionar a região da placa. Alguns parâmetros apresentados, como em filtros de processamento e dimensões da imagem de entrada, foram definidos de forma estática. Dessa forma, assim como no trabalho anterior, imagens que possuam um cenário ou condição de iluminação específicos, podem ter afetar a precisão de detecção da placa.

Diante de algumas propostas, como as estudadas e apresentadas nos parágrafos acima, percebe-se a necessidade de um método que possua maior capacidade de generalização e consiga localizar placas veiculares em ambientes complexos, com menor ou nenhuma dependência de parâmetros. Os trabalhos a seguir utilizam técnicas de aprendizagem profunda e conseguiram obter ótimos resultados aplicando-se em padrões de placas diferentes.

Montazzolli e Jung (2017) apresentaram uma proposta para detectar placas veiculares com o padrão do Brasil utilizando Redes Neurais Convolutivas. Como meta, determinaram que a execução deveria ser concluída em tempo real. Para isso, utilizaram uma técnica de reconhecimento facial para detectar a frente do veículo, diminuindo a região de busca na imagem. Além de localizar placas nas imagens, o trabalho também apresenta um método para reconhecer os caracteres contidos na placa. Contudo, a proposta utiliza imagens com resoluções estáticas, em *Full-HD*, sendo assim não é possível saber como o algoritmo se comporta com imagens de outras resoluções. Além disso, o trabalho levou em consideração imagens onde os veículos aparecem de frente, não sendo possível observar o resultado em imagens das traseiras dos veículos. Outro ponto que pode afetar o resultado do método, é a manter uma proporção fixa para a placa. O trabalho utiliza uma proporção de cerca de 19 vezes menor do que a resolução da imagem de entrada. Valores fixos podem limitar a eficácia do algoritmo quando a imagem de entrada possuir resolução diferente.

Xie et al. (2018) apresentaram uma proposta com uma versão de CNN que possui alta acurácia na detecção de placas. Além de ter foco na acurácia, a proposta consegue ser executada em tempo real. O trabalho apresenta, também, uma correção na inclinação da placa para facilitar a detecção dos caracteres. A proposta utiliza uma base de placas do padrão taiwanês, com 2049 imagens coletadas de câmeras em estacionamentos. Sendo assim,

os experimentos foram realizados utilizando imagens com poucas variações de posições, ângulos e ambientes. Portanto, não se sabe o resultado do algoritmo proposto, em um banco de imagens mais heterogêneo. Além disso, seria necessário realizar um experimento com uma base de imagens de placas com padrão brasileiro, para saber se a acurácia se mantém.

Laroca et al. (2018) utilizaram a CNN implementada no YOLOv2 para treinar e detectar imagens de placas brasileiras encontradas na base OpenALPR (OpenALPR Cloud API, 2018) Além disso, construíram uma nova base de imagens que contenha problemas como variação de posições, iluminação e *background*. A proposta também foi elaborada para que o algoritmo forneça o resultado em tempo real, de modo que seja possível detectar placas em vídeos. Este trabalho apresenta capacidade para detectar o padrão de placas particulares e de transporte, sendo possível também, detectar placas de motocicletas. Por utilizar a versão 2 do YOLO, não se sabe o resultado da aplicação do algoritmo utilizando o YOLOv3, versão mais recente.

De acordo com a tendência de pesquisa no decorrer dos anos e pela grande capacidade de generalização e eficiência nos resultados apresentados, este trabalho foi inspirado para a utilização de algoritmos de aprendizagem profunda. Um dos algoritmos que se destacou na atividade de localização de placas veiculares, foi o YOLOv2. Com recentes trabalhos (XIE et al., 2018) (LAROCA et al., 2018) em que se alcançaram resultados de até 93% de acurácia utilizando 60% de IoU e pela capacidade de ser executada em tempo real, o YOLO foi estudado com maior dedicação. Por fim, verificou-se que neste ano (2018b), foi lançada a terceira versão do YOLO (REDMON; FARHADI, 2018b). Até este trabalho, no melhor do nosso conhecimento, essa versão ainda não foi utilizada para a localização de placas veiculares com o padrão brasileiro.

Diante das propostas dos trabalhos apresentados, este projeto se diferencia pela análise dos algoritmos baseados em aprendizagem profunda YOLOv2 e YOLOv3, seguindo a tendência de pesquisa, executados com um banco de imagens criada em função de grande variações de ângulos, posições, resoluções e ambientes, para o problema de localização de placas veiculares brasileiras. É importante ressaltar que a versão 3 do YOLO, não foi utilizada para a determinada atividade, no melhor do nosso conhecimento. Além disso, foi utilizado o algoritmo TPE para a otimização de hiperparâmetros das redes neurais convolucionais.

4 Detecção da Placa

A localização e detecção de objetos usando algoritmos basados em aprendizagem profunda reduz a necessidade de serem implementadas etapas como a aplicação de filtros de processamento de imagens e segmentação de objetos específicos. Dessa forma, esses algoritmos fornecem grande capacidade de generalização podendo ser aplicados em vários outros problemas alterando apenas a etapa de treinamento. Além disso, é possível potencializar os resultados com a otimização de hiperparâmetros das redes neurais convolucionais utilizadas.

Este trabalho utiliza os algoritmos de aprendizagem profunda YOLOv2 e YOLOv3 para detectar placas veiculares (PV) em imagens com cenários de ambientes complexos. Antes de detectar PV em imagens com esses algoritmos, é necessário executar um treinamento com um conjunto de imagens de exemplo. Para isto, também é necessário definir os valores dos hiperparâmetros para que o treinamento obtenha o melhor desempenho possível. Afim de maximizar o desempenho, este trabalho adotou um conjunto de hiperparâmetros que são encontrados em trabalhos relacionados na literatura. Além disso, este trabalho utilizou o algoritmo TPE para explorar, avaliar e indicar as melhores combinações de hiperparâmetros que potencializam o treinamento considerando o problema de localização de PV.

A Subseção 4.1 descreve como a otimização de hiperparâmetros foi aplicada e quais foram os hiperparâmetros adotados. A Subseção 4.2 descreve como o treinamento dos algoritmos é realizado, e por fim, a Subseção 4.3 descreve como a detecção da placa é realizada neste trabalho.

4.1 Otimização de Hiperparâmetros

Afim de explorar o vasto número de combinações de hiperparâmetros das redes neurais convolucionais utilizadas, este projeto utilizou o TPE em função do *Mean Average Precision* (mAP) para obter a indicação de uma boa combinação de hiperparâmetros. *Mean Average Precision* é uma métrica usada para avaliar o treinamento dos algoritmos, onde na escala de porcentagem, quanto maior for o valor de mAP melhor é o resultado do treinamento. Sendo assim, neste trabalho, uma boa combinação de hiperparâmetros é àquela em que o TPE maximiza o valor do mAP. Este trabalho considerou os seguintes hiperparâmetros como conjunto no espaço de busca para serem otimizados com o TPE de acordo com a utilização em trabalhos relacionados: i) Taxa de aprendizagem; ii) Limiar NMS (NMS); iii) Valor de penalidade de uma predição não-objeto erroneamente; iv) Valor de penalidade de uma predição de um objeto errado; e v) Valor de penalidade de uma

predição de um objeto com coordenadas erradas. Realizada a otimização, o TPE indica os valores de cada hiperparâmetro que maximizam o desempenho do treinamento.

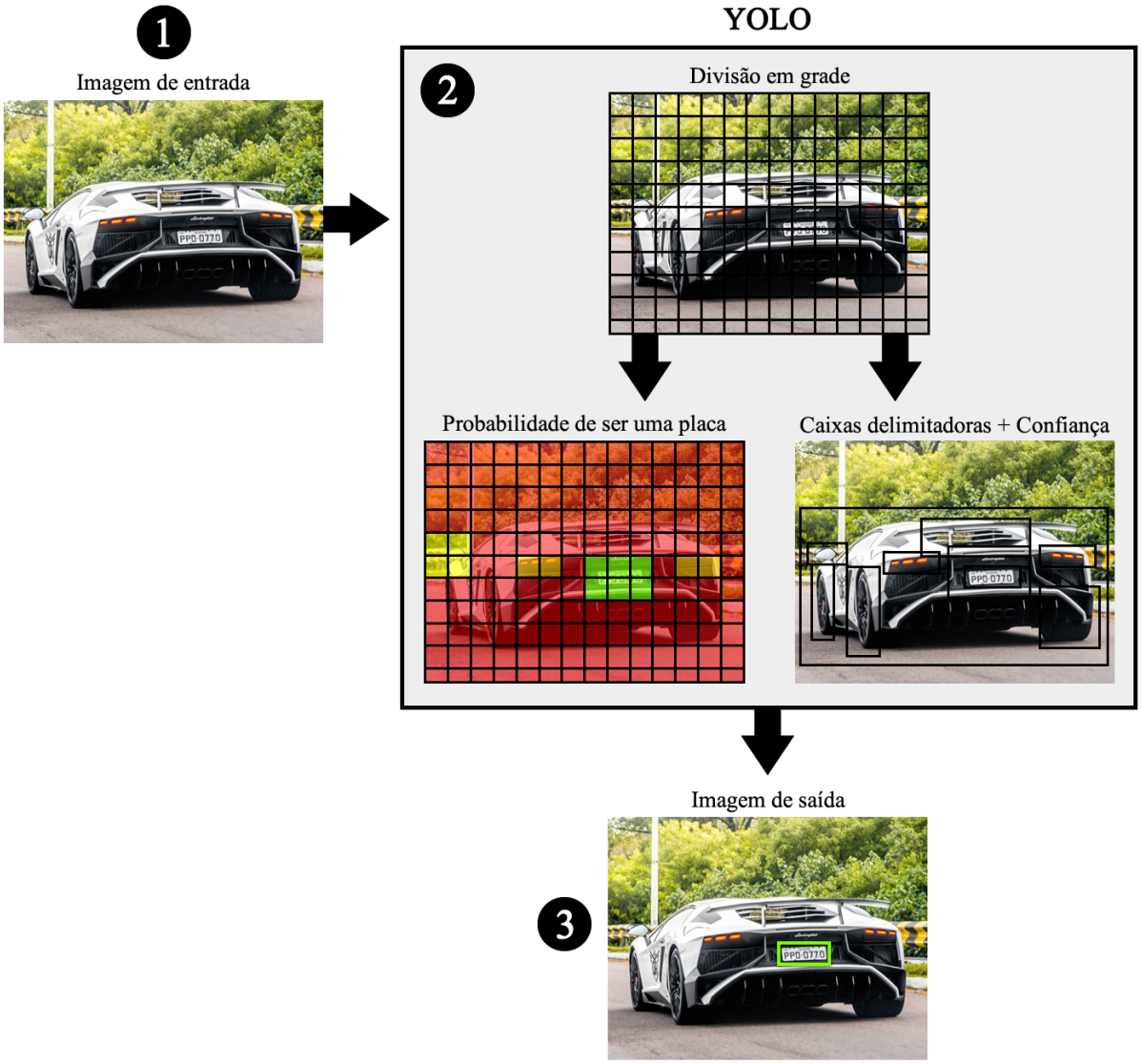
4.2 Treinamento dos algoritmos

O treinamento é uma etapa fundamental para que os algoritmos possam coletar dados sobre as características de uma placa veicular (PV). Esse processo faz com que os algoritmos aprendam a reconhecer uma PV. Após definir os hiperparâmetros do treinamento, foi fornecido um conjunto de imagens exemplares que contém PV e um conjunto de rótulos, o qual indica a posição correta da placa associada a cada respectiva imagem do conjunto de exemplo. Dessa forma, os algoritmos conseguem coletar automaticamente diversas características fornecidas na camada convolucional. Por fim, os algoritmos geram um modelo de treinamento contendo todo os dados sobre PV aprendidos durante a etapa, que pode ser usado para predizer e detectar PV em outras imagens.

4.3 Predição das placas

Predição é a etapa responsável por fazer os algoritmos processarem uma imagem de entrada e, com base no modelo de treinamento, tentarem detectar placas veiculares (PV). A Figura 21 apresenta um sequência de etapas de como os algoritmos YOLOv2 e YOLOv3 seguem para detectar PV neste trabalho. A partir de uma imagem de entrada, retirada de um conjunto de imagens para testes, os algoritmos dividem essa imagem em várias células, de modo semelhante a uma estrutura em grade. Em seguida, duas operações são realizadas. A primeira operação faz com que os algoritmos analisem cada célula da grade definindo o valor da probabilidade que essa célula tem de ser da classe "placa". Essa estimativa é realizada com base no modelo de treinamento. A outra operação faz com que os algoritmos demarquem a imagem em regiões chamadas de caixas delimitadoras. Essa demarcação é realizada de acordo com algoritmos para detecção de bordas e áreas retangulares. Após isso, os algoritmos YOLOv2 e YOLOv3 calculam a taxa de confiança, a qual define a probabilidade dessa região ser um objeto de interesse. Por fim, os algoritmos mesclam as células de maiores pontuações obtidas na primeira operação, com as regiões de maiores taxas de confiança. Finalmente, o resultado dessa mesclagem é a predição da posição da placa veicular na imagem.

Figura 21 – Etapas da detecção da placa.

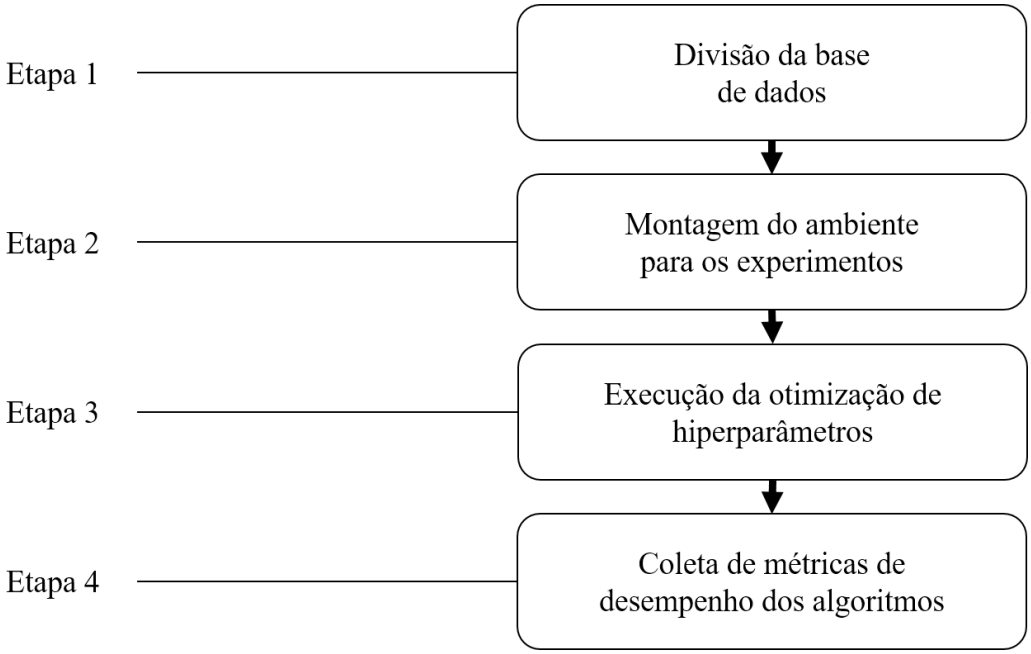


Fonte – Do autor.

4.4 Metodologia

Esta seção apresenta o método utilizado neste trabalho. Dessa forma, com todas as etapas sendo ordenadas e especificadas como mostra a Figura 22, é possível replicar os resultados atingidos.

Figura 22 – Metodologia.



Fonte – Do autor.

4.4.1 Divisão da base de dados

Entre o final de 2017 e começo de 2018 foi construída uma base de dados de placas veiculares brasileiras, com o objetivo de obter boa variedade de imagens em termos de problemas como: ângulo de perspectiva da cena, condições de iluminação, distância entre a placa e a câmera, nível de ruído e objetos que podem ser confundidos com placas veiculares. As imagens dessa base de dados foram coletadas em diversos sites públicos na internet. A Figura 23 apresenta recortes de algumas imagens coletadas, visto que elas possuem diferentes resoluções. No total são 400 imagens rotuladas manualmente, baseadas no formato *PASCAL Visual Object Classes* (VOC). De acordo com Everingham et al. (2010), VOC consiste em se ter um conjunto de dados de imagens e anotações de forma padronizada.

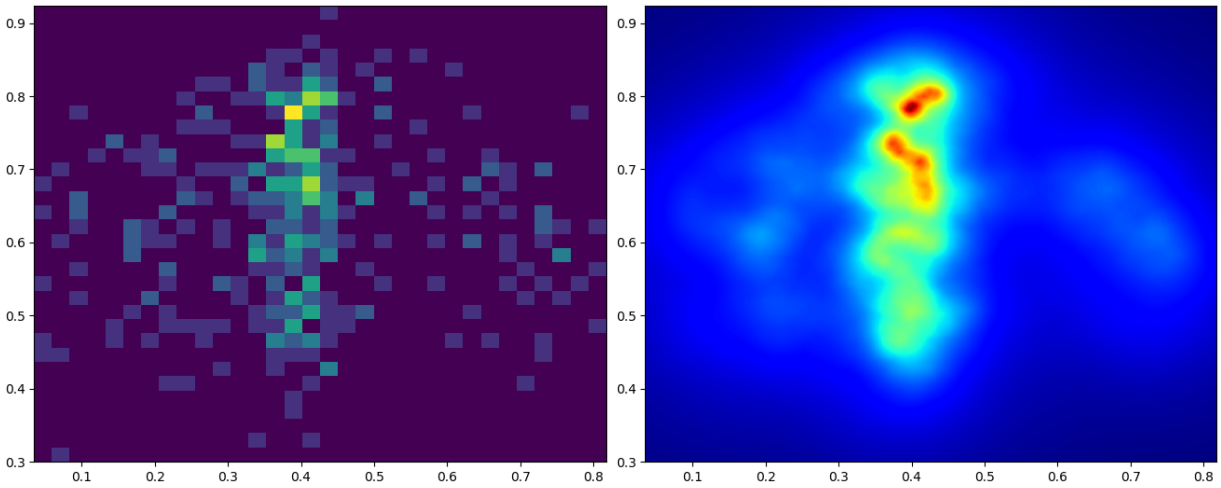
As 400 imagens da base de dados construída neste trabalho foram subdivididas aleatoriamente em subconjuntos com 160 imagens para treinamento, 80 imagens para validação e 160 imagens para testes. As resoluções das imagens são variadas, sendo a menor imagem contendo 269x480 pixels e a maior 2000x1824 pixels. A Figura 24 apresenta o mapa de calor em relação a posição normalizada das placas na base de dados. Este mapa representa a distribuição da posição das placas nas imagens coletadas. A normalização garante que todas as imagens sejam calculadas considerando uma resolução com valores entre 0 e 1, ou seja, 100x100 pixels, mas sem a necessidade de redimensioná-las. Dessa forma, pode-se perceber que há uma maior concentração de placas na região central das imagens da base de dados.

Figura 23 – Recortes de exemplo das imagens na base de dados.

Fonte – Do autor.

Para cada arquivo de imagem da base, existe um arquivo XML com o rótulo da placa descrito. Dessa forma, as imagens não precisam ser replicadas, pois todas as informações estão presentes em seu respectivo arquivo VOC. A Figura 25 apresenta um diagrama com a estrutura dos elementos VOC, onde neste projeto está escrita em XML. O elemento *Annotation* contém todos as informações sobre a imagem. *Path* representa o caminho do arquivo de imagem o qual o VOC está associado. *Size* armazena o tamanho da imagem em pixels (*Width* x *Height*). *Object* contém os dados sobre os objetos presentes na imagem. Nos casos da imagem conter mais de uma placa, por exemplo, no arquivo VOC referente iria existir a mesma quantidade de *Object*. *Name* representa o nome do objeto e *Bndbox* contém as coordenadas da área em que o objeto se encontra. *Xmin* é a coordenada do pixel X que inicia a área do objeto; *Ymin* é a coordenada do pixel Y que inicia a área do objeto; *Xmax* é o ponto limite da área do objeto no eixo X, se dá pela soma de *Xmin* com *Width*. Da mesma forma, *Ymax* é o ponto limite da área do objeto no eixo Y, se dá pela soma de *Ymin* com *Height*.

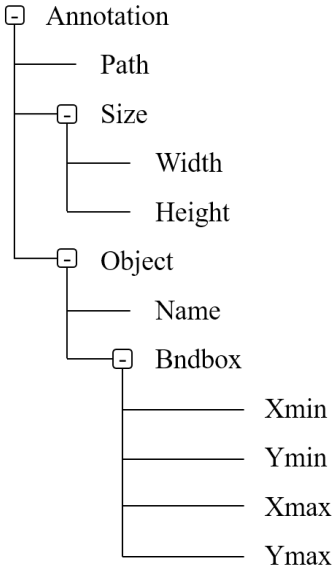
Figura 24 – Mapa de calor em relação a posição normalizada das placas na base de dados.



(a) (b)

Fonte – Do autor.

Figura 25 – Estrutura dos elementos VOC no arquivo XML.



Fonte – Do autor.

4.4.2 Montagem do ambiente para os experimentos

Com maior requerimento de recursos computacionais dos algoritmos YOLOv2 e YOLOv3, como a Unidade de Processamento Gráfico (GPU), este projeto utilizou a plataforma gratuita Google Colaboratory. O serviço *Software as a Service* (SaaS) fornece uma máquina por 12 horas contínuas, com Unidade de Processamento Central (CPU) Intel(R) Xeon(R) 2.20GHz Dual Core, 12GB de memória RAM e 11GB de GPU. Estes

recursos foram sufientes para executar os experimentos. Além disso, o Google Colaboratory fornece um ambiente com o Python 3 (Python Software Foundation, 2018) configurado com o Python Notebook (Jupytercon, 2018), OpenCV (OpenCV Team, 2018) e o TensorFlow (TensorFlow, 2018). Para finalizar a configuração do ambiente, foi necessário instalar manualmente as bibliotecas Keras (Keras, 2018) e Hyperopt (Hyperopt, 2018).

4.4.3 Execução da otimização de hiperparâmetros

Antes de executar os treinamentos dos algoritmos YOLOv2 e YOLOv3, foram executadas iterações do TPE com o objetivo de otimizar os hiperparâmetros, encontrando-se uma boa combinação entre os valores definidos no espaço de busca para os algoritmos. Neste projeto, os hiperparâmetros abordados foram: i) Taxa de aprendizagem; ii) Limiar NMS (NMS); iii) Valor de penalidade de uma predição não-objeto erroneamente; iv) Valor da penalidade de uma predição de um objeto errado; e v) Valor da penalidade de uma predição de um objeto com coordenadas erradas. Ainda há o valor da penalidade de uma predição errar a classe, porém, neste projeto há apenas uma classe: a placa veicular. O conjunto de valores possíveis no espaço de busca estão listados abaixo:

i) Taxa de aprendizagem: 0.0000001, 0.000001, 0.00001, 0.0001, 0.001 e 0.01.

ii) Limiar NMS: 0.70, 0.75, 0.80, 0.85, 0.90 e 0.95

iii) Valor de penalidade de uma predição não-objeto erroneamente: 1, 2, 5, 10 e 15

iv) Valor de penalidade de uma predição de um objeto errado: 1, 2, 5, 10 e 15

v) Valor de penalidade de uma predição de um objeto com coordenadas erradas: 1, 2, 5, 10 e 15

As execuções dos algoritmos durante os testes com o TPE foram realizadas utilizando as configurações descritas na Tabela 1. Ou seja, todas as iterações do TPE, os valores das configurações abaixo não foram alterados. É importante ressaltar que nesta etapa, foram utilizados os conjuntos de treinamento e validação da base de dados. O objetivo desta primeira rodada de execuções com o TPE é coletar conjuntos de hiperparâmetros que se destacam em relação ao Limiar descrito na Tabela 1.

Tabela 1 – Configurações dos algoritmos durante a primeira rodada de experimentos com hiperparâmetros

Configuração	**Valor**
Iterações por época	5
Épocas	3
Limiar	0.7

Após cerca de 60 iterações o TPE indicou quatro conjuntos de hiperparâmetros (Tabelas 2, 3, 4, 5), os quais apresentaram os maiores valores de mAP. A disponibilidade do serviço gratuito fornecido pelo Google Colaboratory limitou o número de iterações deste experimento, visto que as 60 iterações atingiram as 12 horas permitidas.

Tabela 2 – Conjunto 1 de hiperparâmetros indicados pelo TPE

Hiperparâmetro	**Valor**
Taxa de aprendizagem	0.0001
Limiar NMS	0.5
Valor de penalidade de uma predição de um objeto errado	15
Valor de penalidade de uma predição não-objeto erroneamente	1
Valor de penalidade de uma predição de um objeto com coordenadas erradas	10
mAP	**0.82**

Tabela 3 – Conjunto 2 de hiperparâmetros indicados pelo TPE

Hiperparâmetro	**Valor**
Taxa de aprendizagem	0.0001
Limiar NMS	0.7
Valor de penalidade de uma predição de um objeto errado	10
Valor de penalidade de uma predição não-objeto erroneamente	1
Valor de penalidade de uma predição de um objeto com coordenadas erradas	10
mAP	**0.71**

Tabela 4 – Conjunto 3 de hiperparâmetros indicados pelo TPE

Hiperparâmetro	**Valor**
Taxa de aprendizagem	0.0001
Limiar NMS	0.7
Valor de penalidade de uma predição de um objeto errado	15
Valor de penalidade de uma predição não-objeto erroneamente	1
Valor de penalidade de uma predição de um objeto com coordenadas erradas	2
mAP	**0.83**

Tabela 5 – Conjunto 4 de hiperparâmetros indicados pelo TPE

Hiperparâmetro	**Valor**
Taxa de aprendizagem	0.001
Limiar NMS	0.8
Valor de penalidade de uma predição de um objeto errado	5
Valor de penalidade de uma predição não-objeto erroneamente	1
Valor de penalidade de uma predição de um objeto com coordenadas erradas	2
mAP	**0.87**

Após estes testes, houve mais uma rodada de experimentos em relação aos hiperparâmetros. O objetivo dessa segunda rodada é obter o melhor conjunto de hiperparâmetros dentre os já selecionados. Dessa forma, não foi mais preciso utilizar o TPE. As configurações para essa rodada foram definidas de acordo com a Tabela 6. A Tabela 7 apresenta os resultados obtidos no treinamento e validação de cada conjunto. Como o conjunto 1 (Tabela 2) retornou o maior valor de mAP, este foi selecionado como o conjunto a ser utilizado na etapa de treinamento dos algoritmos YOLOv2 e YOLOv3 juntamente com o conjunto de hiperparâmetros encontrados em trabalhos na literatura.

Tabela 6 – Configurações dos algoritmos durante a segunda rodada de experimentos com hiperparâmetros

Configuração	**Valor**
Iterações por época	5
Épocas	6
Limiar	0.5

Tabela 7 – Resultado da segunda rodada de experimentos com hiperparâmetros

Conjunto	**Tabela**	**mAP**
1	**2**	**0.92**
2	3	0.83
3	4	0.77
4	5	0.84

4.4.4 Coleta de métricas de desempenho dos algoritmos

Neste experimento foram executados os treinamentos de forma individualmente, tanto para o YOLOv2 como para o YOLOv3, utilizando os hiperparâmetros encontrados em trabalhos na literatura e os hiperparâmetros indicados pelo TPE. Dessa forma, totaliza-se quatro treinamentos. A princípio o YOLOv2 foi treinado com o conjunto de hiperparâmetros encontrados em trabalhos na literatura (Tabela 8). Estes serão abreviados como CPAPER. Em seguida e da mesma forma, utilizando os hiperparâmetros CPAPER, o YOLOv3 foi treinado. O terceiro treinamento foi novamente do YOLOv2, porém utilizando o conjunto de hiperparâmetros indicados pelo TPE. Estes serão abreviados como CTPE. Por fim, o mesmo conjunto de hiperparâmetros CTPE foi utilizado para o treinamento do YOLOv3. Vale ressaltar que nesta fase, os treinamentos utilizaram o mesmo conjunto de treinamento e validação da base de dados. As configurações fixas utilizadas em cada treinamento estão descritas na Tabela 9. Após o treinamento, os algoritmos retornaram os resultados da validação. A Tabela 10 apresenta o valor de mAP resultante de cada um dos quatro treinamentos, onde o treinamento 4 com o YOLOv3 e CTPE apresentou o maior valor de mAP.

Tabela 8 – Conjunto de hiperparâmetros encontrados em trabalhos na literatura

Hiperparâmetro	**Valor**
Taxa de aprendizagem	0.0001
Limiar NMS	0.5
Valor de penalidade de uma predição de um objeto errado	5
Valor de penalidade de uma predição não-objeto erroneamente	1
Valor de penalidade de uma predição de um objeto com coordenadas erradas	5

Tabela 9 – Configurações dos algoritmos para os treinamentos

Configuração	**Valor**
Iterações por época	5
Épocas	50
Limiar	0.5

Tabela 10 – Valor de mAP resultante de cada treinamento

Treinamento	**YOLO**	**Hiperparâmetros**	**mAP**
1	v2	CPAPER	0.87
2	v3	CPAPER	0.95
3	v2	CTPE	0.86
4	**v3**	**CTPE**	**0.97**

A segunda fase dos experimentos foi a execução dos testes dos algoritmos e coleta das métricas de desempenho: i) Tempo de predição em segundos; ii) IoU; e iii) Taxa de confiança. É importante destacar que todos os testes utilizaram o conjunto de testes da base de dados. A Tabela 11 apresenta a ordem das execuções dos testes de modo que foi feita uma única execução para a coleta das métricas. Primeiro foi executado o teste com o YOLOv2 utilizando o modelo do treinamento com os hiperparâmetros CPAPER, encontrados em trabalhos na literatura. Da mesma forma, em seguida o teste foi executado com o YOLOv3 e os hiperparâmetros CPAPER. Logo após houve a execução do teste com o YOLOv2 e os hiperparâmetros CTPE, selecionados na etapa 4.4.3. Por fim, foi realizado o teste com o YOLOv3 e também com os hiperparâmetros CTPE.

Tabela 11 – Ordem das execuções dos testes

Teste	**YOLO**	**Modelo gerado com os hiperparâmetros**
1	v2	CPAPER
2	v3	CPAPER
3	v2	CTPE
4	v3	CTPE

Todas as métricas foram coletadas em cada predição das respectivas imagens do conjunto de testes da base de dados. No próximo capítulo (5) todos os resultados serão apresentados e discutidos.

5 Resultados

Este capítulo apresenta os resultados coletados durante os experimentos deste trabalho e será dividido em quatro seções: um para cada experimento. As seções a seguir apresentam os valores da média, mediana e desvio padrão das métricas de desempenho obtidas: i) Tempo de predição em segundos; ii) IoU; e iii) Taxa de confiança. Além disso, este capítulo apresenta uma discussão em cima dos resultados obtidos utilizando os algoritmos de aprendizagem profunda YOLOv2 e YOLOv3, e a influência da otimização de hiperparâmetros usando o TPE.

5.1 Experimento 1

O Experimento 1 está relacionado com a predição das imagens do subconjunto para testes definido na base de dados. Este experimento utilizou o algoritmo YOLOv2 considerando os hiperparâmetros encontrados em trabalhos na literatura (CPAPER), conforme apresentado no Teste 1 da Tabela 11. Executando o teste de predição, foram coletados os valores das métricas de desempenho que constam na Tabela 12. A Tabela 13 apresenta a quantidade de placas detectadas e não detectadas neste experimento. Com um tempo médio de 0.065 segundos, IoU médio de 71.2% e taxa de confiança 73.5%, o YOLOv2 com CPAPER detectou 151 das 160 placas, ou seja, 94.3% das imagens da base de dados.

Tabela 12 – Resultados do experimento 1

Métrica	**Média**	**Mediana**	**Desvio Padrão**
Tempo de predição (s)	0.0656	0.0650	0.0038
IoU (%)	71.2	76.7	21.2
Confiança (%)	73.5	78.4	20.4

Tabela 13 – Detecções no experimento 1

Placas encontradas	**Placas não encontradas**
151	9

5.2 Experimento 2

O experimento 2 está relacionado com a predição das imagens do subconjunto para testes definido na base de dados. Este experimento utilizou o algoritmo YOLOv3 considerando os hiperparâmetros encontrados em trabalhos na literatura (CPAPER), de acordo com o Teste 2 na Tabela 11. Executando o teste de predição, foram coletados os

valores das métricas de desempenho que constam na Tabela 14. A Tabela 15 mostra a quantidade de placas detectadas e não detectadas neste experimento. Com um tempo médio de 0.119 segundos, IoU médio de 82.2% e taxa de confiança 97.2%, o YOLOv3 com CPAPER detectou 159 das 160 placas, ou seja, 99.3% das imagens da base de dados.

Tabela 14 – Resultados do experimento 2

Métrica	**Média**	**Mediana**	**Desvio Padrão**
Tempo de predição (s)	0.1190	0.1158	0.0122
IoU (%)	82.2	84.2	09.5
Confiança (%)	97.2	98.7	08.3

Tabela 15 – Detecções no experimento 2

Placas encontradas	**Placas não encontradas**
159	1

5.3 Experimento 3

O Experimento 3 também está relacionado com a predição das imagens do subconjunto de teste, definido na base de dados. Este experimento utilizou o algoritmo YOLOv2 considerando os hiperparâmetros indicados pelo TPE (CTPE), conforme apresentado no Teste 3 da Tabela 11. Executando o teste de predição, foram coletados os valores das métricas de desempenho que constam na Tabela 16. A Tabela 17 apresenta a quantidade de placas detectadas e não detectadas neste experimento. Com um tempo médio de 0.0591 segundos, IoU médio de 68.9% e taxa de confiança 74.5%, o YOLOv2 com CTPE detectou 146 das 160 placas, ou seja, 91.2% das imagens da base de dados.

Tabela 16 – Resultados do experimento 3

Métrica	**Média**	**Mediana**	**Desvio Padrão**
Tempo de predição (s)	0.0591	0.0581	0.0030
IoU (%)	68.9	75.5	24.5
Confiança (%)	74.5	81.2	23.4

Tabela 17 – Detecções no experimento 3

Placas encontradas	**Placas não encontradas**
146	14

5.4 Experimento 4

O Experimento 4 está relacionado com a predição das imagens do subconjunto para testes definido na base de dados. Este experimento utilizou o algoritmo YOLOv3

considerando os hiperparâmetros indicados pelo TPE (CTPE), de acordo com o Teste 4 na Tabela 11. Executando o teste de predição, foram coletados os valores das métricas de desempenho que constam na Tabela 18. A Tabela 19 mostra a quantidade de placas detectadas e não detectadas neste experimento. Com um tempo médio de 0.14 segundos, IoU médio de 83.7% e taxa de confiança 96.1%, o YOLOv3 com CTPE detectou 159 das 160 placas, ou seja, 99.3% das imagens da base de dados.

Tabela 18 – Resultados do experimento 4

Métrica	**Média**	**Mediana**	**Desvio Padrão**
Tempo de predição (s)	0.1404	0.1388	0.0107
IoU (%)	83.7	85.1	09.3
Confiança (%)	96.1	96.7	03.8

Tabela 19 – Detecções no experimento 4

Placas encontradas	**Placas não encontradas**
159	1

5.5 Discussão

Os experimentos realizados possuem objetivo de coletar métricas de desempenho relevantes para que os algoritmos YOLOv2 e YOLOv3 sejam avaliados na atividade de localizar placas veiculares. Para isto, foram executados quatro experimentos, dois por algoritmo, utilizando um conjunto de hiperparâmetros encontrado em trabalhos na literatura, onde neste trabalho foi chamado de CPAPER; e um conjunto de hiperparâmetros indicados pelo otimizador TPE, chamados de CTPE. Além disso, a base de dados com imagens de veículos em ambientes complexos foi dividida de modo que cada experimento pudesse ser justamente executado com as mesmas imagens.

Após a coleta das métricas de desempenho i) Tempo de predição em segundos; ii) IoU; e iii) Taxa de confiança; apresentadas no capitulo anterior (5), pôde-se isolar cada métrica permitindo avaliar os quatro experimentos. A Tabela 20 destaca em negrito, que o experimento 3 apresentou menor tempo de predição para localizar as placas contidas no subconjunto da base de dados para testes, ou seja, o algoritmo YOLOv2 com CTPE mostrou-se mais rápido em relação aos demais experimentos.

Tabela 20 – Valores do tempo de predição em segundos dos experimentos

Experimento	**Média**	**Mediana**	**Desvio Padrão**
1	0.0656	0.0650	0.0038
2	0.1190	0.1158	0.0122
3	**0.0591**	**0.0581**	**0.0030**
4	0.1404	0.1388	0.0107

Em termos de IoU, a Tabela 21 destaca que o experimento 4 apresentou maior IoU, ou seja, o algoritmo YOLOv3 com CTPE mostrou-se ter maior capacidade para demarcar regiões em que há placa veicular nas imagens.

Tabela 21 – Valores do IoU dos experimentos

Experimento	Média	Mediana	Desvio Padrão
1	71.2	76.7	21.2
2	82.2	84.2	09.5
3	68.9	75.5	24.5
4	**83.7**	**85.1**	**09.3**

Isolando a métrica de porcentagem de confiança dos algoritmos, a Tabela 22 mostra um empate dentro da faixa do desvio padrão nos experimentos 2 e 4. Portanto o algoritmo YOLOv3 com CPAPER e CTPE mostraram-se ter tecnicamente a mesma capacidade para identificar placas veiculares em imagens.

Tabela 22 – Valores de confiança dos experimentos

Experimento	Média	Mediana	Desvio Padrão
1	73.5	78.4	20.4
2	**97.2**	**98.7**	**08.3**
3	74.5	81.2	23.4
4	**96.1**	**96.7**	**03.8**

Por fim, têm-se a quantidade de placas veiculares encontradas durante a predição dos algoritmos utilizando o subconjunto da base de dados para testes. A Tabela 23 destaca um empate entre o experimento 2 (YOLOv3 com CPAPER) e o experimento 4 (YOLOv3 com CTPE). Nesse quesito, percebe-se que a diferença entre os hiperparâmetros não afetou o desempenho do YOLOv3, o qual obteve um resultado melhor em relação aos experimentos com o YOLOv2, localizando quase todas as placas veiculares do conjunto da base de dados para testes.

Tabela 23 – Quantidade de placas encontradas e não encontradas dos experimentos

Experimento	Encontradas	Não encontradas
1	151	9
2	**159**	**1**
3	146	14
4	**159**	**1**

A nível de detalhes, foi verificado quais imagens os algoritmos não conseguiram localizar a placa veicular. Sendo assim, a Figura 26 apresenta a única imagem que não foi localizada por nenhum dos quatro experimentos. Um possível motivo para este caso particular pode ser dado por conta da resolução da região da placa. Esta imagem é a única que possui grande resolução de todas as imagens da base de dados.

Figura 26 – Imagem do conjunto de testes da base de dados que não foi detectada nos experimentos.

Fonte – Do autor.

A Figura 27 apresenta recortes de algumas imagens que não foram detectadas nos experimentos utilizando o algoritmo YOLOv2.

Figura 27 – Imagem do conjunto de testes da base de dados que não foi detectada nos experimentos com o YOLOv2.

Fonte – Do autor.

A Figura 28 apresenta as duas maiores imagens as quais os dois algoritmos, utilizando o conjunto de hiperparâmetros CPAPER, localizaram a placa com as maiores taxas de Confiança. Já a Figura 29 mostra as duas maiores imagens as quais os algoritmos YOLOv2 e YOLOv3 utilizando o conjunto de hiperparâmetros CTPE, localizaram a placa também com as maiores taxas de Confiança.

Figura 28 – Imagens com maiores taxas de Confiança usando CPAPER.

(a) YOLOv2: 94.6% (b) YOLOv3: 99.9%

Fonte – Do autor.

Figura 29 – Imagens com maiores taxas de Confiança usando CTPE.

(a) YOLOv2: 95.7% (b) YOLOv3: 99.7%

Fonte – Do autor.

Em alguns casos o YOLOv3 também localizou mais de uma placa na imagem porém, neste trabalho, foi considerada apenas a placa em evidência na imagem. A Figura 30 mostra mostra dois casos em que duas placas foram localizadas na imagem.

Figura 30 – Localização de mais de uma placa na imagem com YOLOv3.

(a) (b)

Fonte – Do autor.

Os resultados desses quatro experimentos para a detecção de placas veiculares se mostraram excelentes, visto que a literatura define um bom resultado com valores de IoU a cerca de 70%. Inclusive, em trabalhos relacionados, considerou-se IoU aceitável acima de 50% (XIE et al., 2018). Para o tempo de predição, o algoritmo YOLOv2 com CTPE apresentou a menor média de tempo (0.0591 segundo) com IoU médio de 68.9%. Em termos de IoU, o YOLOv3 com CTPE apresentou o maior valor (83.7%) com tempo de predição médio de 0.1404 segundos. Já em questão de taxa de confiança, o YOLOv3 com CPAPER e CTPE apresentaram um empate técnico dentro da faixa do desvio padrão, mas com o YOLOv3 e CTPE obtendo o menor desvio padrão.

O YOLOv3 apresentou melhores resultados em relação a YOLOv2 pois, o IoU e a taxa de confiança se mantiveram superiores. Além disso, o YOLOv3 não detectou apenas uma placa, devido a grande resolução ocupada pela placa na imagem, detectando 99.3% da base de dados. Nestes experimentos, a otimização de hiperparâmetros com o TPE aumentou sutilmente o desempenho dos algoritmos, para o YOLOv2, o ganho se mostrou em termos de tempo de predição. Para o YOLOv3, os ganhos foram encontrados em termos de IoU e taxa de confiança. Os melhores resultados entre os experimentos foram obtidos com o conjunto de hiperparâmetros CTPE, indicado pela operação de otimização de hiperparâmetros com o TPE. Sendo assim, comprova-se que a otimização melhorou o desempenho em detectar placas veiculares.

6 Conclusão

Este trabalho realizou um estudo sobre técnicas existentes de processamento de imagem, aplicações de aprendizagem de máquina e aprendizagem profunda para localizar em imagens, placas veiculares (PV) em ambientes complexos. Seguindo a tendência de pesquisa e motivado pelos ótimos resultados, adotou-se dois algoritmos baseados em aprendizagem profunda YOLOv2 e sua versão sucessora, YOLOv3. Esta última, na qual ainda não foi avaliada para localização de PV, até o melhor do nosso conhecimento. Além disso, foi realizado um estudo sobre otimização dos hiperparâmetros dessas redes neurais, uma etapa não encontrada em nenhum outro trabalho relacionado. Neste trabalho foi adotado o TPE para testar várias combinações de hiperparâmetros, com o objetivo de maximizar os resultados.

Para que os algoritmos fossem avaliados igualmente na tarefa de localização de PV em ambientes complexos, uma base de dados foi construída e rotulada com 400 imagens baseadas em problemas como: ângulo de perspectiva da cena, condições de iluminação, distância entre a placa e a câmera, nível de ruído e objetos que podem ser confundidos com placas veiculares.

Os resultados da primeira fase da otimização de hiperparâmetros com o TPE, indicou quatro combinações, os quais passaram para uma segunda etapa onde o melhor conjunto (CTPE) foi selecionado para realizar os treinamentos dos algoritmos. Além disso, foi feito uma pesquisa na literatura para saber quais são os valores dos hiperparâmetros usados comumente (CPAPER).

Os resultados dos quatro experimentos para a predição de PV com os dois algoritmos e os dois conjuntos de hiperparâmetros, mostraram-se satisfatórios. Em termos de tempo de predição, o algoritmo YOLOv2 com CTPE apresentou a menor média de tempo (0.0591 segundo). Em termos de média de IoU, o YOLOv3 com CTPE apresentou o maior valor (83.7%). Já em questão de taxa de confiança, houve um empate dentro da faixa do desvio padrão entre o YOLOv3 com CPAPER e YOLOv3 com CTPE. Sendo o YOLOv3 apresentando menor desvio padrão.

Em linhas gerais, o YOLOv3 apresentou melhores resultados pois o IoU e a taxa de confiança foram maiores; e por não detectar apenas uma placa, devido a grande resolução ocupada pela placa na imagem, alcançando 99.3% de detecção da base de imagens. Nestes experimentos, a otimização de hiperparâmetros com o TPE também foi viável. Para o YOLOv2, o ganho se mostrou em termos de tempo de predição. Para o YOLOv3, os ganhos foram encontrados em termos de IoU e taxa de confiança. Entre os experimentos de forma geral, os melhores resultados foram obtidos com o conjunto de hiperparâmetros

CTPE, indicado pela otimização de hiperparâmetros com o TPE. Portanto, a otimização realmente melhorou o desempenho dos algoritmos.

6.1 Trabalhos Futuros

Para trabalhos futuros, outros algoritmos baseados em aprendizagem profunda podem ser analisados, outras técnicas de otimização de hiperparâmtros, bem como outros hiperparâmetros podem ser aplicados para maximizar os resultados. Poderiam ser feitos mais execuções afim de aumentar a precisão das médias calculadas a partir de cada métrica. Além disso, experimentos com *Data Augmentation* poderiam ser realizados para adicionar mais informações ao treinamento dos algoritmos, reforçando o modelo gerado. Também é válido expandir a base de dados para ser testada em mais padrões, como as placas vermelhas utilizadas em veículos de transporte. Em um teste rápido (Figura 31a), o YOLOv3 localizou uma placa vermelha mesmo sendo treinado apenas com exemplos de placas de veículos particulares (cinza). Da mesma forma, foi realizado um teste rápido sobre o padrão de placas veiculares do Mercosul (Figura 31b), pois nos próximos meses os novos veículos vendidos no Brasil já terão que adotar o padrão de placas do Mercosul (VEJA, 2018). Por fim, outro padrão que pode ser explorado é o de placas para motocicletas brasileiras.

Figura 31 – Localização de outros padrões de placas veiculares com o YOLOv3.

(a) Placa de transporte (b) Mercosul

Fonte – 31a: UFOP-Gate (PEIXOTO et al., 2015). 31b: Internet.

6.2 Contribuições

Durante este trabalho foi construída uma base de dados com 400 imagens de veículos em ambientes complexos, com base em problemas como: ângulo de perspectiva da cena, condições de iluminação, etc. Todas as imagens foram rotuladas manualmente

de modo que outros trabalhos possam executar seus experimentos desde que sejam para fins acadêmicos. Como contribuição, também foi realizado um experimento sobre a nova versão do algoritmo YOLO (REDMON; FARHADI, 2018b). É importante ressaltar que a versão 3 lançada neste ano (2018b) e utilizada neste trabalho, ainda não foi estudada para fins de localização de placas veiculares, até o melhor do nosso conhecimento. Além disso, este projeto utilizou o algoritmo TPE (Hyperopt, 2018) como técnica para otimização de hiperparâmetros das redes neurais convolucionais abordadas. Isto permitiu explorar, de forma automática e eficiente, uma gama de combinações de hiperparâmetros para que os resultados pudessem ser maximizados.

Referências

AL-GHAILI, M. et al. Vertical edge based car license plate. *IEEE Transactions on Vehicular Technology, VOL. 62*, 2013. Citado na página 32.

ANAGNOSTOPOULOS, C. et al. *Using sliding concentric windows for license plate segmentation and processing*. 2005. Citado na página 19.

AREL, I.; ROSE, D. C.; KARNOWSKI, T. P. Deep machine learning - a new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, 2010. Citado 2 vezes nas páginas 24 e 26.

ASTAWA, I. et al. *Detection of License Plate using Sliding Window, Histogram of Oriented Gradient, and Support Vector Machines Method*. 2017. Citado 2 vezes nas páginas 18 e 19.

AUTOIDEIAS. *Cores das Placas dos Carros*. 2018. Disponível em: <http://www.autoideia.com.br/cores_das_placas_dos_automoveis>. Acesso em: 22-05-2018. Citado na página 14.

BENGIO, Y. Deep learning of representations for unsupervised and transfer learning. *Proceedings of the Unsupervised and Transfer Learning challenge and workshop*, 2011. Citado na página 22.

BERGSTRA, J. et al. *Algorithms for hyper-parameter optimization*. 2011. Citado na página 31.

BERGSTRA, J.; YAMINS, D.; COX, D. D. *Making a Science of Model Search*. 2012. Citado na página 31.

BERGSTRA, J.; YAMINS, D.; COX, D. D. *Hyperparameter optimization in hundreds of dimensions for vision architectures*. 2013. Citado na página 31.

CAVALCANTI, G. et al. Brazilian vehicle identification using a new embedded plate recognition system. *Elsevier Measurement VOL. 70*, 2015. Citado na página 33.

CHEN, C. et al. *Application of Image Processing to the Vehicle License Plate Recognition*. 2013. Citado na página 17.

CHEVTCHENKO, S. *A Convolutional Neural Network with Feature Fusion for Real-Time Hand Posture Recognition*. Dissertação (Mestrado) — Universidade Federal Rural de Pernambuco, 2018. Citado 3 vezes nas páginas 21, 22 e 23.

EVERINGHAM, M. et al. *The PASCAL Visual Object Classes (VOC) Challenge*. 2010. Citado na página 39.

Folha SP. *Brasil tem 1 roubo ou furto de veículo a cada minuto; Rio lidera o ranking*. 2017. Disponível em: <http://www1.folha.uol.com.br/cotidiano/2017/10/1931061-brasil-tem-1-roubo-ou-furto-de-veiculo-a-cada-minuto-rio-lidera-o-ranking.shtml>. Acesso em: 18-05-2018. Citado na página 13.

FONSECA-GALINDO, C. et al. Sistema automático para reconhecimento de placas de automóveis baseado em processamento digital de imagens e redes perceptron de múltiplas camadas. *XIX ENMC e VII ECTM*, 2016. Citado 2 vezes nas páginas 13 e 15.

G1 Autoesporte. *Frota brasileira de veículos cresce 1,2% em 2017, diz Sindipeças*. 2018. Disponível em: <https://g1.globo.com/carros/noticia/frota-brasileira-de-veiculos-cresce-12-em-2017-diz-sindipecas.ghtml>. Acesso em: 18-05-2018. Citado na página 13.

George Seif. *I'll tell you why Deep Learning is so popular and in demand*. 2018. Disponível em: <https://medium.com/swlh/ill-tell-you-why-deep-learning-is-so-popular-and-in-demand-5aca72628780>. Acesso em: 16-11-2018. Citado na página 22.

GONÇALVES et al. License plate recognition based on temporal redundancy. *International Conference on Intelligent Transportation Systems*, 2016. Citado na página 33.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 27.

Google. *Google Images*. 2018. Disponível em: <https://www.google.com/images>. Acesso em: 01-03-2018 e 21-06-2018. Citado na página 15.

Google Colaboratory. *Google Colaboratory*. 2018. Disponível em: <https://colab.research.google.com>. Acesso em: 05-10-2018 e 17-11-2018. Citado 3 vezes nas páginas 41, 42 e 43.

GOU, C. et al. *License Plate Recognition Using MSER and HOG Based on ELM*. 2014. Citado na página 18.

HAFEMANN, L. G. *An Analysis of Deep Neural Networks for Texture Classification*. Dissertação (Mestrado) — Universidade Federal do Paraná, 2014. Citado 2 vezes nas páginas 26 e 27.

HE, K. et al. *Deep Residual Learning for Image Recognition*. 2016. Citado na página 29.

HENDERSON, P.; FERRARI, V. End-to-end training of object class detectors. *University of Edinburgh*, 2017. Citado na página 21.

HSU, G. S. et al. *Robust license plate detection in the wild*. 2017. Citado na página 28.

HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology, Wiley Online Library*, 1961. Citado 2 vezes nas páginas 22 e 23.

Hyperopt. *Hyperopt*. 2018. Disponível em: <http://hyperopt.github.io/hyperopt/>. Acesso em: 02-11-2018. Citado 2 vezes nas páginas 42 e 56.

Jupytercon. *The Jupyter Notebook*. 2018. Disponível em: <https://ipython.org>. Acesso em: 02-11-2018. Citado na página 42.

KARN, U. *An Intuitive Explanation of Convolutional Neural Networks*. 2016. Disponível em: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>. Acesso em: 18-11-2018. Citado 2 vezes nas páginas 25 e 27.

Keras. *Keras*. 2018. Disponível em: <https://keras.io>. Acesso em: 02-11-2018. Citado na página 42.

LAROCA, S. et al. A robust real-time automatic license plate recognition based on the yolo detector. *International Joint Conference on Neural Networks*, 2018. Citado 5 vezes nas páginas 14, 15, 20, 28 e 35.

LIU, A. et al. Ssd: Single shot multibox detector. *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016. Citado 2 vezes nas páginas 14 e 20.

MONTAZZOLLI; JUNG. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. *SIBGRAPI*, 2017. Citado na página 34.

O Globo. *Má qualidade do transporte público aumenta a preferência da população por carro*. 2018. Disponível em: <https://oglobo.globo.com/brasil/ma-qualidade-do-transporte-publico-aumenta-preferencia-da-populacao-por-carro-22290803>. Acesso em: 18-05-2018. Citado na página 13.

OpenALPR Cloud API. 2018. Disponível em: <http://www.openalpr.com/cloud-api.html>. Acesso em: 20-12-2018. Citado na página 35.

OpenCV Team. *Open Source Computer Vision Library - OpenCV*. 2018. Disponível em: <https://opencv.org>. Acesso em: 02-11-2018. Citado na página 42.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *CoRR*, 2015. Citado 3 vezes nas páginas 24, 26 e 27.

PEIXOTO, C.-C. et al. Brazilian license plate character recognition using deep learning. *Proceedings of XI Workshop de Visão Computacional*, 2015. Citado 2 vezes nas páginas 14 e 55.

PRATES, D. et al. Brazilian license plate detection using histogram of oriented gradients and sliding windows. *International Journal of Computer Science and Information Technology (IJCSIT)*, 2014. Citado 4 vezes nas páginas 13, 18, 19 e 32.

Python Software Foundation. *Welcome to Python.org*. 2018. Disponível em: <https://www.python.org>. Acesso em: 02-11-2018. Citado na página 42.

QUORA. *How does non-maximum suppression work in object detection?* 2018. Disponível em: <https://www.quora.com/How-does-non-maximum-suppression-work-in-object-detection>. Acesso em: 04-01-2019. Citado na página 30.

RASMUSSEN, C. E.; WILLIAMS, C. K. I. *Gaussian Processes for Machine Learning*. 2006. Citado na página 31.

REDMON, J. et al. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. Citado 2 vezes nas páginas 27 e 28.

REDMON, J.; FARHADI, A. Yolo9000: Better, faster, stronger. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017. Citado na página 28.

REDMON, J.; FARHADI, A. Yolo9000: Better, faster, stronger. *arXiv*, 2018. Citado na página 28.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv*, 2018. Citado 4 vezes nas páginas 28, 29, 35 e 56.

REN, H. et al. Faster r-cnn: Towards realtime object detection with region proposal networks. *Proc. Adv. Neural Inf. Process. Syst.*, 2015. Citado 2 vezes nas páginas 14 e 20.

RIBEIRO, V. et al. Sistema baseado em reconhecimento automático de placas para monitoramento de veículos e caracterização de frequentadores em estabelecimentos. *X Escola Potiguar de Computação e suas Aplicações - SBC*, 2017. Citado 2 vezes nas páginas 13 e 32.

ROSEBROCK, A. *Intersection over Union (IoU) for object detection*. 2016. Disponível em: <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>. Acesso em: 18-11-2018. Citado 2 vezes nas páginas 20 e 21.

SAGHARICHI, P.; SHAKERI, M. License plate automatic recognition based on edge detection. *Artificial Intelligence and Robotics (IRANOPEN)*, 2016. Citado 2 vezes nas páginas 13 e 33.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Elsevier Neural Networks VOL. 61*, 2015. Citado 2 vezes nas páginas 21 e 22.

SILVA, S. M.; JUNG, C. R. *License Plate Detection and Recognition in Unconstrained Scenarios*. 2018. Citado na página 28.

S.JYOTHI et al. *Automatic license plate recognition using pre-processing methods*. 2017. Citado na página 17.

TensorFlow. *TensorFlow*. 2018. Disponível em: <https://www.tensorflow.org>. Acesso em: 02-11-2018. Citado na página 42.

VEJA. *Contran dá novo prazo para implantação de placas com padrão Mercosul*. 2018. Disponível em: <https://veja.abril.com.br/economia/contran-da-novo-prazo-para-implantacao-de-placas-com-padrao-mercosul>. Acesso em: 09-12-2018. Citado na página 55.

WANG, J.; GAO, G.; YANG, H. *The Method Research of Vehicle Image Preprocessing and License Plate Location*. 2009. Citado na página 17.

WANG, Y.-R.; LIN, W.-H.; HORNG, S.-J. *A sliding window technique for efficient license plate localization based on discrete wavelet transform*. 2011. Citado na página 19.

XIE, L. et al. A new cnn-based method for multi-directional. *IEEE Transactions on Intelligent Transportation Systems*, 2018. Citado 5 vezes nas páginas 13, 20, 34, 35 e 53.

YAN-LI, A. *Introduction to digital image pre-processing and segmentation*. 2015. Citado na página 17.

YAZDIAN, T. et al. Automatic ontario license plate recognition using local normalization and intelligent character classification. *Canadian Conference on Electrical and Computer Engineering*, 2014. Citado na página 32.

YUAN, Z. et al. A robust and efficient approach to license plate detection. *IEEE Transactions on Image Processing*, 2016. Citado na página 34.

ZHANG, S. et al. Cat head detection - how to effectively exploit shape and texture features. *European Conference on Computer Vision, vol. 4*, 2008. Citado na página 32.