

Análise de veículos em cruzamentos com semáforos utilizando *Deep Learning*

Luzenildo de Sousa Batista Júnior



CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2019

Luzenildo de Sousa Batista Júnior

Análise de veículos em cruzamentos com semáforos
utilizando *Deep Learning*

Monografia apresentada ao curso Ciência da Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em Ciência da Computação

Orientadora: Thaís Gaudencio do Rêgo

Fevereiro de 2019

Ficha catalográfica: elaborada pela biblioteca do CI.

Será impressa no verso da folha de rosto e não deverá ser contada.
Se não houver biblioteca, deixar em branco.



CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Ciência da Computação intitulado ***Análise de veículos em cruzamentos com semáforos utilizando Deep Learning*** de autoria de Luzenildo de Sousa Batista Júnior, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Tiago Maritan Ugulino de Araújo
Universidade Federal da Paraíba

Prof. Dr. Lincoln David Nery e Silva
Universidade Federal da Paraíba

Prof. Dr. Thaís Gaudencio do Rego
Universidade Federal da Paraíba

João Pessoa, 9 de Novembro de 2018

“Quando você faz em fração de segundo o que os outros levariam horas, tudo parece mágica.”

Steve Jobs

Dedico este trabalho a toda minha família, todos os meus amigos e professores que me acompanharam e me ajudaram nesta jornada.

AGRADECIMENTOS

Meu agradecimento vai para todos que me ajudaram e me acompanharam, me motivando e encorajando para continuar neste caminho e chegar ao objetivo final, que por diversas vezes parece inalcançável, e infelizmente faz com que muitas pessoas acreditem nisso, desistindo no meio da caminhada. Um mantra que eu carrego comigo é o de ser sempre grato por tudo. Pra mim, a maior virtude de um ser humano é saber agradecer, principalmente pelas pequenas coisas.

Primeiramente eu gostaria de agradecer a Deus por todas as bênçãos recebidas, todos os livramentos, toda capacidade e oportunidade que ele proveu à mim e a minha família ao logo de toda nossa vida. Sei que a vida não é fácil, e em João capítulo 16, versículo 33, Jesus nos fala que “Neste mundo vocês terão aflições; contudo, tenham ânimo! Eu venci o mundo.”, entretanto, apesar disso, é perceptível toda a grandiosidade e glória de Deus na minha e na vida da minha família.

Em segundo lugar, eu gostaria de agradecer à minha família, principalmente minha mãe, meu pai e minha irmã, por terem acreditado desde o primeiro momento na minha capacidade de alcançar todos os meus sonhos e objetivos, me apoiando e suportando em todos os momentos difíceis e me motivando sempre que eu pensei em desistir. Muito obrigado especificamente ao meu pai, por sempre estar ao meu lado em todas as minhas decisões, me aconselhando e orientando, por toda luta diária pra sempre oferecer o melhor pra todos, e principalmente por ser um pai incrível e cuidadoso sempre. Muito obrigado também à minha mãe, por toda insistência em querer e lutar pra que os seus filhos conseguissem todas as coisas que, por motivos plausíveis ela não pôde alcançar. Um sincero agradecimento também a minha irmã, que apesar de muito pequena, participou de todas as minhas caminhadas e lutas diárias nessa caminhada.

Em terceiro, gostaria de agradecer à minha namorada Luana, que chegou na minha vida no meio de todo esse caos e conseguiu pegar todas as nuances das minhas agoniias pra conseguir completar essa etapa da minha história. Muito obrigado por todo suporte, por todas as palavras de conforto e encorajamento, por toda ajuda com este trabalho, me ouvindo falar sobre todo tipo de assunto da computação pacientemente, mesmo sem entender nada, por me ajudar a corrigi-lo, melhorando meu vocabulário e principalmente por ser minha melhor amiga.

Em quarto lugar, gostaria de agradecer à todos os meus amigos da vida e da Universidade, principalmente todos os meus amigos do LAViD, que são pessoas muito especiais pra mim, pois estiveram acompanhando e me ajudando em todo meu crescimento profissional. Um agradecimento especial aos meus amigos: Cláudio Djohnnatha, por me aguentar quase que diariamente no laboratório, aperreando por causa da falta de tempo pra estudar e conseguir fazer os trabalhos da universidade, e ao meu amigo Felipe

Sanguinetti, que me ajudou com as imagens do drone. Sem essa ajuda, esse trabalho não teria chance de ter chegado no resultado que chegou. Muito obrigado a todos, vocês foram de grande importância pra minha vida, e independente de qualquer distância que possa vir a nos separar, vocês continuarão pra sempre no meu coração.

E por último, mas não menos importante, quero agradecer aos meus professores e orientadores, em especial à minha orientadora deste trabalho, Professora Dr. Thaís Gaudêncio, que é uma mulher admirável como pessoa e por todas as suas conquistas, muito obrigado por todos os puxões de orelha e cobranças. Gostaria de agradecer também aos meus orientadores de projetos do LAViD: Carlos Eduardo Batista que me orientou nos projetos "Meu olhar", "Auris" e atualmente me orienta no projeto do "Obras-4D" junto com o Professor Dr. Derzu Omaia, ao Professor Dr. Raoni Kulesza que foi meu orientador no projeto do Ministério do Desenvolvimento Social para criação de um aplicativo iOS para o Bolsa Família, e ao meu primeiro orientador Professor Dr. Thiago Maritan. Sou muito grato a todos vocês por terem feito parte do meu crescimento profissional.

Aos que não foram citados aqui, meu muito obrigado por tudo. Saibam que sempre que precisarem, podem contar comigo pra qualquer coisa.

RESUMO

Semáforos são utilizados desde o século 20 em diversas partes do mundo para controlar o tráfego de ruas, mas em consequência do número de veículos aumentando vertiginosamente, há o aumento do fluxo de carro nas ruas e avenidas, podendo gerar congestionamentos em horários de pico, que em algumas situações, como por exemplo, ao acontecer um acidente, pode alcançar quilômetros. Uma solução inteligente e eficiente deve ser desenvolvida para resolver estes problemas, e além disso, otimizar o tempo de viagem e de espera em cruzamentos. Este trabalho propõe a implementação de um sistema de contagem de veículos para ser utilizado em semáforos inteligentes que recebe um fluxo de vídeo e o processa utilizando visão computacional, para detectar e contar os veículos, com o objetivo de testar a praticabilidade de implementar um sistema de controle do fluxo em tempo real. Com este sistema foram realizados testes com imagens feitas a partir da perspectiva de um semáforo, simulando um cenário real, para analisar a viabilidade da utilização desse sistema tomando como base a quantidade de acertos na detecção dos veículos. Os resultados mostraram que de quatro cenários testados, o sistema teve êxito em acertar três deles, sendo que o único erro foi porque um carro parou atrás de um ônibus, dificultando sua detecção. Portanto, este sistema pode ser utilizado por semáforos para detectar e contar veículos, abrindo espaço para trabalhos relacionados que permitam um melhor controle do fluxo de automóveis.

Palavras-chave: <Semáforos Inteligentes>, <Processamento digital de imagens>, <Deep Learning> <TensorFlow> <Cidades Inteligentes>.

ABSTRACT

Traffic lights have been used since the 20th century in many countrys of the world to control street traffic, but with the number of vehicles increasing dramatically, there are many issues with the massive increase of car flow in the streets and avenues, which can cause kilometric congestions. An intelligent and efficient solution should be developed to solve these problems, and in addition, optimize travel and waiting times at street crossings. This work presents a simple implementation of an intelligent traffic light system which receives a video stream and processes it using digital imaging processing to detect and count vehicles, in order to control the flow at real time. With this system, tests were made using images from a traffic light perspective, simulating a real traffic camera scenario, to analyze the feasibility of using this system, based on the number of correct detection results of vehicles. The results aqurided here, showned that, from four scenarios tested, the system obtained success in three of them, and the any main error was because of a car stopped behind a bus, making it very difficult to detect. Furthermore, this system can be used in traffic lights to detect and count vehicles, serving as idea for related works which permits a better control of the traffic flow.

Key-words: <Smart Traffic Lights>, <Digital Imaging Processing>, <Deep Learning> <TensorFlow> <Smart Cities>.

LISTA DE FIGURAS

| | | |
|----|--|----|
| 1 | Exemplo de mapa de profundidade criado via software. Cada nível de cinza é uma segmentação da imagem baseada na distância entre o fundo e o assunto (cinza mais claro) | 18 |
| 2 | Exemplo de rede neural. Primeiro temos as entradas, que recebem os pesos e fluem até os neurônios intermediários para serem calculados. Após o cálculo, eles seguem para os neurônios de saída. | 21 |
| 3 | Estrutura do grafo utilizado pela implementação do TensorFlow. Os círculos com as letras B, W e X são as matrizes de entrada e saída para os nós de operações. De baixo para cima temos as operações: MatMul (multiplicação de matrizes), Add (adição do viés e pesos), ReLu (função de ativação para uma RN contínua) e finalmente a matriz de saída C. | 22 |
| 4 | Exemplo do funcionamento básico do YOLO. Primeiro é feita a divisão da imagem em $S \times S$ blocos, então a partir do processamento, é criado um mapa de probabilidades onde cada bloco recebe uma cor que representa um provável objeto detectado. | 23 |
| 5 | Árvore Hierárquica do YOLOV2 [5]. Os círculos marcados em azul representam as etiquetas disponíveis no treinamento da COCO, enquanto as de cor vermelha são os rótulos do ImageNet. Os círculos pontilhados representam os nós não finais da árvore. | 24 |
| 6 | Mapeamento dos semáforos inteligentes da cidade de Pittsburgh.[13] Cada ponto preto marcado na imagem representa um semáforo inteligente, e entre eles um destaque colorido informa o estado do trânsito, variando entre verde para pouco fluxo e vermelho para fluxo intenso. | 26 |
| 7 | Exemplo de imagem feita pelo drone. | 28 |
| 8 | Drone fazendo imagens a partir da perspectiva de um semáforo da Avenida Epitácio Pessoa, em João Pessoa - Paraíba. | 29 |
| 9 | Parada 1 do Vídeo 1. Os retângulos vermelhos representam os objetos detectados e o verde é a área delimitada o qual tudo que está fora dela deve ser desprezado. Vemos que um ônibus está rotulado como trem (do inglês: <i>train</i>), e dois carros não estão sendo detectados. | 32 |
| 10 | Parada 2 do Vídeo 1. Os retângulos vermelhos envolvem os objetos detectados e o verde é a área delimitada o qual tudo que está fora dela deve ser desprezado. Nesta imagem, uma moto está rotulada como bicicleta e a outra não foi detectada. Um carro também não foi detectado. | 33 |

| | | |
|----|---|----|
| 11 | Parada 1 do Vídeo 2. Os retângulos vermelhos são os objetos detectados e o verde é a área delimitada o qual tudo que está fora dela deve ser desprezado. Podemos ver que um carro que não está na rua está sendo detectado. | 33 |
| 12 | Parada 2 do Vídeo 2. Os retângulos vermelhos são os objetos detectados e o verde é a área delimitada o qual tudo que está fora dela deve ser desprezado. Podemos ver que além do carro fora da via estar sendo detectado, dois veículos dentro da rua não estão sendo considerados. | 34 |
| 13 | Diferença entre o processamento no vídeo completo e apenas na área delimitada. Os retângulos vermelhos mostram os veículos detectados. Podemos concluir que a detecção na área delimitada foi melhor, pois conseguiu detectar mais objetos da imagem. | 35 |
| 14 | Calculo elaborado para definir onde o processamento deve ser feito focando apenas a rua. Primeiramente deve-se definir as coordenadas de inicio e final da via no ponto $y = 0$, assim como o inicio e final da rua no ponto $y = \text{onde a via termina}$ dentro da caixa delimitadora que vai ser processada. Com essas coordenadas, podemos prever o que é rua a ser utilizada e o que deve ser desprezado. | 36 |
| 15 | Diferença entre o processamento na área delimitada sem utilização do cálculo da via na imagem da esquerda, e exemplo do mesmo quadro com o cálculo desprezando os veículos fora da rua. Os retângulos vermelhos mostram os veículos detectados. | 37 |
| 16 | Parada 1 do Vídeo 1. Os retângulos vermelhos mostram os veículos detectados. Vemos que apenas um carro não pode ser detectado, pois ele está exatamente atrás do ônibus. | 38 |
| 17 | Parada 2 do Vídeo 1. Os retângulos vermelhos mostram os veículos detectados. Nesta imagem podemos ver que todos os veículos foram detectados corretamente. | 38 |
| 18 | Parada 1 do Vídeo 2. Os retângulos vermelhos mostram os veículos detectados. Todos os carros estão sendo detectados corretamente. | 39 |
| 19 | Parada 2 do Vídeo 2. Os retângulos vermelhos mostram os veículos detectados. Pela imagem podemos concluir que todos os carros também estão sendo detectados corretamente. | 39 |
| 20 | Da esquerda para direita: ônibus sendo detectado como caminhão, moto sendo detectada como carro, caminhão sendo identificado como carro e caminhão sendo etiquetado como ônibus. O retângulo vermelho indica os objetos detectados. | 41 |

LISTA DE TABELAS

| | | |
|---|--|----|
| 1 | Tabela mostrando para o Vídeo 1: as duas vezes que o semáforo ficou vermelho, forçando os carros a pararem (Parada), a contagem do processamento (Contagem Proc), a quantidade real de carros que deveriam ser detectadas (Contagem Real) e o erro. | 32 |
| 2 | Tabela mostrando para o Vídeo 2: as duas vezes que o semáforo ficou vermelho, forçando os carros a pararem (Parada), a contagem do processamento (Contagem Proc), a quantidade real de carros que deveriam ser detectadas (Contagem Real) e o erro. | 32 |
| 3 | Tabela mostrando para o Vídeo 1 depois da delimitação da área, a contagem do processamento, a quantidade de carros que deveriam ser detectadas e o erro. | 37 |
| 4 | Tabela mostrando a contagem do processamento, a quantidade de carros que deveriam ser detectadas e o erro, para o Vídeo 2 com a área delimitada. | 37 |
| 5 | Tabela mostrando as resoluções, quadros por segundo do vídeo, quadros por segundo do processamento e tempo de processamento. No fluxo de vídeo, o tempo de processamento não foi calculado (n/c), já que o processamento estava sendo feito em tempo real. | 40 |

LISTA DE ABREVIATURAS

COCO – Objetos comuns em um contexto (do inglês: *Common Objects in Context*)

FPS – Quadros por segundo (do inglês: *Frames per Second*)

GPU – Unidade de processamento gráfico (do inglês: *Graphic Processor Unity*)

PDI – Processamento Digital de imagens

RN – Redes Neurais

VOC – Desafio de classes de objetos visuais (do ingêns: *Visual Object Classes Challenge*)

YOLO – Você só olha uma vez (do inglês: *You Only Look Once*)

Sumário

| | |
|---|-----------|
| 1 INTRODUÇÃO | 16 |
| 1.1 Definição do Problema | 16 |
| 1.2 Objetivo geral | 16 |
| 1.3 Objetivos específicos | 16 |
| 1.4 Estrutura da monografia | 17 |
| 2 CONCEITOS GERAIS E REVISÃO DA LITERATURA | 18 |
| 2.1 Processamento digital de imagens | 18 |
| 2.2 Visão computacional | 19 |
| 2.3 Aprendizado de Máquina | 20 |
| 2.3.1 TensorFlow | 21 |
| 2.3.2 Yolov2 e Darkflow | 22 |
| 3 TRABALHOS RELACIONADOS | 25 |
| 4 METODOLOGIA | 27 |
| 4.1 Testes | 27 |
| 4.1.1 Avaliação dos testes | 27 |
| 4.1.2 Hardware Utilizado | 30 |
| 5 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS | 31 |
| 5.1 Resultados da execução | 31 |
| 5.1.1 Processamento do vídeo bruto | 31 |
| 5.1.2 Processamento da área delimitada | 34 |
| 5.2 Avaliação de desempenho | 40 |
| 5.3 Problemas encontrados e suas soluções | 41 |
| 5.3.1 Erros de detecção | 41 |
| 5.3.2 Problemas de Hardware | 41 |
| 6 CONCLUSÕES E TRABALHOS FUTUROS | 43 |
| 6.1 Trabalhos futuros | 43 |

1 INTRODUÇÃO

1.1 Definição do Problema

Datado de aproximadamente 1869 [19], o semáforo, vindo diretamente da sinalização de trens como um mecanismo a gás para organizar as locomotivas que circulavam pelas ferrovias, foi introduzido primeiramente na cidade de Londres pelo engenheiro ferroviário John Peake Knight como uma forma de controlar a passagem de trens entre duas intersecções de ruas. O primeiro semáforo de trânsito apareceu a primeira vez no começo do século 20 na cidade de New York para organizar o fluxo do crescente número de carros, pessoas e carroças que dividiam as ruas da cidade [20].

Atualmente, devido a sua praticidade e comodidade, há um aumento natural do número de veículos. Esse aumento, juntamente com o fato de que, na maioria dos casos, as ruas não foram projetadas para tal quantidade de automóveis, ocasiona aglomerados de veículos parados em longos congestionamentos, principalmente em horários considerados de pico, ficando evidente a importância da busca para resolver este problema que atinge o mundo todo. Uma dessas soluções é o controle eficiente do fluxo de automóveis que passam pelos cruzamentos sinalizados, na tentativa de otimizar o tempo de paradas e melhorar o tempo de viagem de maneira inteligente.

Para essa solução, pode-se utilizar sensores, ou até mesmo sistemas inteligentes, como o descrito neste trabalho, que utiliza *Deep Learning* juntamente com processamento digital de imagem para detectar e contar a quantidade de veículos na rua e controlar o fluxo. Os benefícios deste sistema vão muito além do simples controle do tráfego, já que ao otimizar o tempo de parada, faz com que o trânsito flua adequadamente, contribuindo para a melhora da segurança, principalmente em lugares perigosos, onde o índice de criminalidade é alto.

1.2 Objetivo geral

O principal objetivo deste trabalho é construir um sistema capaz de receber um fluxo de vídeo via internet, ou até mesmo diretamente de alguma câmera, processar esta imagem para detectar e contar os veículos que estão em uma determinada área de interesse, que neste caso é na rua a qual o semáforo controla, a fim de futuramente usar estas informações para desenvolver um produto que consiga controlar a abertura e o fechamento do semáforo de maneira eficiente.

1.3 Objetivos específicos

- Fazer uma investigação sobre os atuais modelos de semáforos inteligentes.

- Implementar um sistema para ser utilizado por semáforos para contagem de veículos.
- Desenvolver um sistema de código fonte aberto para que possa ser modificado, utilizado ou estudado por qualquer pessoa interessada no assunto.

1.4 Estrutura da monografia

A estrutura deste trabalho está organizada da seguinte maneira:

O primeiro capítulo apresenta a definição do problema, mostrando um pouco da história dos semáforos, assim como os objetivos gerais e específicos da solução apresentada.

No segundo capítulo está toda a fundamentação teórica, contendo a descrição e conceituação dos assuntos necessários para o entendimento de todos os elementos propostos na solução do problema que este sistema propõe resolver.

Na terceira seção estão as implementações de sistemas semelhantes, que tentam resolver o mesmo problema proposto por este trabalho, reforçando que apenas implementações de empresas privadas que vendem seus sistema como um produto foram encontrados.

No quarto capítulo está toda a metodologia empregada na solução, incluindo o método utilizado para conseguir capturar os vídeos que foram utilizadas para teste, assim como os atributos destas imagens. Neste capítulo também são apresentados detalhes sobre a implementação, como os métodos para conseguir, a partir de uma imagem inteira, processar apenas uma área de interesse.

Na seção de número cinco estão contidos os resultados dos testes feitos utilizando o sistema junto com os vídeos descritos na seção anterior, focando tanto nos resultados práticos do sistema com a contagem de veículos e os erros, como nos resultados dos testes de desempenho, que mostram a quantidade de quadros por segundo conseguida pelo hardware utilizado para teste.

No sexto e último capítulo deste trabalho, está a conclusão da viabilidade de utilização do sistema em um cenário real e algumas ideias para trabalhos futuros, que podem ser construídos utilizando toda a implementação utilizada neste trabalho.

2 CONCEITOS GERAIS E REVISÃO DA LITERATURA

2.1 Processamento digital de imagens

Antes de falarmos sobre Processamento Digital de Imagens (PDI), devemos primeiramente pontuar que uma imagem digital é um conjunto de pontos (pixels) $f(x,y)$, onde x e y são coordenadas espaciais, e a função f é chamada de “intensidade dos níveis de cinza em um dado ponto”[1], ou seja, dado um conjunto de pixels de tamanho n por m , temos que a função f retorna um valor finito que caracteriza a intensidade do nível de cinza de um determinado ponto x e y dentro do intervalo de pixels n por m .

Portanto, PDI é a utilização de algoritmos computacionais para analisar e extrair dados úteis de uma imagem digital. Os primeiros artigos sobre este tema datam do início dos anos 1960 e inicialmente tinham como tema principal objetivo o processamento de imagens para satélites. Com a constante evolução da tecnologia e do hardware dos computadores, é muito comum vermos o uso de PDI no nosso dia-a-dia, como por exemplo, ao tirarmos uma foto, os celulares mais modernos possuem processadores específicos para tratar do processamento dessa imagem. Alguns destes processadores tem como objetivo segmentar a imagem em diferentes níveis de profundidade entre o assunto (objeto em primeiro plano) e o plano de fundo da imagem, aplicando níveis de filtro de desfoque em cada um destes segmentos. A Figura 1 mostra a segmentação de uma imagem em diferentes níveis de profundidade, recurso presente nos novos processadores A12 Bionic dos novos iPhone de décima segunda geração.

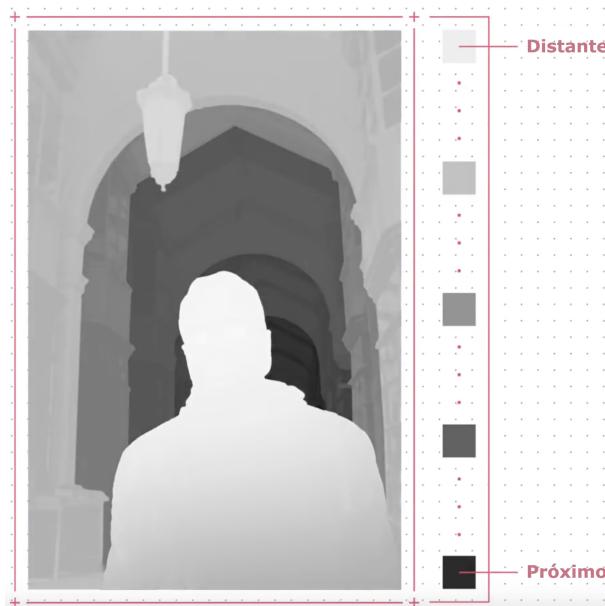


Figura 1: Exemplo de mapa de profundidade criado via software. Cada nível de cinza é uma segmentação da imagem baseada na distância entre o fundo e o assunto (cinza mais claro)

Como descrito em [2], PDI pode ser dividido em 3 níveis principais: processamento de nível baixo, que consiste no processamento de imagens digitais onde a entrada será uma imagem, e a saída será uma outra imagem modificada com a aplicação de filtros. Este nível é muito utilizado para processamento de fotografias digitais, como no uso de filtros de redução de ruído ou filtros de aumento de nitidez para imagens profissionais. O segundo nível é o nível médio e é o processamento onde a entrada é uma imagem, e a saída são os seus atributos. Um exemplo prático da utilização deste nível é quando utilizamos software para detecção de objetos ou segmentação de uma imagem. O último é o nível alto, que consiste no processamento em que a entrada são as características da imagem, geralmente adquiridas pelo processamento de nível médio, e a saída é a descrição ou entendimento dessas características, que resultam em alguma ação autônoma, como nos sistemas de decisão de um automóvel autônomo, onde a partir das características de um processamento de imagem, uma inteligência artificial deve decidir qual ação o carro deve tomar.

Como abordado anteriormente, o processamento digital é dividido em três níveis, onde o nível médio é o nível intermediário entre o nível baixo e o nível alto e consiste no processamento da imagem, que na maioria das vezes é proveniente do processamento de baixo nível, onde a saída será os seus atributos, como por exemplo, a posição de um certo objeto na imagem, ou a sua forma [6]. Esse é o nível de processamento dedicado especificamente para anotações, reconhecimento, detecção de objetos e correspondência de similaridade.

O autor de [7] compara o processamento de imagens ao processamento de texto, onde ele descreve que um texto é dividido em palavras e que o processamento destas palavras passa por diversos níveis, que vão desde o processamento das derivações de um verbo, até a remoção de palavras comumente utilizadas através de um filtro. Ele indica que a imagem é similar a um documento, e assim como cada letra é a menor unidade de uma palavra, os pixels são as menores unidades de uma imagem, enquanto as palavras de um texto são análogas a cada segmento local desta imagem, permitindo através da análise e processamento de cada um desses segmentos, extrair informações importantes ou até interpretações sobre a cena descrita na imagem a ser processada.

2.2 Visão computacional

Temos que a visão é um dos sentidos mais importantes dos seres humanos, já que com ela temos a percepção dos objetos e do mundo que nos cerca. Utilizando os conceitos prévios sobre o processamento de imagem de nível médio, podemos conceituá-la como a atribuição ao computador através do aprendizado de máquina, não apenas as características, mas as funcionalidades análogas ao olho humano, incluindo as características análogas ao processamento cerebral, entretanto, diferente do cérebro humano, a capa-

cidade de processamento no caso computacional, depende do *hardware* utilizado para processar essas imagens.

Atualmente, a visão computacional está presente em diversas áreas da ciência da computação, passando pelo reconhecimento de objetos e padrões, com a utilização de algoritmos de classificação e detecção de objetos, até a operação autônoma de robôs em uma fábrica. Além disso, tal assunto apresenta uma forte presença de interdisciplinaridade, como na área de medicina, onde se utiliza a visão computacional para, por exemplo, predizer e diagnosticar doenças. O autor [8] apresenta alguns exemplos do uso da visão computacional em conjunto com a medicina para o combate ao câncer através de algoritmos de processamento de imagens, assim como o uso para diagnosticar outras doenças com a ajuda de aparelhos já conhecidos da medicina como os de ressonância magnética por imagem (MRI), onde com estes dados junto de algoritmos de segmentação de imagens, é possível criar um modelo 3D do órgão do paciente e detectar anomalias muito antes de qualquer intervenção cirúrgica.

2.3 Aprendizado de Máquina

Aprendizagem é um conceito muito complexo, pois é o conjunto de mudanças de comportamento que obtemos através das experiências correlacionadas com fatores emocionais, neurológicos, relacionais e ambientais. Desde a descoberta e com a constante evolução da capacidade dos computadores, fala-se sobre máquinas autônomas capazes de reproduzir ações ininterruptas e repetidas que seriam atribuídas como entrada por humanos. Como uma área do campo de estudo de Inteligência Artificial (IA), o aprendizado de máquina (*Machine Learning*) está presente em quase todos os dispositivos que mais utilizamos, como por exemplo, quando os celulares identificam e aprendem os locais de preferência do usuário, sugerindo outros lugares semelhantes, assim como, quando governos utilizam sistemas para pontuar e ranquear os cidadãos baseando-se nos seus comportamentos e ações nas ruas, como acontece atualmente na China [9].

A abordagem de aprendizado de máquina que será utilizada neste trabalho é o aprendizado baseado em Redes Neurais (RN), que tem como inspiração a maneira como o sistema nervoso biológico funciona. Portanto, similarmente à biologia, na computação as RNs funcionam como um conjunto gigante de elementos de processamento interconectados (neurônios) que tem como finalidade resolver problemas específicos, como problemas de reconhecimento de padrão e problemas de classificação de dados. Assim como o sistema nervoso humano, as Redes Neurais funcionam por aprendizado através de exemplos, e vão ajustando as ligações (pesos e viés) entre os neurônios na tentativa de diminuir o erro do aprendizado [10]. A Figura 2 mostra um exemplo simples de uma rede neural.

Utilizando os conceitos de RN, temos a definição de *Deep Learning*, que é uma

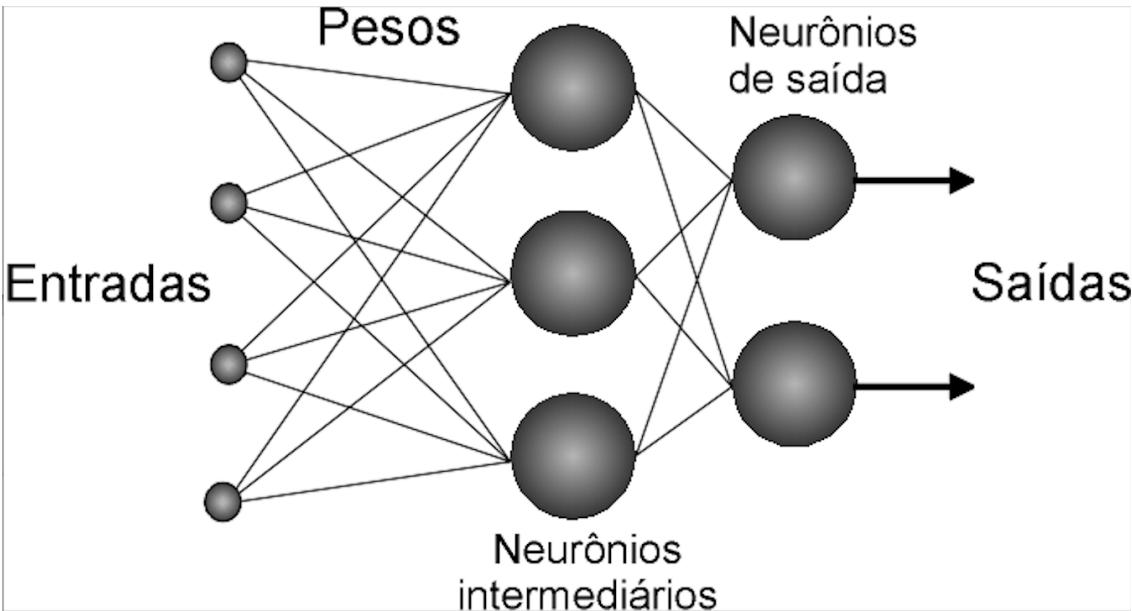


Figura 2: Exemplo de rede neural. Primeiro temos as entradas, que recebem os pesos e fluem até os neurônios intermediários para serem calculados. Após o cálculo, eles seguem para os neurônios de saída.

maneira de classificar uma RN com várias camadas intermediárias, usualmente com mais de 5 camadas, formando uma rede conectada de neurônios, em que cada camada conecta com a posterior, formando uma RN profunda. Cada camada de uma RN profunda tem seus próprios viés e pesos diferentes para cada neurônio, o que permite que vários cálculos sejam feitos para chegar em um resultado ótimo nos neurônios de saída.

2.3.1 TensorFlow

Com os conceitos de *Deep Learning* e Redes neurais apresentados anteriormente, podemos conceituar o *framework* TensorFlow [3], que é uma interface implementada pelo Google, criada em 2010, com o objetivo de facilitar a criação de modelos de teste e a utilização de algoritmos de aprendizado de máquina utilizando redes neurais. Por ser uma interface, ela pode ser facilmente portada para dispositivos diferentes sem que haja grandes diferenças de implementação pra cada dispositivo, ou seja, uma mesma implementação pode ser executada tanto em dispositivos móveis (iOS e Android), quanto em máquinas dedicadas contendo uma ou várias unidades de processamento gráficas (do inglês, *Graphic Processor Unity* - GPU).

Como toda rede neural, as entradas fluem pelas operações e funções em uma estrutura semelhante a um grafo, a diferença é que no TensorFlow os desenvolvedores imaginaram um grafo onde os nós conteriam as operações, que podem receber zero ou mais entradas e retornar zero ou mais saídas, que podem ser, por exemplo, as multiplicações dos pesos e adições do viés, que no caso do TensorFlow são representados por matrizes n

$x m$. Já as arestas do grafo seriam as ligações contendo as matrizes de entrada e saída que percorreriam de um nó (ou operação) a outro. Para essas ligações eles deram o nome de tensores. Além disso, foi implementado o conceito de sessão, que cuida da interação entre as operações e os tensores, onde para iniciar os cálculos da rede neural, o cliente deve iniciar uma sessão e passar como entrada o grafo construído com os nós e os tensores à serem calculados. A Figura 3 mostra como funciona a estrutura básica de grafo do TensorFlow.

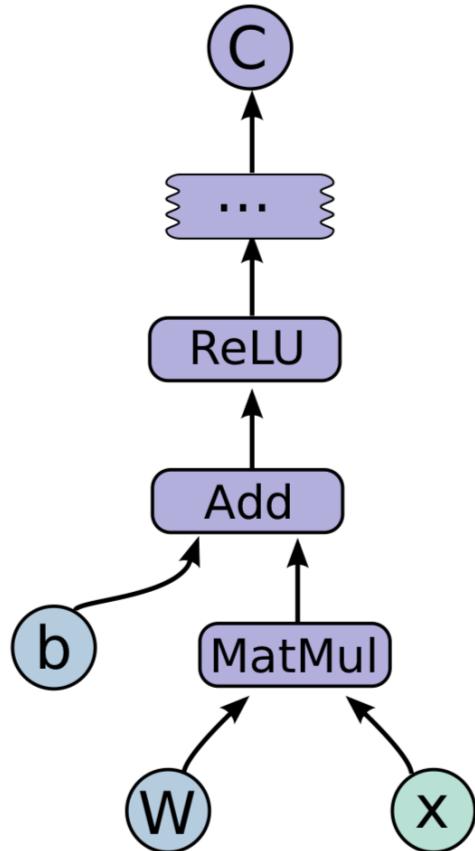


Figura 3: Estrutura do grafo utilizado pela implementação do TensorFlow. Os círculos com as letras B, W e X são as matrizes de entrada e saída para os nós de operações. De baixo para cima temos as operações: MatMul (multiplicação de matrizes), Add (adição do viés e pesos), ReLU (função de ativação para uma RN contínua) e finalmente a matriz de saída C.

2.3.2 Yolov2 e Darkflow

O TensorFlow ainda é uma ferramenta muito complicada de ser utilizada, visto que, para conseguir implementar uma rede neural para, por exemplo, detectar e classificar objetos em uma imagem utilizando *Deep Learning*, é necessário muito tempo e domínio do assunto. Por causa disso, no objetivo de facilitar esse processo, ferramentas e algoritmos

que utilizam todas as funcionalidades do TensorFlow foram criados, um exemplo é o YOLO.

O YOLO é um dos métodos mais inovadores de detecção de imagens, uma vez que, diferente de outras implementações onde são utilizados classificadores para cada classe conhecida, avalia-se diferentes escalas e locais nas imagens recebidas como entrada, fazendo com que múltiplos processos repetitivos sejam feitos diversas vezes numa mesma imagem. O YOLO utiliza redes neurais para separar as imagens em diferentes setores e associar a cada um desses setores uma probabilidade para todas as classes de objeto a serem detectadas [4]. Como a imagem é dividida em blocos, a rede neural do YOLO recebe como entrada cada um desses blocos para prever todas as classes simultaneamente. A Figura 4 mostra como o YOLO divide as imagens e cria um mapa de probabilidade de classes para detectar e classificar os objetos na imagem.

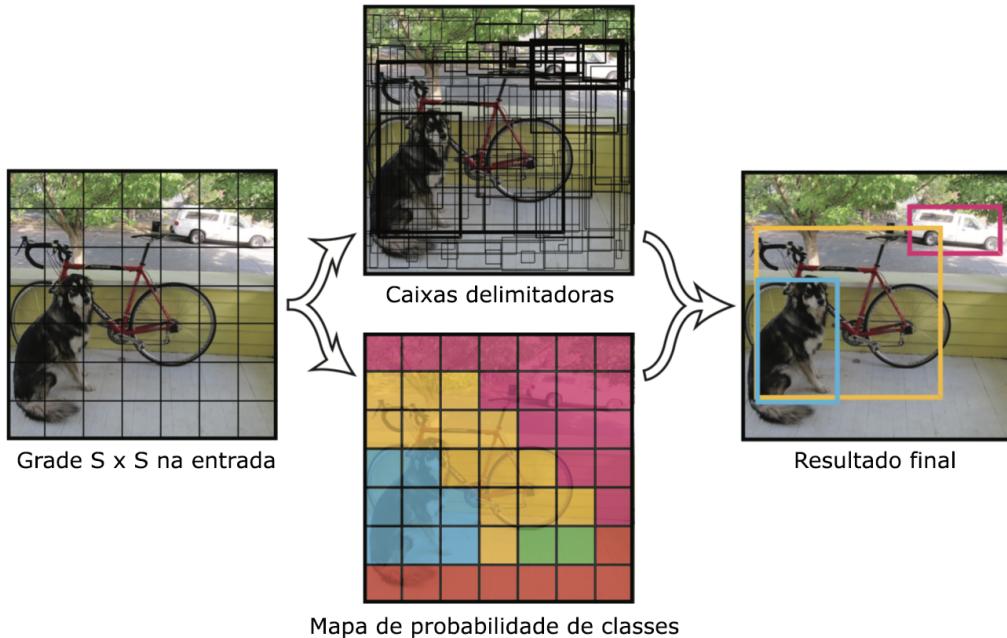


Figura 4: Exemplo do funcionamento básico do YOLO. Primeiro é feita a divisão da imagem em $S \times S$ blocos, então a partir do processamento, é criado um mapa de probabilidades onde cada bloco recebe uma cor que representa um provável objeto detectado.

O YOLOv2 é uma atualização do YOLO em que os desenvolvedores criaram um algoritmo capaz de ser treinado usando dois conjuntos de dados diferentes, um para detecção e outro para classificação, criando uma árvore de palavras juntando todas as classes de classificação e detecção, permitindo que os objetos sejam detectados de uma maneira mais específica. Por isso, o YOLOv2 pode detectar uma infinidade de classes diferentes em tempo real, sem perda de performance [5]. A Figura 5 mostra como é essa junção de dois conjuntos de dados diferentes para criar uma árvore hierárquica de

classificação de objetos. Utilizando as etiquetas de detecção da COCO [17], que é um conjunto de dados de detecção, segmentação e legenda de objetos em grande escala, e as de classificação da ImageNet [21], que corresponde a um banco de dados de mais de 100 mil classes com 1000 imagens cada uma, sendo catalogadas por humanos para ser utilizada em sistemas de classificação, pode-se construir uma árvore hierárquica que contém um número muito maior de classificadores, para aprimorar a rotulação dos objetos da imagem.

O YOLO juntamente com a *Darknet*, que é a rede neural utilizada pela YOLO para fazer todo o processamento, foram implementados na linguagem C, que é conhecida por ser uma linguagem mais complexa, de difícil implementação e difícil entendimento do código escrito. Por esse motivo, Python é uma das linguagens mais utilizadas para processamento de imagem por causa da sua facilidade de implementação, uma vez que por ser mais parecida com uma linguagem natural, os códigos em Python são mais fáceis de serem lidos e entendidos, principalmente por pessoas que não tem muita familiaridade com programação. Logo, o *Darkflow* foi criado para ser uma tradução da implementação do *framework* YOLO para ser utilizado com o TensorFlow em Python.

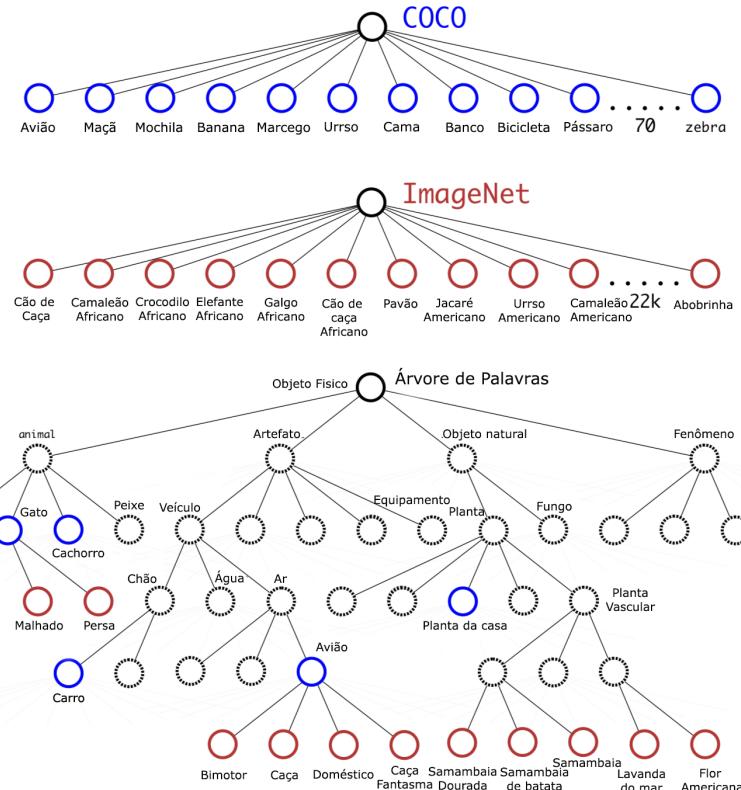


Figura 5: Árvore Hierárquica do YOLOv2 [5]. Os círculos marcados em azul representam as etiquetas disponíveis no treinamento da COCO, enquanto as de cor vermelha são os rótulos do ImageNet. Os círculos pontilhados representam os nós não finais da árvore.

3 TRABALHOS RELACIONADOS

Para este trabalho, foram feitas pesquisas em portais de artigos acadêmicos e em sites de buscas, na intenção de procurar trabalhos relacionados. Na maioria dos casos, foram encontradas implementações privadas de modelos similares com o sistema desenvolvido neste trabalho. A reportagem do autor [11] apresenta um modelo de semáforo inteligente que está sendo utilizado em St. Petersburg - Flórida. Este modelo utiliza sensores no asfalto para detectar a presença de carros para fazer o controle de abertura e fechamento do semáforo. O processamento é feito através de um controlador presente em cada um dos semáforos, que recebe as entradas dos sensores de presença. Quando um carro é detectado, o controlador é avisado para começar a contagem e abrir o semáforo, controlando assim o fluxo de carros. Um sistema semelhante é utilizado no Brasil na cidade de Rios Claros [13], onde uma faixa azul no chão possui sensores indutivos que ao detectar um carro na intersecção, aciona o semáforo fazendo ele ficar verde por 20 segundos. Este método é muito limitado pois não consegue fazer um controle preciso em situações de engarrafamento, já que os sensores são apenas de presença de automóveis, e não de contagem para um melhor monitoramento do fluxo.

Além deste sistema, a reportagem [12] mostra detalhes sobre o sistema de Controle Escalável de Tráfego Urbano (do inglês, *Scalable Urban Traffic Control - SURTRAC*) desenvolvido na Universidade de Carnegie Mellon situada em Pittsburgh - Pennsylvania, que utiliza câmeras e sensores de presença para criar uma rede interconectada de semáforos, que controlam em tempo real, o tempo em que os sinais dos grandes centros ficarão com a luz verde ou vermelha, de acordo com o fluxo de carros em cada rua. Neste sistema, cada semáforo pode fazer seu próprio plano de controle e todos os dados são compartilhados entre os sinais adjacentes, para que eles se programem e se antecipem para o fluxo de carros que irão receber. Estudos feitos por [13] mostram que o tempo de viagem em ruas que possuem os semáforos inteligentes desenvolvidos pela SURTRAC diminuíram em até 26%, além de diminuírem o tempo de paradas em até 31% e o tempo de espera em intersecções em até 41%. A Figura 6 mostra o mapeamento dos 50 semáforos inteligentes da cidade de Pittsburgh, onde cada ponto é um semáforo e as ligações entre eles são marcadas por cores que determinam o estado do fluxo de automóveis.

No Brasil, uma empresa brasileira chamada Seebot criou um sistema muito parecido com o sistema do SURTRAC [15], onde o semáforo possui com dispositivo com uma câmera, que em tempo real visualiza o fluxo de carros observando a quantidade e a velocidade dos veículos para fazer o controle de quanto tempo o semáforo deve ficar aberto ou fechado. Um teste feito por 20 dias comprovou que a melhora no fluxo do trânsito foi de até 49%, melhorando a sua fluidez e até mesmo a segurança dos condutores, já que o tempo de espera, principalmente em momentos de pouco fluxo de carros, foi muito menor.

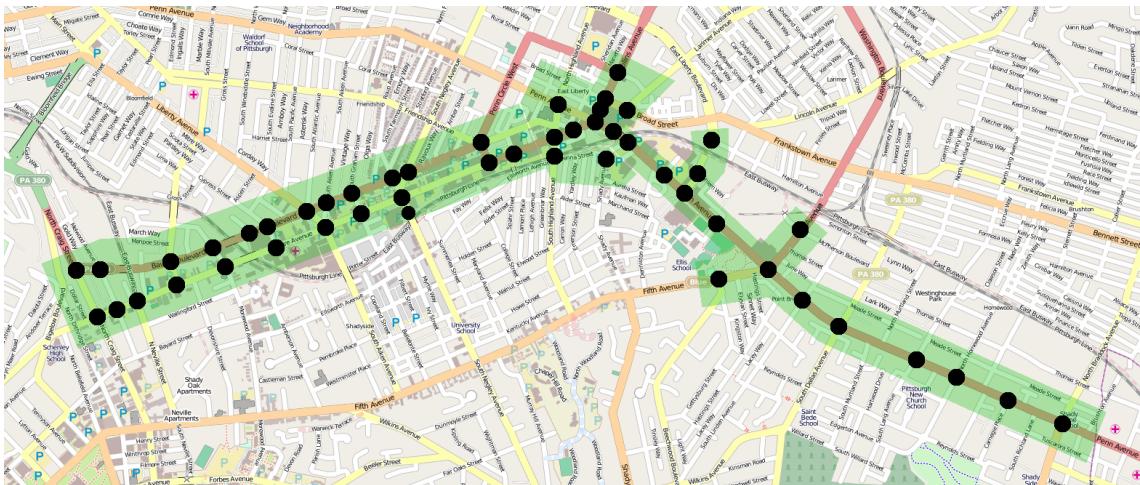


Figura 6: Mapeamento dos semáforos inteligentes da cidade de Pittsburgh.[13] Cada ponto preto marcado na imagem representa um semáforo inteligente, e entre eles um destaque colorido informa o estado do trânsito, variando entre verde para pouco fluxo e vermelho para fluxo intenso.

A Audi também resolveu entrar neste ramo, desenvolvendo um sistema parecido para tentar resolver os problemas de espera excessiva em semáforos. Eles lançaram o sistema de veículo para infraestrutura (do inglês, *Vehicle to Infrastructure* - V2I ou V-to-I), que diferente de todos os métodos apresentados nesta Seção, o carro é quem manda informações em tempo real para uma central de tráfego que informa ao veículo se o próximo semáforo irá fechar ou abrir. Atualmente o sistema funciona de forma muito simples, mas já existem planos de melhorá-lo usando essas informações fornecidas pelos carros para fazer um controle mais preciso do fluxo dos semáforos [14].

Um modelo semelhante ao descrito neste trabalho é descrito pelo autor [22], em que ele descreve um sistema que utiliza processamento de imagem, para definir a densidade de veículos em uma determinada área do video, sendo filmado da perspectiva de um semáforo. O problema, neste caso, decorre do fato que nesse trabalho não foi utilizado visão computacional, por causa disso, não conta a quantidade de veículos na imagem, uma vez que, ele calcula apenas a densidade de veículos em uma determinada parte da imagem.

O sistema proposto por esse trabalho, se assemelha ao do SURTRAC, por ser um sistema que utiliza câmeras para contar a quantidade de veículos, para que permita uma análise dessa contagem para definir uma heurística de controle do tráfego, diferente dos sistemas que utilizam sensores para detectar a presença do carro no semáforo. Análogo à esses sistemas, o sistema proposto neste trabalho tem como objetivo ser de código fonte aberto e ser disponível para qualquer pessoa interessada em utilizá-lo ou estudá-lo.

4 METODOLOGIA

A metodologia deste trabalho começa com a obtenção de possíveis soluções para o problema de otimização do tempo em semáforos e a busca por trabalhos bibliográficos para servir de inspiração para uma boa solução. A partir da pesquisa, apesar de existirem métodos que utilizam sensores para detectar a presença de veículos, a utilização de câmeras e sistemas de detecção de objetos em imagens, solucionam o problema de maneira mais precisa, já que consegue contar exatamente a quantidade de automóveis parados no semáforo, com a utilização de câmeras de boa qualidade, juntamente como um computador integrado para fazer o processamento das imagens e definir uma heurística de controle de fluxo.

Portanto, este trabalho tem como objetivo a implementação de um software que receberá as imagens das câmeras dos semáforos inteligentes e fará o processamento a fim de detectar, classificar e contar os veículos na imagem para, em trabalhos futuros, definir quais sinais de trânsito ficarão abertos ou fechados através de uma heurística de controle de tráfego. Toda programação deste trabalho foi feita utilizando a linguagem de programação Python, com a ajuda dos *frameworks* de RN Tensorflow e o *framework* de detecção de imagens DarkFlow e Yolov2, utilizando modelos pré-treinados da COCO Trainval e VOC 2007+2012 [16].

4.1 Testes

Para obtenção dos vídeos de testes, foi utilizado um drone para fazer dois vídeos. O Vídeo 1 tem duração de 2 minutos e 40 segundos, e o Vídeo 2 tem duração de 3 minutos e 20 segundos. Ambos foram feitos focando o trânsito na perspectiva de um semáforo de trânsito, na intersecção da Avenida Epitácio Pessoa com a Avenida Monteiro Lobato, para servir de modelo de teste. Por causa do clima chuvoso e da interferência dos fios de alta tensão dos postes no local, o drone teve de ficar à uma distância de aproximadamente 11 metros do chão, logo, aproximadamente 6 metros acima do sinal, enquanto a câmera teve de ficar com uma angulação de aproximadamente -20 graus. A Figura 7 mostra um exemplo da filmagem do drone na perspectiva de um semáforo de trânsito. Todos os vídeos foram gravados em *Full-HD* (1920x1080 pixels) à uma taxa de 60 quadros por segundo. A Figura 8 mostra aproximadamente a distância em que o drone ficou do semáforo.

4.1.1 Avaliação dos testes

Para avaliação dos testes, foram executadas aproximadamente 10 instâncias do software, utilizando os dois vídeos feitos pelo drone em resolução total (1920x1080 pixels), e utilizando apenas o quadrante importante da imagem a ser processada, cujas



Figura 7: Exemplo de imagem feita pelo drone.

coordenadas foram definidas manualmente, selecionando os pontos (x_1, y_1) para o inicio do retângulo, e (x_2, y_2) para o final do retângulo, criando uma caixa delimitadora com dimensões de 457×688 pixels para o primeiro vídeo e 497×714 para o segundo vídeo. Foi utilizado o modelo pré-treinado da COCO [17], comparando os resultados de acertos e erros no reconhecimento dos veículos que estavam passando na via, assim como os FPSs durante o processamento e a contagem de carros presentes no vídeo, com o objetivo de simular uma contagem de automóveis parados no semáforo.

Como o objetivo era contar apenas os veículos que estavam na rua e no sentido do fluxo que o semáforo controla, foi elaborado um cálculo de recorte para ser considerada apenas a área da via à ser processado pela Yolo, nos dois vídeos feitos pelo drone.

Com os resultados do processamento, foi feita uma análise da contagem dos veículos nos dois momentos em que o semáforo esteve vermelho e os automóveis pararam, comparando a quantidade detectada pelo sistema com a quantidade que deveria efetivamente ser contada. O erro é calculado subtraindo a quantidade detectada da contagem real. A solução ótima é quando o erro é igual ou próximo de zero.

Para a análise de desempenho, foram utilizados os melhores resultados da taxa de quadros por segundo médio, assim como o tempo total em minutos, necessário para o processamento inteiro do vídeo. Foram utilizados as instâncias dos vídeos em resolução de 1920×1080 pixels, e com a área delimitada com resoluções de 457×688 pixels e 497×714 . Neste caso, quanto menor o tempo, e maior o numero de quadros por segundo, o processamento é mais próximo do tempo real.



Figura 8: Drone fazendo imagens a partir da perspectiva de um semáforo da Avenida Epitácio Pessoa, em João Pessoa - Paraíba.

4.1.2 Hardware Utilizado

Para realizar todos os testes, o hardware utilizado foi um Notebook Lenovo Y50 com as seguintes especificações:

- Processador: *Intel Core i7 4700HQ @ 2.4Ghz.*
- Memória RAM: 8GB DDRL3 1600Mhz.
- GPU: *Nvidia GTX 860M 2GB GDDR5*

5 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Nesta seção serão apresentados os resultados da utilização do software de detecção de carros para semáforos inteligentes, utilizando os recursos descritos na Seção 4 deste trabalho. Esta seção está dividida em três partes principais: a primeira mostra os resultados da execução do software, incluindo os acertos e erros de detecção. A segunda mostra uma breve avaliação do desempenho, e a terceira parte mostra os problemas encontrados nesta solução e possíveis soluções para minimizá-los ou resolvê-los.

5.1 Resultados da execução

Devido ao baixo desempenho do processamento em tempo real feito pelo *hardware* utilizado, uma vez que o processador e a GPU são versões de *notebook*, portanto, são menos eficientes em relação a equipamentos mais potentes e modernos, e para facilitar a compreensão dos resultados, os Vídeos 1 e 2 feitos pelo drone foram processados previamente, utilizando o software. O resultado foi salvo em arquivo de vídeo separado. As tabelas dessa Seção, mostram a contagem de veículos no momento em que o semáforo ficou vermelho e os automóveis tiveram de parar. As Figuras são quadros do video final processado.

5.1.1 Processamento do vídeo bruto

Nos primeiros testes, foi feito o processamento do vídeo em resolução total, para se ter uma ideia de qual seria a quantidade de carros que o sistema iria identificar em cada Vídeo. Posteriormente, o código foi limitado a apresentar os resultados que estavam dentro de uma caixa delimitadora focando apenas a porção importante da imagem, que neste caso é a parte da via cujo fluxo é controlado pelo semáforo. A Tabela 1 mostra os resultados deste processamento para o Vídeo 1. As Figuras 9 e 10 mostram imagens do processamento, desprezando tudo que está fora da caixa delimitadora verde para o Vídeo 1, nos momentos em que o semáforo ficou vermelho, ocasionando a parada dos veículos antes da faixa de pedestre.

A Tabela 2 mostra os resultados para o Vídeo 2, enquanto as Figuras 11 e 12, semelhantemente as duas anteriores, mostram o momento em que os automóveis pararam, respeitando o semáforo de trânsito no Vídeo 2.

Como podemos visualizar nas Tabelas 1 e 2, foram encontrados muitos erros na contagem dos veículos, o que dificulta caso o sistema seja utilizado para criar uma heurística de controle de abertura do semáforo. Isso é resultado do processamento do Vídeo em resolução total. Por este motivo, foi necessário limitar a área de processamento do vídeo

Tabela 1: Tabela mostrando para o Vídeo 1: as duas vezes que o semáforo ficou vermelho, forçando os carros a pararem (Parada), a contagem do processamento (Contagem Proc), a quantidade real de carros que deveriam ser detectadas (Contagem Real) e o erro.

| Parada | Contagem Proc | Contagem Real | Erro |
|--------|---------------|---------------|------|
| 1 | 2 | 4 | -2 |
| 2 | 4 | 6 | -2 |

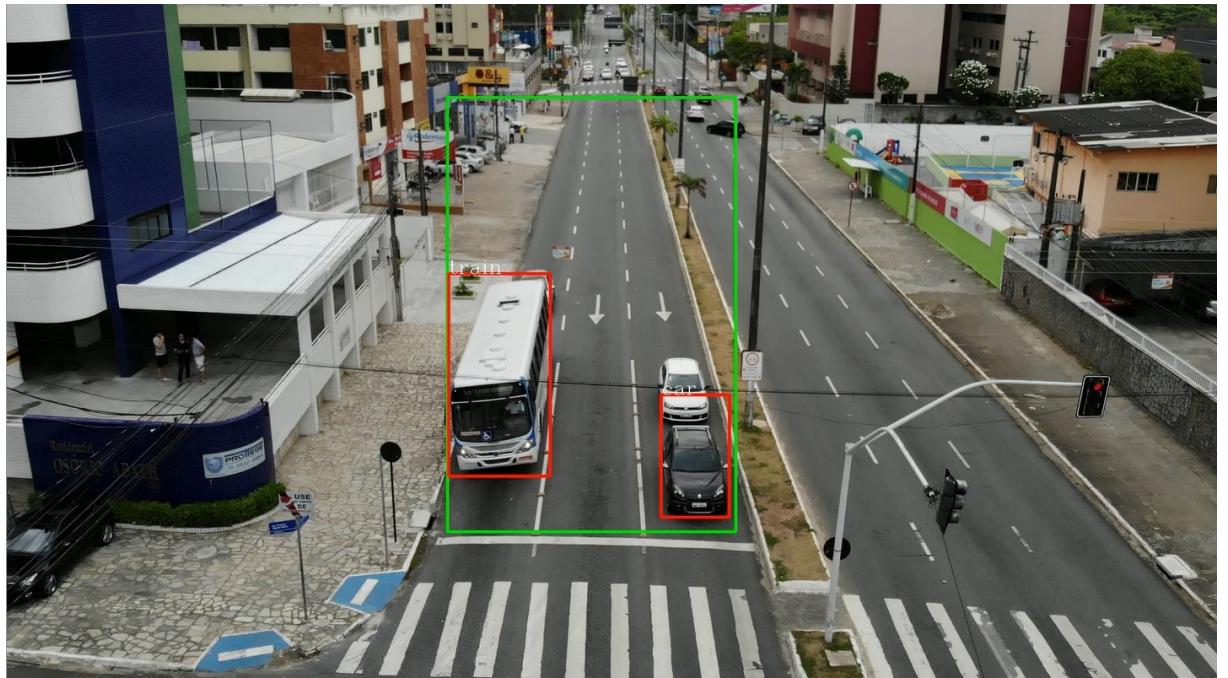


Figura 9: Parada 1 do Vídeo 1. Os retângulos vermelhos representam os objetos detectados e o verde é a área delimitada o qual tudo que está fora dela deve ser desprezado. Vemos que um ônibus está rotulado como trem (do inglês: *train*), e dois carros não estão sendo detectados.

Tabela 2: Tabela mostrando para o Vídeo 2: as duas vezes que o semáforo ficou vermelho, forçando os carros a pararem (Parada), a contagem do processamento (Contagem Proc), a quantidade real de carros que deveriam ser detectadas (Contagem Real) e o erro.

| Parada | Contagem Proc | Contagem Real | Erro |
|--------|---------------|---------------|------|
| 1 | 6 | 5 | -1 |
| 2 | 6 | 7 | 1 |

apenas para a porção da rua que deve ser feita a contagem de veículos, que é considerada a área de interesse da imagem.

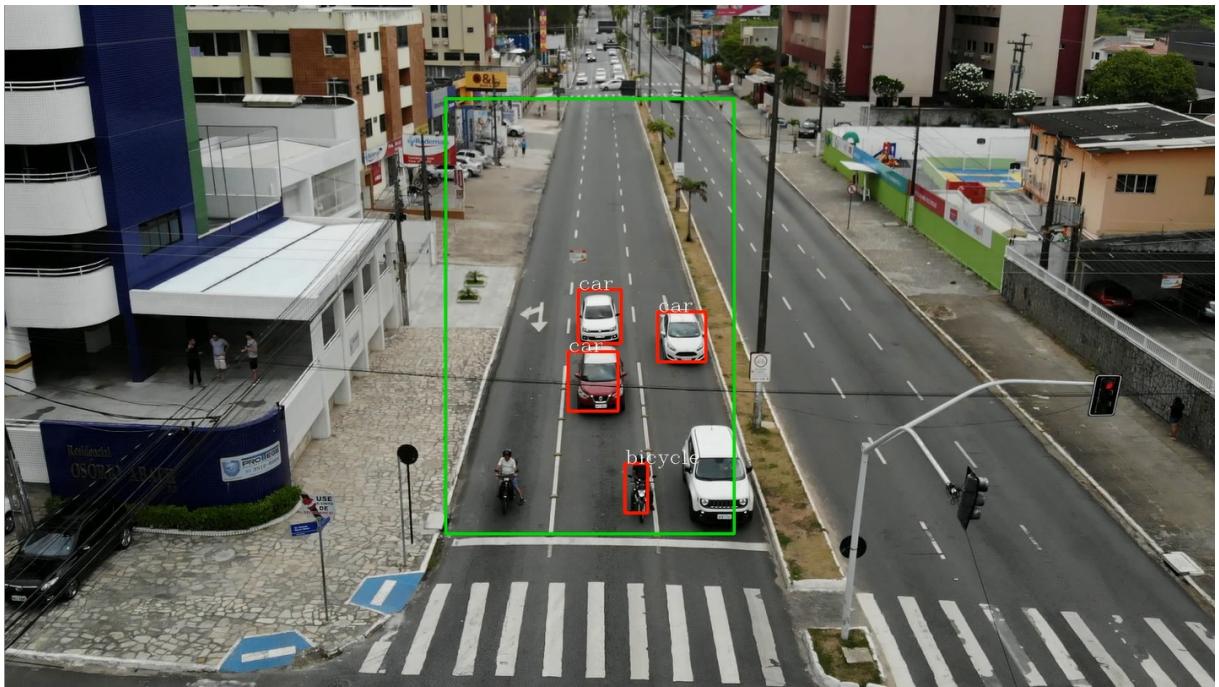


Figura 10: Parada 2 do Vídeo 1. Os retângulos vermelhos envolvem os objetos detectados e o verde é a área delimitada o qual tudo que está fora dela deve ser desprezado. Nesta imagem, uma moto está rotulada como bicicleta e a outra não foi detectada. Um carro também não foi detectado.

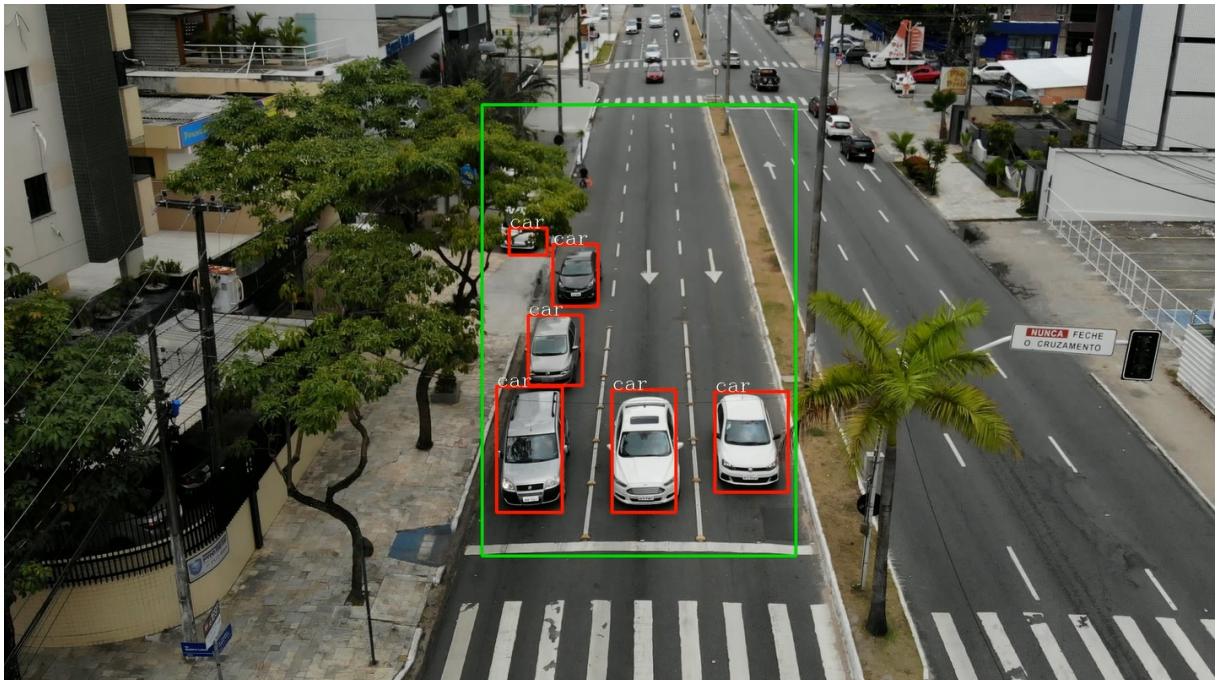


Figura 11: Parada 1 do Vídeo 2. Os retângulos vermelhos são os objetos detectados e o verde é a área delimitada o qual tudo que está fora dela deve ser desprezado. Podemos ver que um carro que não está na rua está sendo detectado.

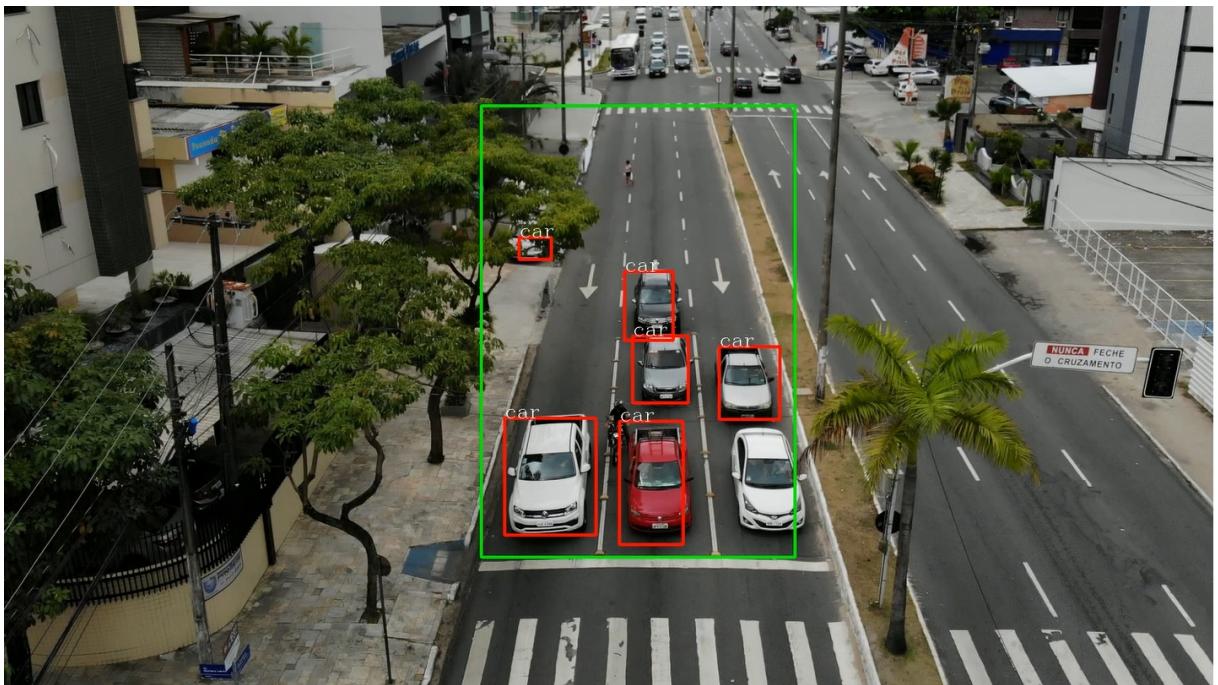


Figura 12: Parada 2 do Vídeo 2. Os retângulos vermelhos são os objetos detectados e o verde é a área delimitada o qual tudo que está fora dela deve ser desprezado. Podemos ver que além do carro fora da via estar sendo detectado, dois veículos dentro da rua não estão sendo considerados.

5.1.2 Processamento da área delimitada

Este teste consistiu em fazer o processamento apenas da área de interesse da imagem, desprezando no processamento tudo que estava fora dela. Tendo em vista que esta área é uma parte da imagem total, a resolução do vídeo a ser processado diminuiu, ficando em 457×688 para o Vídeo 1 e 497×714 para o Vídeo 2. A melhora na velocidade do processamento foi sensível, como está descrito na Seção 5.2 deste trabalho, mas houve uma grande melhora na detecção dos veículos. A Figura 13 mostra a diferença entre a detecção da imagem completa apresentado na Seção 5.1.1 e a detecção com o processamento apenas na área limitada.

Entretanto, apesar do processamento na área delimitada melhorar e destacar apenas a parte da via mais importante, podemos ver ainda na Figura 13 que alguns objetos detectados que não estão dentro da rua também estão etiquetados. Este comportamento não é interessante, uma vez que os carros que trafegam no sentido contrário da via, ou que estão estacionados na calçada não devem ser considerados pelo sistema. Para resolver este problema, foi utilizado um cálculo para desprezar os objetos detectados que estão fora da via que o semáforo controla.

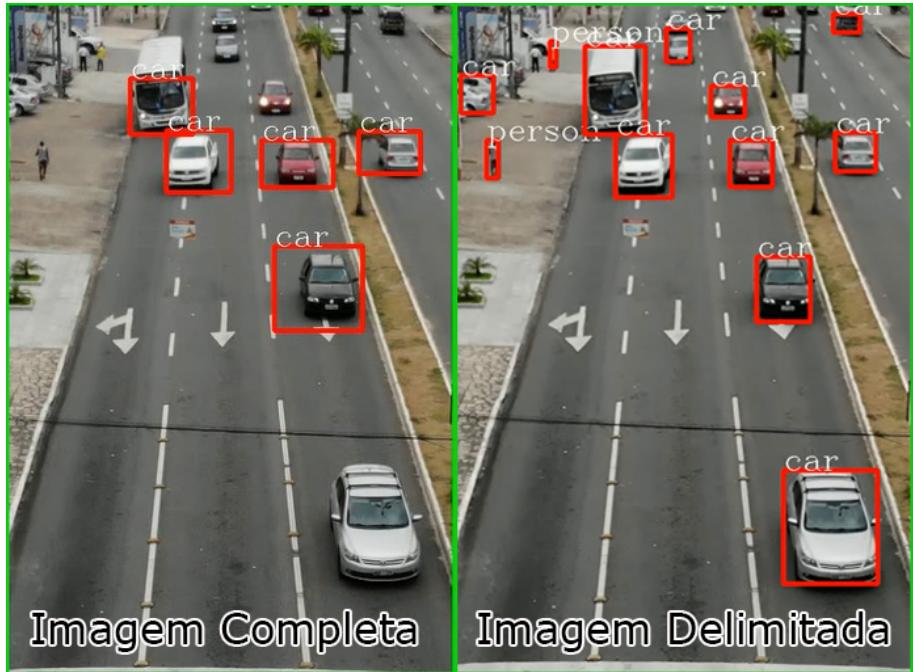
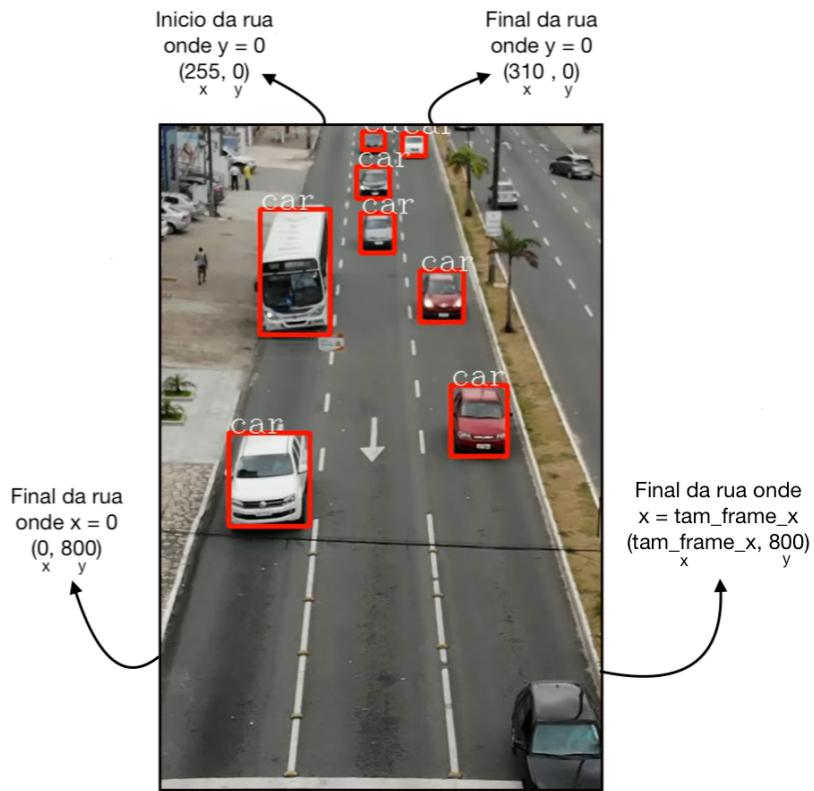


Figura 13: Diferença entre o processamento no vídeo completo e apenas na área delimitada. Os retângulos vermelhos mostram os veículos detectados. Podemos concluir que a detecção na área delimitada foi melhor, pois conseguiu detectar mais objetos da imagem.

Abaixo estão descritos os passos para fazer esse cálculo.

1. Para o lado esquerdo, define-se a coordenada x_1 onde a rua inicia no ponto $y = 0$ e a coordenada y_1 onde a rua inicia no ponto $x = 0$. Temos então as coordenadas $(x_1, 0)$ e $(0, y_1)$
2. Para o lado direito, define-se a coordenada x_2 onde a rua termina no ponto $y = 0$ e a coordenada y_2 onde a rua termina no ponto $x = 0$. Temos então as coordenadas $(x_2, 0)$ e $(0, y_2)$
3. Para predizer se dado a coordenada (x_3, y_3) em que um objeto foi detectado está dentro ou fora da delimitação de processamento, ou seja, se o objeto está dentro da rua, temos que definir para o lado esquerdo o cálculo da rua será a expressão $le = x_1 - ((x_1/y_1) * y_3)$. Da mesma forma, para o lado direito temos a expressão $ld = x_2 + ((x_2/y_2) * y_3)$.
4. Caso x_3 seja maior que le e menor que ld , podemos afirmar que o objeto detectado está dentro da área delimitada de processamento.

A Figura 14 mostra uma explicação do cálculo elaborado, exemplificando como adquirir as coordenadas necessárias para calcular a via.



$LE = 255 - ((255/800) * y_{do_objeto_detectado})$
 $LD = 310 + ((310/800) * y_{do_objeto_detectado})$
 Se ($LE < x_{do_objeto_detectado} < LD$) -> Veículo está na rua

Figura 14: Calculo elaborado para definir onde o processamento deve ser feito focando apenas a rua. Primeiramente deve-se definir as coordenadas de inicio e final da via no ponto $y = 0$, assim como o inicio e final da rua no ponto $y = onde\ a\ via\ termina$ dentro da caixa delimitadora que vai ser processada. Com essas coordenadas, podemos prever o que é rua a ser utilizada e o que deve ser desprezado.

A Figura 15 mostra uma comparação entre os resultados antes, mostrando todos os resultados encontrados, e depois do processamento tomando os objetos que estão dentro da área limitada pelo calculo descrito acima.

Após a definição de todas as coordenadas e valores de delimitação do lado direito e lado esquerdo para os Vídeos 1 e 2, o sistema foi executado mais uma vez. A Tabela 3 mostra os resultados para o Vídeo 1 e a Tabela 4 mostra os resultados para o Vídeo 2.

As Figuras 16 e 17 mostram o resultado final do processamento das imagens do Vídeo 1 para as duas vezes que o semáforo ficou vermelho, enquanto as Figuras 18 e 19 mostram o resultado final do processamento para o Vídeo 2.

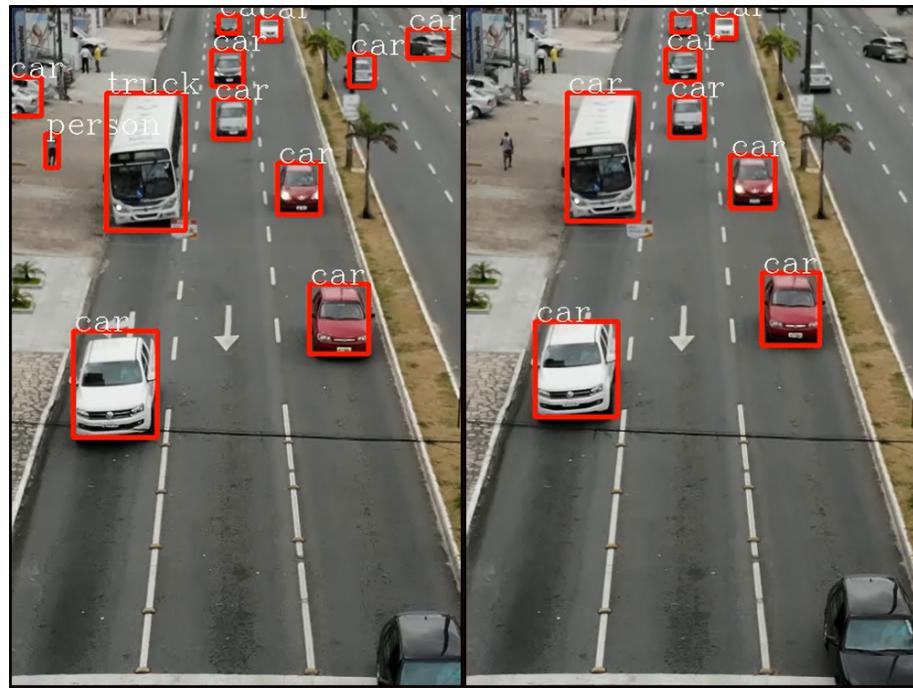


Figura 15: Diferença entre o processamento na área delimitada sem utilização do cálculo da via na imagem da esquerda, e exemplo do mesmo quadro com o cálculo desprezando os veículos fora da rua. Os retângulos vermelhos mostram os veículos detectados.

Tabela 3: Tabela mostrando para o Vídeo 1 depois da delimitação da área, a contagem do processamento, a quantidade de carros que deveriam ser detectadas e o erro.

| Parada | Contagem Proc | Contagem Real | Erro |
|--------|---------------|---------------|------|
| 1 | 3 | 4 | -1 |
| 2 | 6 | 6 | 0 |

Tabela 4: Tabela mostrando a contagem do processamento, a quantidade de carros que deveriam ser detectadas e o erro, para o Vídeo 2 com a área delimitada.

| Parada | Contagem Proc | Contagem Real | Erro |
|--------|---------------|---------------|------|
| 1 | 5 | 5 | 0 |
| 2 | 7 | 7 | 0 |



Figura 16: Parada 1 do Vídeo 1. Os retângulos vermelhos mostram os veículos detectados. Vemos que apenas um carro não pode ser detectado, pois ele está exatamente atrás do ônibus.



Figura 17: Parada 2 do Vídeo 1. Os retângulos vermelhos mostram os veículos detectados. Nesta imagem podemos ver que todos os veículos foram detectados corretamente.

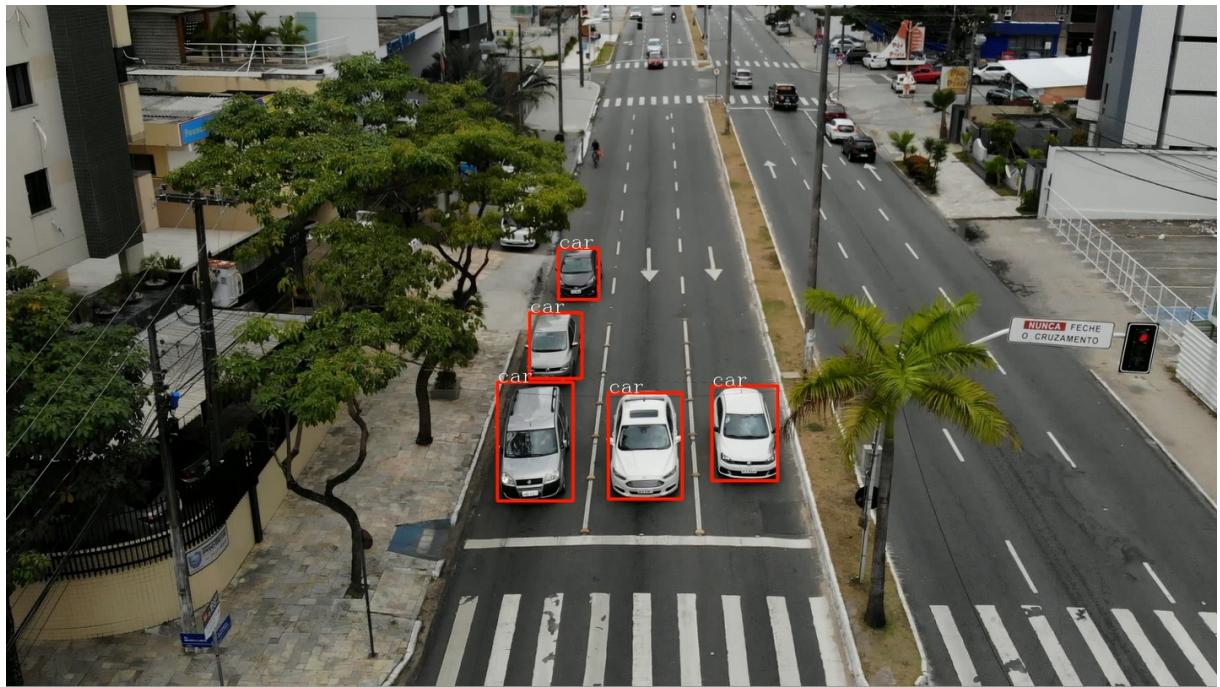


Figura 18: Parada 1 do Vídeo 2. Os retângulos vermelhos mostram os veículos detectados. Todos os carros estão sendo detectados corretamente.



Figura 19: Parada 2 do Vídeo 2. Os retângulos vermelhos mostram os veículos detectados. Pela imagem podemos concluir que todos os carros também estão sendo detectados corretamente.

Com os resultados descritos nas Tabelas 3 e 4, é possível ver que o erro no resultado do processamento em três das quatro análises, foram reduzidos a zero. O que mostra que a contagem de veículos foi precisa e correta. Entretanto, esses testes não foram feitos em tempo real, já que, ao tentar fazer este processamento, o desempenho foi muito baixo, uma vez que, utilizando os vídeos em resolução *Full-HD* com uma taxa de quadros de 60 FPS, o *framerate* ficou muito baixo, o que não permitiu que todos os quadros fossem processados em tempo real.

5.2 Avaliação de desempenho

Como descrito anteriormente, por causa do baixo desempenho, devido ao processador *Core i7* de *notebook* possuir baixa performance, e a falta de memória de vídeo, que neste caso é de 2 *Gigabytes*, a taxa quadros por segundo não foi suficiente para um processamento do vídeo em tempo real, uma vez que, a taxa de atualização é muito superior a taxa o qual o computador consegue processar, e ao utilizar uma câmera de qualidade superior, que filme em resolução *1920x1080* pontos com uma taxa de quadros de 60 FPS, a demanda de processamento é maior do que a oferecida pelo *notebook* utilizado nos testes. No teste feito no mundo real, utilizando um fluxo de vídeo obtido na internet, possuindo imagens oriundas de uma câmera com resolução de 800x600 pontos, situada em uma auto-estrada americana, nas quais a taxa de quadros por segundo variam entre 4 e 6 FPS, a situação é completamente diferente, possibilitando um processamento em tempo real. A Tabela 2 mostra os resultados do processamento para os Vídeos 1 e 2 em *Full-HD* e com a área delimitada apresentada na Seção 5.1.2. Além disso, mostra o teste feito com o fluxo de vídeo obtido na internet.

Tabela 5: Tabela mostrando as resoluções, quadros por segundo do vídeo, quadros por segundo do processamento e tempo de processamento. No fluxo de vídeo, o tempo de processamento não foi calculado (n/c), já que o processamento estava sendo feito em tempo real.

| Vídeo | Resolução | FPS Vídeo | FPS Proces | Tempo |
|-----------------------|-----------|-----------|------------|--------|
| Vídeo 1 | 1920x1080 | 60FPS | 6.4FPS | 43 Min |
| Vídeo 2 | 1920x1080 | 60FPS | 5.4FPS | 30 Min |
| Vídeo 1 delimitado | 457x688 | 60FPS | 7.1FPS | 35 Min |
| Vídeo 2 delimitado | 497x714 | 60FPS | 6.3FPS | 26 Min |
| Fluxo de vídeo | 640x480 | 8FPS | 6FPS | n/c |

Não foram feitos testes em um hardware com processador e GPU mais potentes, mas com o avanço da capacidade de processamento de placas de vídeo mais modernas,

o resultado do processamento para vídeos feitos em uma câmera de boa qualidade que filma em resolução de 1920x1080 pontos pode ser feito, caso sejam utilizados os pontos descritos na Seção 5.3.2 deste trabalho.

5.3 Problemas encontrados e suas soluções

5.3.1 Erros de detecção

Como neste trabalho, foram utilizados modelos pré-treinados de detecção e classificação, é inevitável que alguns erros sejam encontrados. No caso deste sistema, não são erros graves, como etiquetar um objeto carro (do inglês: *car*) como pessoa (do inglês: *person*), mas etiquetar, por exemplo, um caminhão (do inglês: *truck*) ou um ônibus (do inglês: *bus*) como carro, ou até mesmo uma moto (do inglês: *motorbike*) como bicicleta (do inglês: *bicycle*). A Figura 20 mostra alguns exemplos em que os objetos foram classificados de maneira errada.

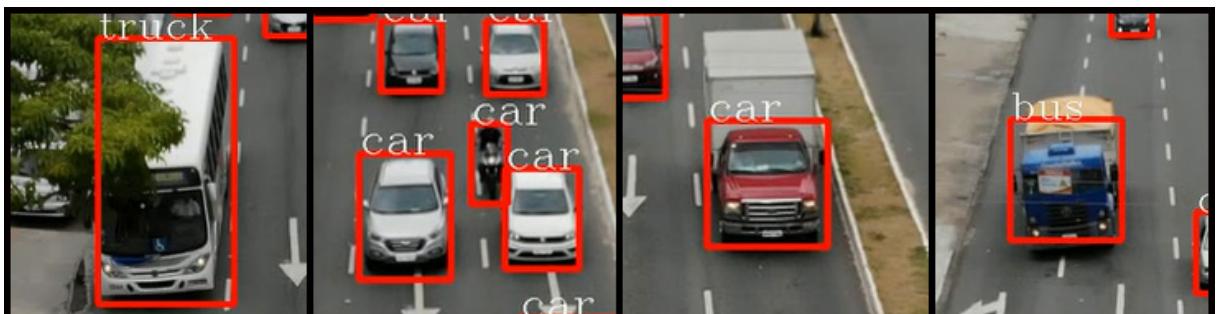


Figura 20: Da esquerda para direita: ônibus sendo detectado como caminhão, moto sendo detectada como carro, caminhão sendo identificado como carro e caminhão sendo etiquetado como ônibus. O retângulo vermelho indica os objetos detectados.

Para solucionar este problema, pode ser feito um modelo de treinamento específico, utilizando as imagens feitas pelo drone da perspectiva de um semáforo para capturar veículos que devem ser detectados. Desta forma, como o modelo para detecção seria específico, o desempenho e a precisão da detecção poderiam ser melhores.

5.3.2 Problemas de Hardware

O processamento de imagens em tempo real é muito caro, já que necessita de um *hardware* com processadores mais modernos e potentes, assim como GPUs com mais memória dedicada, para se obter um resultado satisfatório com taxas de quadro por segundo acima do natural para o olho humano, que é 24 FPS. Logo, como mostrado na

Seção 5.1, o processamento em tempo real de imagens em resolução a partir de 1920×1080 pontos dependendo do *hardware* ainda é muito lenta, onde, nos testes feitos, para um Vídeo de 2 minutos e 40 segundos de duração, o tempo de processamento foi de 26 minutos, tornando inviável o processamento ser feito diretamente pelo semáforo, uma vez que, neste caso, o hardware a ser utilizado teria de ser pequeno e barato, utilizando por exemplo uma *Raspberry pi* [18]. O problema é que um hardware pequeno e barato não tem condições de processar um vídeo de tamanha resolução.

Para solucionar este problema, podemos:

- Utilizar um hardware menor e mais barato nos semáforos em conjunto com uma câmera de boa qualidade que filme com resolução 1920×1080 pontos, e com acesso a internet, enviar as imagens da câmera para um servidor com hardware dedicado para processá-las e enviar de volta o resultado do processamento.
- Diminuir a taxa de quadros por segundo do vídeo a ser processado, uma vez que, a diferença entre os quadros de uma imagem por segundo nesta situação não varia a ponto de ser necessário fazer o processamento a 60 FPS e utilizando uma câmera de menor resolução, com no mínimo 1280×720 pontos.
- Processar apenas um quadro da imagem a cada 2 ou 3 segundos para ter uma estimativa de quantos carros estão na imagem.

6 CONCLUSÕES E TRABALHOS FUTUROS

Semáforos inteligentes é uma solução ótima para um problema recorrente em todas as cidades do mundo. Junto do avanço da tecnologia, os sistemas devem ficar mais rápidos e eficientes, com processamento de vídeos de melhor qualidade em tempo real e capacidade de processar as informações de múltiplos semáforos simultaneamente. Portanto, neste trabalho foi proposto um sistema simples que utiliza *Deep Learning* e processamento de imagens para detectar veículos em um fluxo de vídeo de uma câmera instalada em semáforos, para que a partir da análise dessas imagens, possa ser feito o controle de abertura e fechamento dos sinais de trânsito.

Como vimos nos resultados, o comportamento do sistema com o processamento feito na área delimitada não gerou quase nenhum problema na contagem dos veículos quando o semáforo está fechado, levando o erro a 0 na maioria dos casos, exceto por apenas um caso em que o automóvel ficou atrás de um ônibus, fazendo com que o sistema não o detectasse.

Como os testes foram feitos em um *hardware* com processador de baixo desempenho e pouca memória de vídeo, como mostrado na Seção 4.1.2 deste trabalho, o processamento em tempo real dos vídeos feitos pelo drone não foram satisfatórios em virtude da baixa taxa de quadros por segundos. Entretanto, como mostrado na Seção 5.3.2, este problema pode ser solucionado utilizando um *hardware* mais potente, aumentando o custo do sistema, ou utilizando câmeras de menor resolução com taxas de quadros por segundo menores, viabilizando assim o uso deste sistema inclusive em *hardwares* menores e mais baratos.

Esse sistema ainda está em um estágio inicial, portanto o intuito deste trabalho foi apenas descobrir a viabilidade e fazer uma implementação simples. Com os resultados obtidos, podemos concluir que apesar de alguns problemas de classificação na detecção dos veículos, esse problema não afetou a contagem, que é o ponto mais importante do resultado do processamento da imagem, pois vai permitir o controle preciso do fluxo de automóveis dos semáforos.

6.1 Trabalhos futuros

Por ser uma área ampla, e ser um trabalho inicial, este sistema permite uma vasta gama de opções para incremento. De imediato, a primeira modificação é utilizar o processamento de dois semáforos e criar uma heurística de controle para decidir qual deles ficará aberto e por quanto tempo. Como inicialmente o objetivo é deixar o semáforo responsável apenas por capturar as imagens e enviar para um servidor processá-las, é preciso criar um protocolo de comunicação entre este servidor e o semáforo, para que ele possa mandar comandos de abertura e fechamento através do resultado da análise das

imagens.

Além disso, criar uma rede interconectada de semáforos inteligentes que conseguem prever o fluxo de automóveis se deslocando de um outro semáforo também é interessante, pois com essa análise o sinal poderia usar todo o conjunto de dados para ajudar no cálculo do tempo e até aprender quais os horários de maior quantidade de veículos, para otimizar o tempo de parada entre todos estes semáforos.

Uma outra aplicação para esse sistema seria, além de contar os veículos no semáforo, verificar as placas para buscar por carros roubados junto ao sistema dos órgãos de trânsito do estado. Também poderia, em conjunto com um sensor de velocidade, calcular a velocidade dos carros aplicando multas para os veículos acima da velocidade permitida, de forma mais discreta sem a necessidade de instalar redutores de velocidade em cada ponto da via, diminuindo assim o custo para o governo.

Ademais, esse sistema também seria útil para fazer o monitoramento de vias, como por exemplo, verificando veículos estacionados em local proibido, aplicando multas para os veículos em situação irregular, detecção de ambulâncias e veículos da polícia em atividade, para manter os semáforos das ruas sempre abertos em casos de emergência, comunicando os outros semáforos.

REFERÊNCIAS

- [1] GONZALES, Rafael C; WOODS, Richard E. “**Digital Image Processing**”. Estados Unidos: Pearson Education, 2008.
- [2] CAPONETTI, Laura; CASTELLANO, Giovanna. “**Fuzzy Logic for Image Processing**”. Itália: Springer, 2017.
- [3] ABADI Martín et al. “**TensorFlow: Large-scale machine learning on heterogeneous systems**”, 2015. Software available from tensorflow.org.
- [4] REDMON, Joseph et al. “**You Only Look Once: Unified, Real-Time Object Detection**”. 2016. Disponível em: <https://arxiv.org/pdf/1506.02640.pdf>. Acesso em: 29 set. 2018.
- [5] REDMON, Joseph; FARHADI, Ali. “**YOLO9000: Better, Faster, Stronger**”. 2016. Disponível em: <https://arxiv.org/pdf/1506.02640.pdf>. Acesso em: 29 set. 2018.
- [6] PRICE, Sarah. “Medium Level Image Processing”. 1996. Disponível em: <http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCALCOPIES/MARBLE/medium/medium.htm>. Acesso em: 06 out. 2018.
- [7] MARTINET, Jean; ELSAYAD, Ismail. “**Mid-level image descriptors**”. In: YAN, Li; MA, Zongmin. Intelligent Multimedia Databases and Information Retrieval: Advancing Applications and Technologies. [S.l.]: IGI Global, 2011. cap. 03, p. 46-62.
- [8] GAO, Junfeng et al. “**Computer Vision in Healthcare Applications**”. Journal of Healthcare Engineering 2018 (2018): 5157020. PMC. Web. 10 Oct. 2018.
- [9] MA, Alexandra. “China has started ranking citizens with a creepy ‘social credit’ system — here’s what you can do wrong, and the embarrassing, demeaning ways they can punish you”. Disponível em: <https://www.businessinsider.com/china-social-credit-system-punishments-and-rewards-explained-2018-4>. Acesso em: 13 out. 2018.
- [10] STERGIOU, Christos; SIGANOS, Dimitrios. “Neural Networks”. Disponível em: <https://www.doc.ic.ac.uk/nd/surprise-96/journal/vol4/cs11/report.html#What-20is-20a-20Neural-20Network>. Acesso em: 13 out. 2018.
- [11] MORRIS, David. “Smart cars, meet smart signals”. Disponível em: <http://fortune.com/2015/08/20/smart-traffic-signals/>. Acesso em: 02 out. 2018.
- [12] “PITTSBURGH cuts travel time by 25% with smart traffic lights: The smart traffic technology has slashed car emissions by 20%”. Disponível em: <https://apolitical.co/solution-article/pittsburgh-cuts-travel-time-25-smart-traffic-lights>. Acesso em: 02 out. 2018.

- [13] “SURTRAC Deployment at Urban Grid Networks in Pittsburgh Neighborhoods”. Disponível em: <https://www.rapidflowtech.com/blog/surtrac-deployment-at-urban-grid-networks-in-pittsburgh-neighborhoods>. Acesso em: 02 out. 2018.
- [14] ALECRIM, Emerson. **Semáforos realmente inteligentes estão chegando:** Aos poucos, mas estão. O sistema da Audi que “conversa” com semáforos é um bom começo.. Disponível em: <https://tecnoblog.net/199957/semaforos-realmente-inteligentes>. Acesso em: 02 out. 2018.
- [15] SANTOS, Vinicius. **REDES NEURAIS - USP - INTRODUÇÃO A REDES NEURAIS.** Disponível em: <http://www.computersciencemaster.com.br/2018/01/usp-introducao-redes-neurais.html>. Acesso em: 25 out. 2018.
- [16] EVERINGHAM, Mark et al. **The PASCAL Visual Object Classes (VOC) Challenge.** 2009. Disponível em: <http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham10.pdf>. Acesso em: 27 out. 2018.
- [17] COCO - Common Objects in Context. 2018. Disponível em: <http://cocodataset.org/#home>. Acesso em: 27 out. 2018.
- [18] RASPBERRY Pi. 2018. Disponível em: <https://www.raspberrypi.org>. Acesso em: 27 out. 2018.
- [19] THE MAN who gave us traffic lights: We must thank a Nottingham engineer for making drivers stop and go at busy road junctions. Disponível em: <http://www.bbc.co.uk/nottingham/content/articles/2009/07/16/john-peake-knight-traffic-lights-feature.shtml>. Acesso em: 30 out. 2018.
- [20] GARDNER, Andrew. **A Brief History of Traffic Lights.** Disponível em: <https://www.artsy.net/article/artsy-editorial-history-traffic-lights>. Acesso em: 30 out. 2018.
- [21] IMAGENET. Disponível em: <http://www.image-net.org>. Acesso em: 30 out. 2018.
- [22] KANUNGO, Anurag; SHARMA, Ayush; SINGLA, Chetan. Smart Traffic Lights Switching and Traffic Density Calculation using Video Processing. 2014. Disponível em: <https://www.researchgate.net/profile/Anurag-Kanungo/publication/269310721-Smart-traffic-lights-switching-and-traffic-density-calculation-using-video-processing/links/563e5ec108ae45b5d28c563a.pdf>. Acesso em: 11 nov. 2018.