UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA

Aplicação de Visão Computacional no Auxílio ao Levantamento de Defeitos em Pavimento Rodoviário

Marcos Meyer Hollerweger

Orientador: Eduardo Luiz Ortiz Batista

Florianópolis, Julho de 2019.

MARCOS MEYER HOLLERWEGER

APLICAÇÃO DE VISÃO COMPUTACIONAL NO AUXÍLIO AO LEVANTAMENTO DE DEFEITOS EM PAVIMENTO RODOVIÁRIO

Trabalho de Conclusão de Curso submetido ao curso de Engenharia Eletrônica da Universidade Federal de Santa Catarina como requisito para aprovação da disciplina EEL7806 - Trabalho de Conclusão de Curso (TCC).
Orientador: Eduardo Luiz Ortiz Batista.

FLORIANÓPOLIS
2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Hollerweger, Marcos Meyer
Aplicação de Visão Computacional no Auxílio ao Levantamento de Defeitos em Pavimento Rodoviário / Marcos Meyer Hollerweger ; orientador, Eduardo Luiz Ortiz Batista, 2019.
61 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2019.

Inclui referências.

1. Engenharia Eletrônica. 2. Aprendizado de máquina. 3. Visão computacional. 4. Detecção de objetos. 5. Rodovias. I. Batista, Eduardo Luiz Ortiz. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Marcos Meyer Hollerweger

APLICAÇÃO DE VISÃO COMPUTACIONAL NO AUXÍLIO AO LEVANTAMENTO DE DEFEITOS EM PAVIMENTO RODOVIÁRIO

Este Trabalho de Conclusão de Curso foi julgado adequado no contexto da disciplina EEL7890 - Trabalho de conclusão de Curso (TCC), e aprovado em sua forma final pelo Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina.

Florianópolis, Julho de 2019.

Prof. Jefferson Luiz Brum Marques, Ph.D.
Coordenador do Curso

Prof. Eduardo Luiz Ortiz Batista, Ph.D.
Orientador

Banca examinadora:

Prof. Richard Demo Souza, Sc.D.
Universidade Federal de Santa Catarina

Prof. Danilo Silva, Ph.D.
Universidade Federal de Santa Catarina

Prof. Joceli Mayer, Ph.D.
Universidade Federal de Santa Catarina

Agradecimentos

Agradeço a meus pais Marcos e Jussara, e ao meu irmão Guilherme pelo apoio durante a graduação.
Ao orientador desse trabalho, Eduardo, e a todos os professores do Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina por todo o conhecimento compartilhado.
A Pedro, Lucas e João, por fomentarem a ideia central que levou ao desenvolvimento desse trabalho.
A Caroline, Rafael, Larah e Vitor, pelos divertidos almoços.
A Hanna, pelo companheirismo.
A Paul e Rosanna, por todas as oportunidades de melhoria pessoal e profissional propiciadas, e a todos os amigos e colegas do curso de Engenharia Eletrônica da Universidade Federal de Santa Catarina que de alguma forma participaram do processo de aprendizado ao longo da jornada.

RESUMO

Este trabalho é focado no estudo da viabilidade e eficácia de estratégias de visão computacional aplicadas a sistemas de auxílio ao levantamento de defeitos em pavimento rodoviário. A ideia central é aumentar o nível de automatização e assertividade de tais sistemas. Nesse contexto, são abordadas redes neurais para classificação e detecção de objeto, com foco nas topologias ResNet (*Residual Neural Network*) e YOLO (*You Only Look Once*). A situação específica a ser estudada é a identificação, dentre um conjunto de imagens de pavimento obtidas a partir de uma câmera em um veículo em movimento, das imagens que mostram defeitos no pavimento. O principal defeito a ser estudado é o buraco (ou panela). A tecnologia desenvolvida pode melhorar a assertividade na obtenção de imagens dos buracos na estrada, podendo significar um avanço substancial nos processos utilizados hoje no levantamento desse tipo de defeito, principalmente em grandes áreas (como as estradas do Brasil).

Palavras-chave: Aprendizado de máquina. Visão computacional. Buracos. Classificadores binários. Detecção de objeto. Rodovias. Concessionárias de estradas.

ABSTRACT

This is a study about the effectiveness of computer vision strategies applied to electronic systems that aid the finding of pathologies in highway pavement, with the main goal of obtaining an increase in the levels of automation and assertiveness of these systems. Neural networks for binary classification and object detection are addressed, focusing on the ResNet (Residual Neural Network) and YOLO (*You Only Look Once*) topologies. The particular situation that this study focuses on is the identification, among a set of pavement pictures obtained from a camera on the inside of a moving car, of the images that better represent pavement pathologies. The pathology being studied is the pothole. The developed technology could increase the usefulness of such aiding systems by providing a substantial improvement in the accuracy of pathology pictures obtained from a moving car, thus reducing the time spent in the processes to find these types of defect. Results could be even more noticeable when dealing with large areas, such as the roads of a big country like Brazil.

Keywords: Machine learning. Computer vision. Potholes. Binary classification. Object detection. Highways. Toll road.

Lista de Figuras

Lista de Tabelas

Sumário

CAPÍTULO 1

Introdução

No Brasil, o modal predominante de transportes é o rodoviário. No ano de 2011, 52% do transporte de cargas e 96% do transporte de passageiros foi realizado utilizando tal modal. Parte essencial para a efetividade desse tipo de transporte são as rodovias. No Brasil, ao final de 2017, a malha pavimentada tinha extensão de cerca de 213 mil km, sendo que 9,24% destes estão concedidos a empresas privadas [1].

A concessão de rodovias no Brasil tem garantido o investimento e a manutenção constantes, sendo aplicada em trechos estratégicos para o desenvolvimento da infraestrutura do país. As rodovias que fazem parte de programas de concessão costumam ter alto fluxo de veículos, e por consequência, rápido desgaste do pavimento [2].

As empresas responsáveis pelas rodovias sob concessão (as concessionárias) devem seguir normas previstas no contrato de concessão, e são assim obrigadas a manter o pavimento dentro de um rígido padrão de qualidade. O estado do pavimento é constantemente avaliado, tanto pelas concessionárias quanto pelo órgão concedente, e caso esteja fora do padrão, as empresas podem ser multadas em valores milionários. Além do impacto financeiro, a manutenção do pavimento rodoviário também possui um importante papel social, já que milhões de pessoas

trafegam por estas estradas diariamente, e a qualidade do pavimento tem direta influência no número de acidentes ocorridos [3].

1.1 Avaliação da superfície de pavimentos

Existem diversos tipos de ensaios e métricas de avaliação que podem ser aplicados na análise da superfície de um determinado pavimento. Em geral, a escolha do ensaio/métrica está relacionada ao ambiente no qual o pavimento se encontra. Como um exemplo, as ruas de uma pequena cidade possuem infraestrutura diferente de uma rodovia de alto fluxo. Isso afeta as patologias que o pavimento desenvolverá ao longo do tempo, e por consequência, os ensaios realizados para identificá-las.

A patologia que mais causa impacto aos usuários de uma estrada é o buraco (ou panela), o qual é demonstrado na Figura 1.1. Apesar do grande risco que os buracos apresentam à integridade física dos veículos que trafegam por estradas com alta incidência dos mesmos, tais irregularidades são, em geral, facilmente identificáveis através de inspeção visual.

Figura 1.1: Exemplo de buraco em pavimento rodoviário.

1.1.1 Levantamento visual contínuo

Uma das estratégias utilizadas para a identificação de buracos (entre outras patologias) é o Levantamento Visual Contínuo. Tal procedimento é regulamentado pelo Departamento Nacional de Infraestrutura

de Transportes (DNIT) [4] e, portanto, executado com frequência por todas as concessionárias de rodovias brasileiras. A norma padroniza que o procedimento deve ser realizado a partir de um carro em velocidade média de 40 km/h percorrendo a rodovia em um único sentido. A equipe deve ser constituída por no mínimo dois técnicos, além do motorista do veículo. Ainda é estabelecido que deve ser evitada a realização de levantamentos em dias chuvosos, e que o levantamento deve ser feito em segmentos de 1 km. As condições definidas pela norma serão utilizadas como premissas na situação estudada neste trabalho.

1.1.2 Índice de condição dos pavimentos flexíveis e semi-rígidos (ICPF)

O ICPF é um índice utilizado para avaliar a condição da superfície do pavimento. Ele leva em consideração a frequência de patologias em um determinado segmento de 1 km de extensão, incluindo buracos.

Conversas com integrantes de equipes técnicas de concessionárias revelaram que o procedimento do levantamento visual contínuo é realizado para obter os números dos índices, mas estes não são suficientes para realizar as correções. Posteriormente, é necessário um outro levantamento mais detalhado, parando o veículo em cada ocorrência de patologia encontrada. Nesta segunda etapa, são catalogadas informações sobre o tamanho do buraco, localização e registro fotográfico, que serão encaminhados para a equipe que realizará a intervenção corretiva.

1.2 Sistemas de auxílio ao levantamento

Existem sistemas disponíveis no mercado que buscam auxiliar o levantamento visual contínuo ou expandir a gama de dados obtidos a partir de tal procedimento. Um tipo de solução consiste na instalação de um dispositivo embarcado no veículo cuja principal interface com o usuário seja um botão. Ao visualizar uma ocorrência da patologia, um dos técnicos presentes no carro pressiona o botão, realizando um registro georreferenciado daquela ocorrência [5]. Posteriormente, é possível seccionar os segmentos de 1 km e calcular as respectivas frequências de patologias de forma automática, dispensando a utilização do odômetro e velocímetro do veículo para esta tarefa. Otimizações dessa natureza poderiam diminuir o custo do procedimento ao dispensar a presença de

um dos técnicos dentro do veículo.

A adição de uma câmera ao sistema embarcado descrito permitiria a obtenção do registro fotográfico das ocorrências sem a necessidade de um segundo levantamento com paradas. Entretanto, uma dificuldade adicional seria imposta ao técnico, uma vez que o acionamento do sistema teria que ser feito quando o carro estivesse em posição favorável para o enquadramento do buraco na imagem. Uma estratégia para resolver tal dificuldade, seria a implementação de um *buffer* capaz de armazenar imagens dos últimos metros percorridos. Nesse caso, ao ocorrer o acionamento do sistema, ainda seriam coletadas algumas imagens dos metros seguintes. Uma solução eficiente consistiria na utilização de um sistema de visão computacional para processar esse conjunto de imagens, visando determinar a que tem maior chance de conter um bom enquadramento do buraco observado.

1.3 Objetivos

Neste trabalho serão estudadas técnicas de aprendizado de máquina que possam ser implementadas em um sistema embarcado e possibilitem a escolha, dentre um conjunto de imagens obtidas a partir de um carro em movimento, desde instantes anteriores à passagem do veículo pela região de ocorrência de uma patologia no pavimento (buraco) até momentos após, da imagem que contém tal patologia em enquadramento adequado.

1.3.1 Objetivos Específicos

As seguintes etapas serão essenciais para atingir o objetivo citado:

- Encontrar um conjunto de dados contendo imagens de buracos em pavimento rodoviário, que esteja disponível online e possua anotações tanto binárias (existência de buraco na imagem ou não) quanto referentes à localização dos buracos (*boxes*);
- Elencar algoritmos populares de visão computacional que possam ser utilizados nas tarefas de classificação binária e detecção de objeto;
- Através da técnica de aprendizado por transferência e processo de ajuste fino ao conjunto de dados de interesse, obter algoritmos

de classificação e detecção de objetos que possam ser utilizados no contexto abordado nesse trabalho;

- Definir as métricas de avaliação de desempenho que serão utilizadas para escolher o melhor modelo dentre os estudados;

- Avaliar o impacto de fatores ambientais (iluminação, sombras e condições climáticas) sobre o desempenho de classificação e detecção dos algoritmos estudados;

- Indicar possíveis continuações futuras do trabalho.

1.4 Trabalhos prévios

Diversos trabalhos abordaram problemas similares aos considerados no presente trabalho, realizando abordagens tanto de classificação quanto de detecção de objetos para identificação de defeitos no pavimento. A seguir, alguns desses trabalhos são descritos em conjunto com a descrição de aspectos em que este trabalho difere deles.

O trabalho desenvolvido por NIENABER, S et al. [6] aborda a classificação binária, porém utilizando apenas métodos de processamento de imagem, sem envolver aprendizado de máquina.

MAEDA, H et al. [7] abordaram o problema da avaliação da superfície do pavimento com aprendizado de máquina, porém o conjunto de dados utilizado em tal trabalho tem foco em trincamentos e outros tipos de defeito, contendo uma quantidade muito pequena de imagens de buracos.

Também existem abordagens que não utilizam visão computacional, como a proposta por MEDNIS, A et al. [8], onde o algoritmo de aprendizado de máquina trabalha apenas com dados obtidos a partir do acelerômetro de um *smartphone*. Tal estratégia, no entanto, só é eficaz quando um dos pneus do veículo de fato atinge o buraco, não sendo capaz de identificar defeitos no restante da faixa de rolagem.

FORNARI, P [9] traz uma abordagem combinando a utilização de acelerômetro com sensores ultrassônicos montados na parte inferior do veículo, buscando cobrir toda a faixa de rolagem. Essa estratégia, entretanto, não permite obter imagens do defeito para utilização posterior pelas equipes envolvidas.

1.5 Estrutura do trabalho

Os demais capítulos deste trabalho abordarão os seguintes tópicos:

- No Capítulo 2 será apresentada a fundamentação teórica a respeito de *Machine Learning* e Visão Computacional, assim como alguns algoritmos e topologias de redes neurais que serão utilizados;
- O Capítulo 3 tratará sobre o conjunto de dados utilizado, o pré-tratamento do mesmo, os experimentos a serem realizados e as métricas de avaliação;
- O Capítulo 4 mostrará os resultados obtidos a partir dos experimentos descritos no Capítulo 3;
- O Capítulo 5 apresentará um estudo de viabilidade da implantação dos algoritmos estudados em um sistema embarcado, focando nas limitações da câmera e processador de dispositivos disponíveis no mercado;
- O Capítulo 6 discorrerá sobre conclusões obtidas a partir dos resultados, e fará sugestões sobre trabalhos futuros com base nas dificuldades e limitações encontradas durante o desenvolvimento deste.

CAPÍTULO 2

Fundamentação Teórica

No campo acadêmico, as primeiras pesquisas sobre inteligência artificial datam de 1956. Porém, foi apenas no final da década de 1990 que grandes sucessos foram obtidos em aplicações práticas, principalmente devido ao aumento da capacidade computacional disponível e acessível. Outro vetor que possibilitou o sucesso foi o foco em resolver problemas específicos, ao invés de apenas tentar reproduzir o pensamento humano de forma artificial [10].

Este capítulo busca fazer uma breve introdução do leitor aos conceitos teóricos que compõem as estratégias relacionadas a inteligência artificial utilizadas neste trabalho.

2.1 Visão computacional

Visão computacional pode ser definida como a área que compreende o conjunto de processos (aquisição, processamento, análise e compreensão) aplicados a imagens digitais, com objetivo de automatizar tarefas que necessitam da percepção proporcionada pela visão humana. Dentre as técnicas envolvidas estão o simples processamento de imagens (transformação, codificação, transmissão) e o reconhecimento de pa-

drões estatísticos [11] - modernamente conhecido como aprendizado de máquina.

2.2 Aprendizado de máquina

Aprendizado de máquina pode ser definido como o campo de estudo de algoritmos e modelos estatísticos que computadores utilizam para executar tarefas sem serem explicitamente programados, mas através do seguimento de padrões ou utilização de inferência [12].

De um ponto de vista matemático-estatístico, podemos considerar que, dado um conjunto de ocorrências Y, um número p de preditores $X_1, X_2, ..., X_p$, existe uma relação entre Y e $X = (X_1, X_2, ..., X_p)$ tal que

$$Y = f(X) + \epsilon. \tag{2.1}$$

Aqui, f é uma função desconhecida de $X_1, X_2, ..., X_p$, e ϵ é uma parcela de erro aleatória, de média nula. A função f representa a informação sistemática que X contém sobre Y, ou seja, como as variações em X influenciam o comportamento de Y [13]. As principais razões pelas quais é interessante encontrar f são:

- **Predição** - situação que ocorre quando o conjunto de preditores X está disponível, e desejamos obter predições dos valores de Y. Nesses moldes, podemos escrever que

  $$\hat{Y} = \hat{f}(X), \tag{2.2}$$

  onde $\hat{f}$ representa uma estimativa de f, e $\hat{Y}$ representa uma estimativa de Y. Este é o caso de interesse para este trabalho.

- **Inferência** - situação em que o objetivo é entender melhor a relação entre X e Y. Uma aplicação pode ser o desejo de conhecer quais preditores apresentam maior influência no resultado.

Existem diversos métodos matemáticos que podem ser utilizados na inferência da função f [13], porém este trabalho não os abrangerá. É válido citar que os problemas em que amostras de X e Y são conhecidas e podem ser utilizadas para encontrar a função f são chamados

de aprendizado supervisionado. Há outros casos, denominados aprendizado não-supervisionado, em que deseja-se inferir alguma informação a respeito da distribuição de X sem de fato possuir informações sobre Y, como o agrupamento de amostras que apresentam similaridade em algum preditor de X. As práticas utilizadas neste trabalho são apenas de aprendizado supervisionado.

2.2.1 Regressão e Classificação

Os problemas de aprendizado de máquina podem ser divididos em dois grandes grupos:

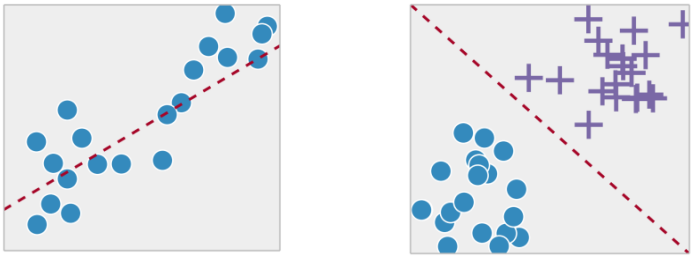
- **Regressão** - quando as ocorrências Y envolvem variáveis contínuas, como a idade de uma pessoa, ou o preço de uma casa. Um exemplo de proposta de solução simplificada para um problema de regressão está demonstrado na Figura 2.1a, que utiliza uma aproximação linear para descrever as ocorrências.

- **Classificação** - quando as ocorrências Y são categóricas, ou seja, precisam ser enquadradas em classes discretas pré-definidas. Um exemplo é o problema abordado nesse trabalho, que busca classificar imagens entre as classes (1) possui um buraco na estrada e (2) não possui um buraco na estrada. Tais problemas que contém duas classes também são conhecidos como problemas de classificação binária.

  A Figura 2.1b demonstra uma proposta de solução simplificada para um problema de classificação binária, através de uma aproximação linear da fronteira entre as ocorrências de classes diferentes.

2.2.2 Detecção de Objeto

Para obter completo conhecimento a respeito do conteúdo de um conjunto de imagens, muitas vezes não basta apenas classificá-las com base em alguma característica particular, é necessário estimar a localização dos objetos presentes em cada imagem. Essa tarefa é conhecida como detecção de objeto [14]. A Figura 2.2 [15] demonstra um exemplo do que se deseja alcançar, costumeiramente, com um algoritmo de detecção de objeto.



(a) Regressão. (b) Classificação binária.

Figura 2.1: Exemplos de problemas de aprendizado de máquina.

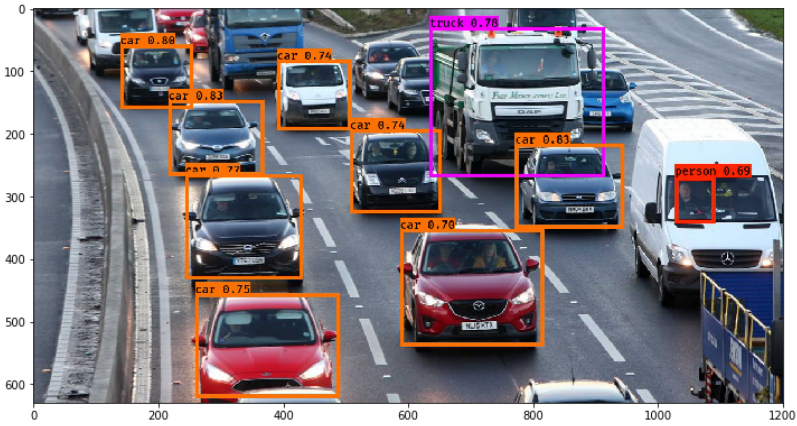


Figura 2.2: Exemplo de detecção de objeto.

O problema de detecção de objeto pode ser dividido em três etapas básicas [14]:

- **Seleção da região informativa** - trata da escolha da posição e formato da região que contém um objeto. Este problema é complexo pois os objetos de interesse podem estar em qualquer lugar da imagem e ter tamanho e formato variável. Varrer toda a imagem com filtros de diferentes tamanhos e formatos é uma possibilidade pouco eficiente devido à complexidade computacional, enquanto aplicar uma pequena quantidade de filtros pode trazer resultados imprecisos.

- **Extração de características** - trata da extração e reconhecimento de características presentes nos objetos que permitam uma representação robusta do mesmo. A principal dificuldade está associada às diferentes condições de iluminação e situação (plano de fundo) em que o objeto pode estar inserido, que tornam complexo o projeto de um extrator de características que descreva perfeitamente qualquer objeto em qualquer situação.

- **Classificação** - trata do algoritmo que, a partir da região informativa e das características extraídas, é capaz de identificar, dentre um conjunto pré-definido de classes, à qual classe o objeto pertence. Qualquer problema de detecção de objeto deve conter pelo menos duas classes, sendo uma delas o plano de fundo da imagem, e a outra, a classe de interesse. Este é o caso do problema abordado no presente trabalho.

2.3 Redes Neurais e *Deep Learning*

De modo simplificado, uma rede neural artificial — ou simplemente rede neural — é uma estrutura de processamento de dados com estrutura inspirada no sistema nervoso humano, onde determinados estímulos de entrada culminam em uma ou mais saídas [16].

A Figure 2.3 [17] demonstra um exemplo de topologia de rede neural, onde as unidades (ou neurônios) amarelos, à esquerda, representam a camada de entrada, os neurônios verdes representam as camadas internas, e os neurônios laranjas representam a camada de saída.

As unidades básicas das redes neurais, também conhecidos como neurônios ou *perceptrons*, são elementos compostos a partir das seguintes operações matemáticas básicas [18]:

- Multiplicação das entradas $X_1, X_2, ..., X_n$ por pesos $w_1, w_2, ..., w_n$;

- Combinação linear dos sinais de entrada ponderados pelos pesos e somados a coeficientes de *bias*;

- Introdução de não-linearidade através da aplicação de função não-linear (também conhecida como função de ativação).

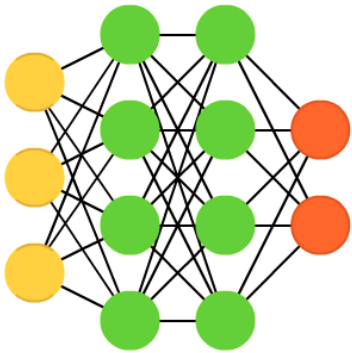


Figura 2.3: Exemplo de topologia de rede neural.

Essas operações básicas podem ser resumidas através da fórmula

$$\mathbf{y} = g(\mathbf{X}\mathbf{w} + \mathbf{b}), \tag{2.3}$$

onde $\mathbf{X}$ representa as entradas, $\mathbf{w}$ os pesos, $\mathbf{b}$ os coeficientes de *bias*, $\mathbf{y}$ a saída, e g a função não-linear. Um diagrama ilustrativo pode ser visualizado na Figura 2.4.

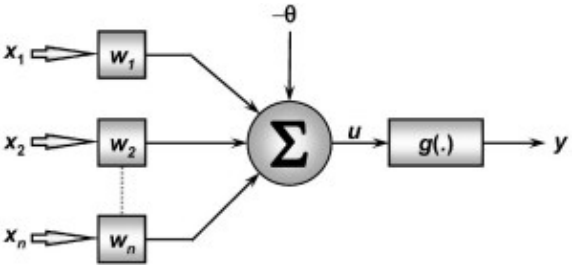


Figura 2.4: Representação da estrutura de um neurônio.

Os primeiros trabalhos com redes neurais consistiam de topologias similares à representada na Figura 2.3. É importante notar que, nessa representação, cada um dos neurônios das camadas internas está conectado a todos os neurônios das camadas adjacentes — camadas estruturadas de tal forma são denominadas densas ou totalmente conectadas. As redes neurais que possuem um grande número de camadas internas

são chamadas de redes neurais profundas, e representam um avanço significativo que levou aos resultados impressionantes obtidos recentemente com inteligência artificial. *Deep Learning*, a área dedicada ao estudo das redes neurais profundas, tem origens antigas assim como o aprendizado de máquina de forma geral. Porém, tal área se tornou popular na última década devido ao advento de algoritmos mais eficientes para o treinamento de redes neurais, e da disponibilidade de poder computacional para executá-los com redes mais profundas.

Diversos outros tipos de camadas existem, de acordo com a aplicação da rede neural. No campo de visão computacional, há uma dificuldade imposta na escalabilidade de camadas densas conforme o aumento da resolução das imagens do conjunto de dados. Imaginando um exemplo com imagens quadradas, com 200 *pixels* de altura, 200 *pixels* de largura e 3 canais de cor (RGB), cada imagem é composta por 120000 variáveis de informação. A análise de uma imagem desse tipo com uma camada densa requer, no mínimo, os mesmos 120000 pesos — um para cada variável de entrada — tornando complexo (e caro, computacionalmente) o treinamento de tal rede [19]. Além disso, a natureza unidimensional das camadas densas causaria a perda da informação espacial da imagem, que costuma ter uma característica bi-dimensional (para o caso de imagens em escala de cinza) ou tri-dimensional (imagens coloridas e codificadas no sistema RGB).

Em geral, a extração de características de uma imagem deve levar em consideração a espacialidade de uma entrada, já que a informação de um *pixel* costuma estar relacionada à informação dos *pixels* adjacentes. Logo, a solução proposta é a organização das camadas de neurônios em três dimensões, que podem ser imaginados como uma "pilha" de filtros bi-dimensionais. Cada um desses filtros tem tamanho de uma fração da imagem, por exemplo, 16x16, e varrerá as diferentes regiões da imagem em busca de uma característica. A terceira dimensão trata da possibilidade de utilizar diversos filtros em uma mesma camada. O problema de escalabilidade também é resolvido, já que o número de pesos escala apenas com o tamanho do filtro, e não com o tamanho total da imagem. Essas camadas são conhecidas como camadas convolucionais, devido à natureza da operação de varredura do filtro pela imagem.

Para facilitar a compreensão desse tipo de camada, e também de como funciona a extração de características de imagens em geral, pode-

mos utilizar um exemplo do reconhecimento de caracteres manuscritos. A Figura 2.5 [20] demonstra esse problema com a letra “X”.

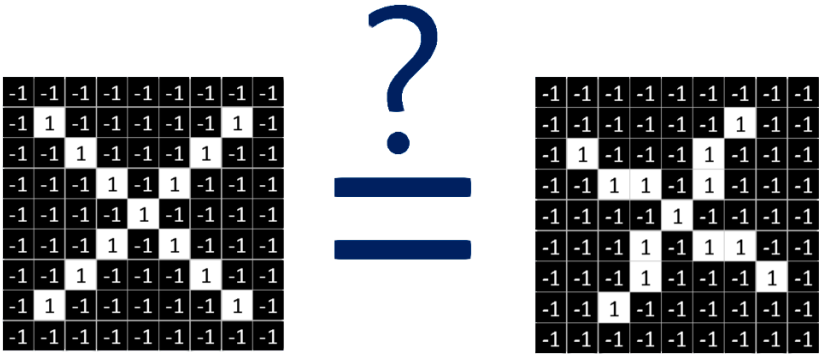


Figura 2.5: Problema de reconhecimento de caracteres manuscritos.

Neste exemplo fica claro que não basta apenas comparar a imagem por completo, mas sim algumas características particulares da imagem que representam a letra “X”, como demonstrado na Figura 2.6 [20].

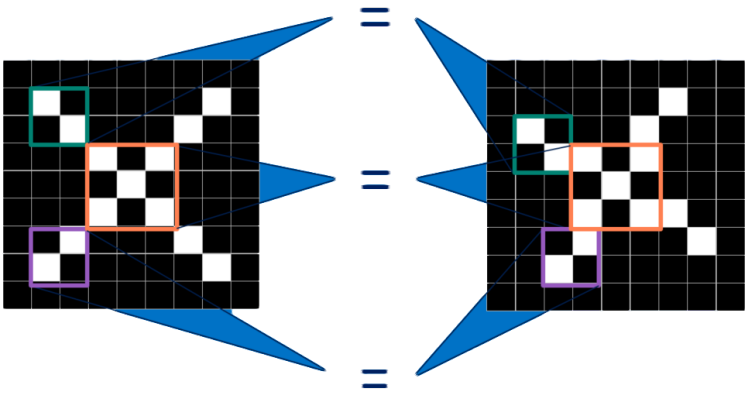


Figura 2.6: Extração de características de caracteres manuscritos.

Para concluir o raciocínio, a Figura 2.7 [20] mostra o resultado da convolução de 3 filtros gerados a partir das propostas de características apresentadas na Figura 2.6, com a imagem original.

As topologias de redes neurais abordadas neste trabalho utilizam,

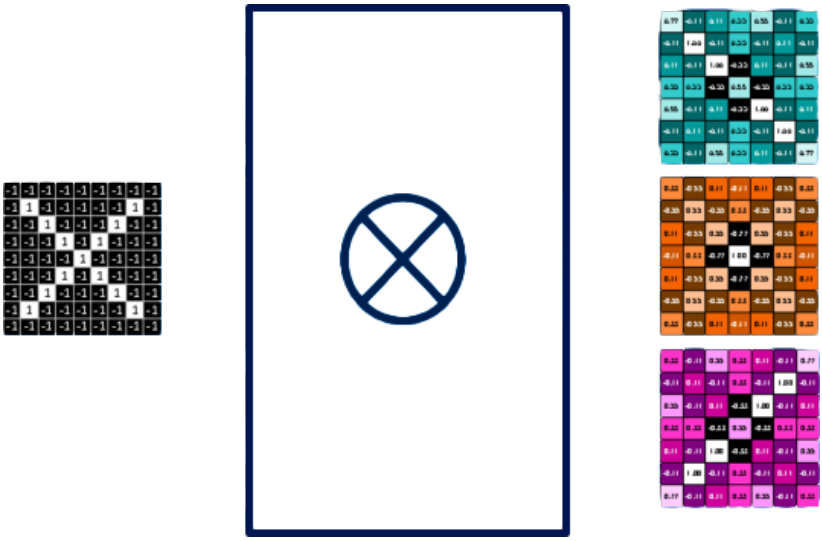


Figura 2.7: Exemplos de saídas de uma camada convolucional.

primariamente, camadas convolucionais.

2.3.1 Profundidade e Desempenho

Diversos experimentos têm demonstrado que apenas aumentar a profundidade de uma rede neural com camadas convolucionais não necessariamente melhora seu desempenho [21]. A partir de um certo ponto, a capacidade da rede fica muito alta e começa a haver *overfitting* (quando a rede aprende preditores que funcionam apenas para o conjunto de treinamento, mas não são suficientemente bons em outros conjuntos), ou até mesmo desempenho pior do que variantes com menos camadas. A Figura 2.8 [21] ilustra o desempenho de duas redes, com 20 e 56 camadas convolucionais, no conjunto de dados CIFAR-10 [22] e demonstra perfeitamente esse fenômeno, onde a rede de 56 camadas sempre apresenta desempenho pior que a de 20 camadas.

2.3.2 ResNet - *Residual Neural Network*

A utilização de topologias residuais em redes neurais foi proposta por um grupo de pesquisadores da Microsoft [21], e consiste na criação

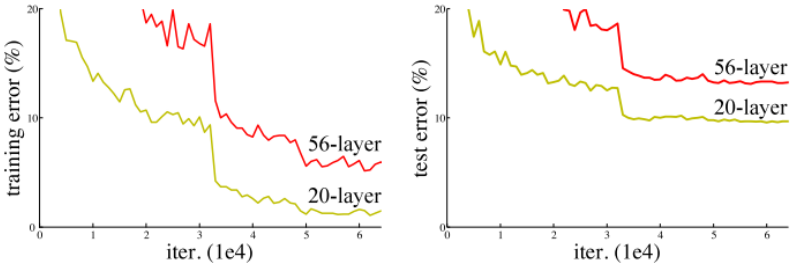


Figura 2.8: Desempenho de redes com diferente número de camadas.

de "atalhos" dentro da rede, de forma que os valores na saída de uma camada "pulam" uma ou mais camadas seguintes, sendo posteriormente combinados aos valores de saída das camadas que foram puladas. Esse conceito é demonstrado na Figura 2.9.

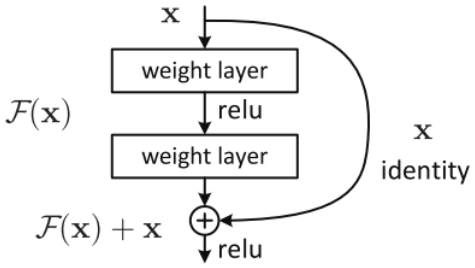
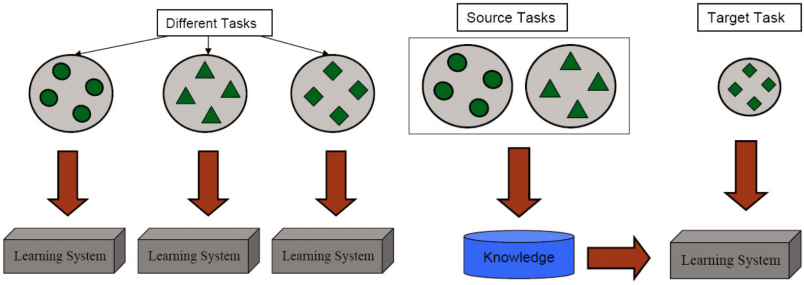


Figura 2.9: Conexões de atalho em redes neurais residuais.

A introdução das redes neurais residuais permitiu alcançar novos níveis de desempenho com algoritmos de inteligência artificial, reduzindo as limitações previamente existentes quanto à profundidade da rede. O artigo que realiza a sugestão inicial de utilização da topologia realiza testes com uma rede de 152 camadas, enquanto o estado da arte anterior envolvia redes de 16 camadas [23].

2.4 Aprendizado por Transferência e Ajuste Fino

É uma prática comum, principalmente em problemas de aprendizado de máquina e visão computacional onde a disponibilidade de ima-



(a) Aprendizado tradicional.

(b) Aprendizado por transferência.

Figura 2.10: Comparação de aprendizado por transferência com estratégias tradicionais.

gens no conjunto de treinamento é limitada, fazer o uso de aprendizado por transferência, ilustrado na Figura 2.10. Nesta modalidade, utiliza-se uma topologia comum, que já tenha sido treinada em algum outro conjunto de dados similar aos dados de interesse, e troca-se apenas as últimas camadas da rede, que realizam a classificação [24]. No caso específico das topologias ResNet, são re-treinadas apenas as novas camadas que foram inseridas.

Outra prática, adicional ao aprendizado por transferência, é o ajuste fino, que consiste no re-treinamento de alguns blocos-chave da rede de origem, tendo como ponto inicial os pesos pré-treinados em outro conjunto de dados.

As principais vantagens desses métodos são o tempo reduzido de treinamento da rede neural, já que apenas algumas de suas camadas precisam ser treinadas, e a possibilidade de sucesso com conjuntos de dados pequenos, que não seriam suficientes para treinar uma rede inicializada com pesos aleatórios. O presente trabalho utilizará ambas as técnicas de aprendizado por transferência e ajuste fino.

2.5 Estratégias para detecção de objeto

Na Seção 2.2.2, foram apresentados alguns passos necessários para obter sucesso em uma tarefa de detecção de objeto. Em algoritmos mais antigos, como o R-CNN [25], a execução desses passos requeria processar a imagem através de redes neurais duas vezes e, apesar de

trazerem bons resultados no que diz respeito à acurácia das detecções, é impossível executar detecções em tempo real devido à complexidade computacional requerida. Recentemente, novas estratégias mostraram resultados similares de acurácia, porém com complexidade computacional muito menor, através da execução de todo o processamento da imagem através de apenas uma rede neural.

2.5.1 YOLO - *You Only Look Once*

Em 2016 foi apresentada uma estratégia para possibilitar a detecção de objetos em tempo real [26], a qual foi denominada YOLO (*You Only Look Once*). Enquanto as estratégias anteriores utilizam uma rede neural projetada para classificação, na tarefa de detecção de objeto, a YOLO apresenta um método focado apenas em detecção. O procedimento de predição é relativamente simples de compreender com a ajuda de algumas imagens de exemplo, e consiste nos seguintes passos:

(*i*) A imagem é dividida em *grid* de 13 por 13 células, como demonstrado na Figura 2.11a.

(*ii*) Cada uma das 169 células é responsável pela predição de 5 caixas de seleção. Tais caixas podem ter qualquer tamanho ou formato, e correspondem à área onde a rede acredita que exista um objeto. Neste passo, também são obtidos *scores* de confiança para cada uma das caixas. Ainda não há nenhuma informação sobre a classe do objeto, apenas caixas que podem (ou não) conter um objeto qualquer. Essas caixas estão ilustradas na Figura 2.11b, onde caixas com a moldura mais espessa representam caixas com maior *score* de confiança.

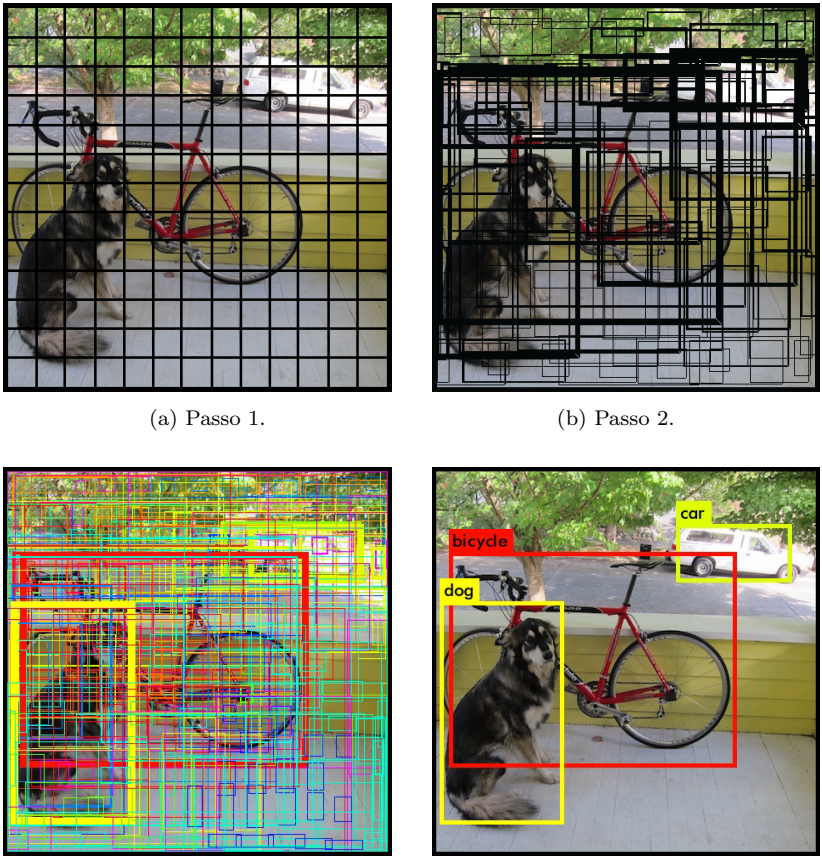
(*iii*) Para cada uma das 845 caixas de seleção, a rede faz a predição de uma classe. O *score* de confiança de que a caixa contenha um objeto é combinado com um novo *score* que representa a probabilidade de acerto no objeto classificado. O resultado deste passo pode ser visualizado na Figura 2.11c, onde caixas de diferentes cores representam diferentes classes de objetos.

(*iv*) Por fim, a grande maioria das caixas terá um *score* muito baixo, e são então eliminadas as caixas com *score* abaixo de um certo limiar (que costuma ser em torno de 30%). Após tal operação,

são obtidas as caixas com maior chance de conter um objeto, e que esse objeto tenha sido reconhecido pela rede. O resultado final pode ser visualizado na Figura 2.11d.

A principal vantagem é que a arquitetura do método YOLO permite realizar todos esses passos "olhando" para a imagem apenas uma vez (fato que nomeia o método), apresentando desempenho muito superior à de outros métodos.

O presente trabalho abordará duas variações da estratégia YOLO para detecção de objeto. A primeira é denominada YOLOv2, que foi introduzida no artigo original em 2016. Também será utilizada a YOLOv3, uma revisão apresentada em 2018 [27] que utiliza uma rede neural de 53 camadas, enquanto a versão anterior possui apenas 19 camadas.



(a) Passo 1.

(b) Passo 2.

(c) Passo 3.

(d) Passo 4.

Figura 2.11: Estratégia YOLO.

CAPÍTULO 3

Experimentos

Este capítulo é dedicado à descrição dos experimentos realizados no contexto do presente trabalho. Inicialmente, serão feitas breves descrições das ferramentas, do conjunto de dados e do pré-processamento utilizados, além das métricas usadas na avaliação dos resultados. Ao fim, serão apresentadas as estratégias utilizadas para aferir o desempenho das estratégias consideradas em diferentes condições ambientais, e para escolha da imagem que melhor representa um buraco.

3.1 Linguagens, *Frameworks* e Ambientes

De acordo com um levantamento feito a partir das vagas de emprego disponíveis no site indeed.com em 2017 [28], Python é a linguagem de programação mais utilizada na indústria para projetos de aprendizado de máquina. Este fato provavelmente está relacionado à grande disponibilidade de bibliotecas e *frameworks* (em sua maioria de código aberto), que por consequência gerou uma grande comunidade de troca de informação. Considerando que o autor já utilizou Python em projetos com outras finalidades (*web scraping*, aplicativos *desktop*, sistemas embarcados), utilizar Python como a linguagem-base para este projeto

foi uma escolha óbvia.

3.1.1 Keras

Keras [29] é uma biblioteca de código aberto, escrita em Python, que tem como principal objetivo ser uma interface para facilitar a implementação de algoritmos de aprendizado de máquina. A biblioteca trabalha apenas a nível de modelo, através de blocos construtivos de alto nível, mas não possui implementação das operações computacionais de baixo nível, como produtos vetoriais e convoluções. Para tais operações, faz-se uso de um *framework backend* especializado, como o TensorFlow [30], Theano [31] ou PlaidML [32].

Através da biblioteca Keras, é possível criar modelos de redes neurais a partir da simples especificação de suas camadas, assim como executar o treinamento, predições e avaliação. Também estão disponíveis módulos com métricas, *callbacks* e diversas outras ferramentas que facilitam o processo de otimização do modelo.

3.1.2 Plataformas computacionais

O processo de treinamento de uma rede neural profunda requer capacidade computacional considerável. Para as execuções computacionais deste trabalho, foi utilizado um computador Apple Mac Pro, com processador Intel Xeon W3690 e 32gb de memória RAM, executando o Keras 2.2.4 com *backend* TensorFlow 1.13.1 sobre o sistema operacional macOS 10.14.4. Também foi utilizado, em alguns casos, o *backend* PlaidML, que permite utilizar a GPU AMD Radeon R9 280 3gb disponível na máquina.

Para o treinamento de alguns modelos também foi utilizado o Google Colaboratory [33], plataforma online gratuita do Google que permite a utilização de uma máquina com GPU nVidia Tesla T4. Demais especificações técnicas da GPU utilizada, obtidos através do comando NVIDIA-SMI, podem ser visualizados na Figura 3.1.

3.1.3 Jupyter Notebook

Jupyter Notebook [34] é uma plataforma de código aberto que permite a criação de documentos que incluem a execução de código em

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.67       Driver Version: 410.79       CUDA Version: 10.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   40C    P8    15W /  70W |      0MiB / 15079MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

Figura 3.1: Especificações técnicas da GPU nVidia Tesla T4.

Python ou outras linguagens, permitindo a fácil leitura, execução e replicação dos mesmos [35].

Para este trabalho, o principal valor agregado a partir da utilização dessa ferramenta foi a capacidade de executar pequenos segmentos do código (células) de cada vez, com facilidade para visualizar os resultados (incluindo imagens) e sempre mantendo as variáveis ativas no ambiente de trabalho.

3.2 Conjunto de dados

O conjunto de dados utilizado neste trabalho [36] foi obtido por um grupo de pesquisadores da Stellenbosch University, para a realização de um trabalho com objetivo similar, porém utilizando apenas técnicas de processamento de imagem [6]. As imagens foram obtidas com uma câmera GoPro HERO3+ *Silver Edition*, fixada na parte interna de um veículo, apontando para a frente. As imagens têm resolução de 3680x2760 *pixels*, e foram capturadas a partir da configuração de um intervalo de meio segundo, com o carro em movimento a velocidade média de 40 km/h. As Figuras 3.2 e 3.3 mostram um exemplo negativo e um positivo, respectivamente, do conjunto de dados.

As imagens estão divididas em dois subconjuntos, chamados *Simplex* e *Complex*. O subconjunto *Simplex* contém imagens supostamente mais simples, onde a identificação poderia ser feita de maneira mais evidente, como ilustrado na Figura 3.4. O subconjunto *Complex* contém imagens mais elaboradas, com sombra de objetos externos ou incidência solar muito elevada, dificultando a visualização do pavimento, como ilustrado na Figura 3.5. Ambos os subconjuntos estavam divididos em partes de teste e treinamento, porém o conjunto de teste continha apenas exemplos positivos (com buracos). Apesar de o artigo original [6]

Figura 3.2: Exemplo negativo do conjunto de dados.

mencionar a captura de 48913 imagens, o subconjunto *Simplex* possui apenas 5387 imagens, enquanto o subconjunto *Complex* possui 8093 imagens - um total de 13480 imagens. Os autores alertam, em um arquivo *README* anexo ao conjunto, quanto à existência de imagens em comum entre os dois subconjuntos.

Para este trabalho, também foi criado um novo conjunto, desfazendo as divisões existentes, somando um total de 8644 imagens - 2459 exemplos positivos e 6185 negativos - após a eliminação das duplicatas. Além da classificação binária das imagens, também foi fornecida a anotação do conjunto em caixas de seleção, apontando exatamente a localização dos buracos, quando existentes. A anotação foi fornecida em formato TXT, contendo as coordenadas de um retângulo que envolve o defeito. A Figura 3.6 mostra o detalhe da área de interesse principal da Figura 3.3 com a adição da caixa de seleção. O autor não fornece detalhes específicos a respeito do procedimento de rotulação do conjunto de dados, nem sobre os critérios exatos utilizados na divisão entre subconjuntos *Simplex* e *Complex*, porém devido à natureza do

Figura 3.3: Exemplo positivo do conjunto de dados.

equipamento utilizado na aquisição das imagens, é possível inferir que estas foram realizadas em etapa posterior à aquisição, e não durante.

3.3 Organização e Pré-Processamento

Existem técnicas de organização e pré-processamento que podem ser utilizadas para melhorar o conjunto de dados antes de realizar o treinamento das redes. Tais técnicas são extremamente importantes para o sucesso de uma aplicação de aprendizado de máquina. A seguir, as técnicas consideradas no presente trabalho são brevemente descritas.

3.3.1 Redimensionamento das imagens

Como pode ser observado na Figura 3.3, as imagens contém muitos elementos desnecessários para o objetivo da detecção, como uma parte do painel do veículo na parte inferior, e o céu na parte superior. A área de interesse é a área onde pode haver ocorrência de buracos,

Figura 3.4: Exemplo positivo do subconjunto *Simplex*.

ou seja, apenas a faixa horizontal que mostra a estrada. Eliminar as demais informações é particularmente importante para a classificação binária, pois evita que outras características da foto não relacionadas ao problema principal, acabem causando tendências no modelo.

Nesse caso, já que existe a disponibilidade das anotações em torno dos buracos, foi possível analisar exatamente a área, dentre todas as imagens disponíveis, que contém buracos. A partir das anotações para os 2459 exemplos positivos do conjunto de dados, torna-se possível fazer uma boa generalização da área de interesse. Uma representação visual desta análise está demonstrada na Figura 3.7, que contém as caixas de todas as anotações do conjunto de dados, plotadas sobre o plano de fundo da Figura 3.3.

O resultado da análise mostrou que todas as caixas de seleção do conjunto de dados estão contidas entre os pontos $(0, 1194)$ e $(3676, 2018)$, sendo estas as coordenadas (x, y) que descrevem um retângulo coplanar à imagem. A Figura 3.8 mostra a mesma imagem da Figura 3.3 após o redimensionamento.

Figura 3.5: Exemplo positivo do subconjunto *Complex*.

As redes neurais estudadas tomam apenas imagens quadradas na entrada, o que causará uma deformação das imagens após redimensionamento. Isso pode até ser benéfico, dada a natureza "achatada" da visualização dos buracos a partir do ponto de vista da câmera (podemos visualizar na Figura 3.7 que o formato da maioria das caixas é um retângulo na horizontal), é possível que a deformação torne o buraco mais visível.

3.3.2 *Data Augmentation*

A prática de *Data Augmentation* consiste em aplicar transformações sobre um conjunto de dados para gerar novos exemplos de maneira sintética, que compartilhem preditores em comum com os exemplos originais (como ilustrado na Figura 3.9). É principalmente eficiente para problemas de classificação em conjuntos de dados pequenos, quando a quantidade de informação existente não é suficiente para atingir bons resultados com uma rede neural profunda [37].

Figura 3.6: Exemplo positivo do conjunto de dados com caixa de seleção.

As técnicas de *Data Augmentation* que serão utilizadas nesse trabalho são:

- Alteração de brilho - criar versões mais claras e mais escuras da imagem;
- Deslocamento - criar versões da imagem com o conteúdo deslocado em alguma direção;
- Espelhamento - criar versões da imagem espelhadas verticalmente ou horizontalmente.

A Figura 3.11 ilustra duas imagens obtidas utilizando a técnica de *Data Augmentation*. Ambas as duas foram geradas a partir da mesma imagem original, ilustrada na Figura 3.10.

3.3.3 Re-organização de subconjuntos

O conjunto foi divido em três subconjuntos: treinamento, validação e teste. O conjunto de treinamento é utilizado para de fato executar o treinamento da rede neural. O conjunto de validação é utilizado para avaliar o desempenho do modelo durante o treinamento, e evitar

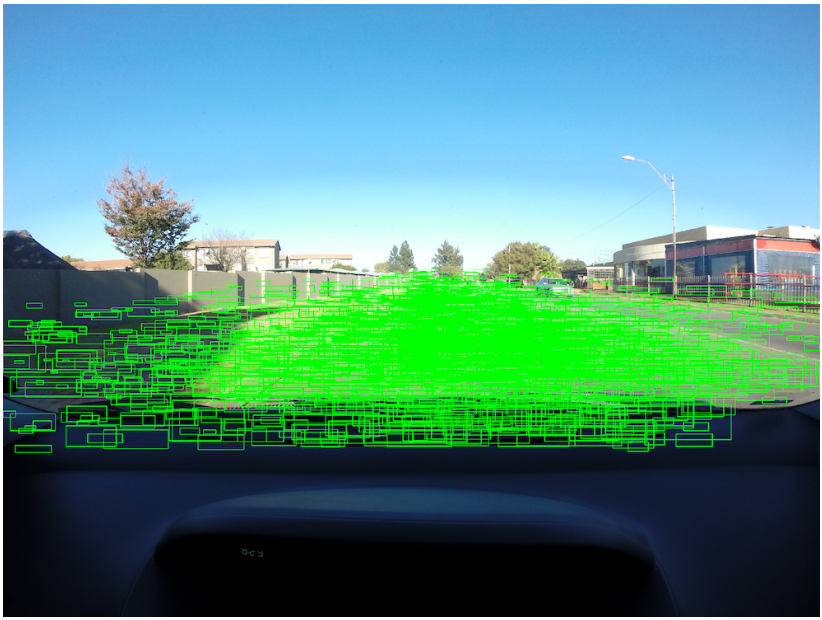


Figura 3.7: Demonstração de todas as anotações do conjunto de dados.

overfitting. O conjunto de teste contém imagens que não serão, de forma alguma, expostas para o modelo durante o treinamento, e somente serão utilizadas com o objetivo de fazer a avaliação final do modelo.

O conjunto de teste conterá 30% da totalidade das imagens. Os 70% restantes serão divididos entre treinamento e validação em uma razão de $4:1$, respectivamente.

Neste trabalho, devido ao fato de as fotos terem sido obtidas a partir de um carro em movimento, com um intervalo fixo, diversas fotos são muito similares. Foi necessário atentar a esse fato para evitar a presença de fotos muito semelhantes nos três subconjuntos, o que pode mascarar a existência de *overfitting*. Para tal, foi necessário atentar à natureza temporal do conjunto e fazer a divisão não de forma aleatória, mas sim escolhendo imagens que tenham sido obtidas em momentos diferentes (por consequência, em locais diferentes). Apesar de essa informação não estar explícita no conjunto de dados, foi possível obter a data e hora de captura original das imagens a partir dos dados complementares em formato EXIF, que são embutidos na imagem pela câmera que a

Figura 3.8: Exemplo positivo do conjunto de dados após redimensionamento.

Figura 3.9: Exemplos de *Data Augmentation*.

capturou.

A análise temporal revelou que as imagens foram obtidas entre os dias 11/04/2014 e 27/04/2014, entretando o maior esforço foi realizado especificamente em 26/04/2014, dia em que foram obtidas 7530 das 8644 imagens. A Figura 3.12 mostra um histograma da distribuição das imagens ao longo deste dia.

A Figura 3.13 mostra a distribuição horária para todo o conjunto de dados. É possível ver que o mesmo é formado por imagens obtidas em quase todos os horários, apesar de a grande maioria das imagens terem sido obtidas perto do meio-dia. Considerando que o pôr do sol em Stellenbosch, África do Sul (cidade onde foram obtidos os dados) nesses dias aconteceu por volta das 18h15m, é possível afirmar que o conjunto de dados conta com imagens obtidas à noite.

A Figura 3.14 mostra um exemplo de imagem obtida à noite.

Figura 3.10: Imagem antes do processo de *Data Augmentation*

(a) Aplicação de deslocamento vertical e horizontal.

(b) Aplicação de variação positiva de brilho e espelhamento horizontal.

Figura 3.11: Imagens resultantes do processo de *Data Augmentation*.

3.3.4 Avaliação de Condições ambientais

Para a avaliação em diferentes condições ambientais, será comparado o desempenho do algoritmo entre o subconjunto *Simplex*, que contém apenas imagens em condições ideais, e o conjunto de dados completo, que contém imagens onde a visualização do pavimento pode não ser trivial.

3.4 Métricas de avaliação

Tratando-se de um trabalho que envolve o treinamento de um algoritmo de inteligência artificial, o treinamento é dividido em épocas, sendo cada época um período de treinamento em que o algoritmo tenha

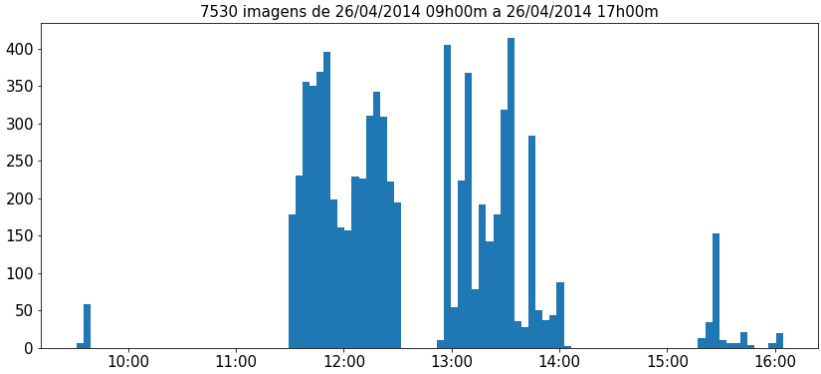


Figura 3.12: Distribuição das imagens obtidas no dia 26/04/2014.

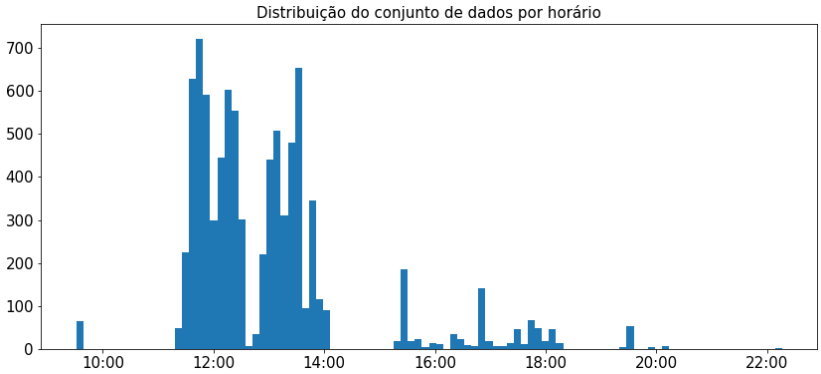


Figura 3.13: Distribuição do conjunto de dados por horário.

percorrido todo o conjunto de treinamento pelo menos uma vez. Nos casos que foi utilizado *Data Augmentation*, uma época poderá percorrer todo o conjunto de treinamento mais algumas variações sintéticas do mesmo.

Durante o treinamento, a principal métrica de avaliação é a função perda, que é a função principal que se deseja minimizar neste processo [38]. Esta deve ser avaliada tanto sobre o conjunto de treinamento quanto sobre o conjunto de validação. Caso a perda no conjunto de treinamento esteja muito menor que a perda no conjunto de validação, conclui-se que está havendo *overfitting* - o algoritmo aprendeu características sobre o conjunto de treinamento que não se aplicam ao conjunto

Figura 3.14: Exemplo de imagem obtida à noite.

de validação - ou seja, o algoritmo não terá bom desempenho quando aplicado em outros dados ainda não conhecidos.

Após um certo número de épocas, serão analizadas outras métricas, específicas para classificação ou detecção de objeto.

3.4.1 Classificação

No problema de classificação, serão avaliadas as seguintes métricas sobre os conjuntos de validação e teste:

- ***Precision*** - é a razão entre o número de predições positivas corretas, e o número total de predições positivas (corretas e incorretas, ou seja, $tp+fp$), como ilustrado na Equação 3.1. É um valor entre 0 e 1, e quanto maior, melhor [39].

$$Precision = \frac{tp}{tp + fp}. \tag{3.1}$$

- ***Recall*** - também conhecida como sensibilidade ou taxa de positivos corretos (TPR), é a razão entre o número de prediçoes positivas corretas, e a soma de predições positivas corretas e predições negativas incorretas (fn), como ilustrado na Equação 3.2. É um valor entre 0 e 1, e quanto maior, melhor.

$$Recall = \frac{tp}{tp + fn}. \tag{3.2}$$

- **F1** ***Score*** - é uma métrica calculada a partir da média harmônica de *precision* e *recall*, como ilustrado na Equação 3.3. É um valor

entre 0 e 1, e quanto maior, melhor.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}. \tag{3.3}$$

- **Acurácia balanceada** - por tratar-se de um conjunto de dados com grande desbalanceamento entre classes, é necessário utilizar uma métrica de acurácia que considere esse desbalanceamento. A acurácia balanceada (Equação 3.5) pode ser calculada como a média aritmética entre a taxa de positivos corretos (Equação 3.2), e a taxa de negativos corretos (Equação 3.4). É um valor entre 0 e 1, e quanto maior, melhor.

$$TNR = \frac{tn}{tn + fp}. \tag{3.4}$$

$$bAcc = \frac{TPR + TNR}{2} = \frac{recall + TNR}{2}. \tag{3.5}$$

3.4.2 Detecção de Objeto

Para melhor compreensão das métricas que serão utilizadas nesse problema, é necessário introduzir os conceitos que são utilizados para a definição de predições positivas e incorretas:

- **Razão *IoU (Intersection over Union)*** - trata-se da razão entre a intersecção e a união das caixas de seleção real e prevista. Tal razão é ilustrada na Figura 3.15 [40], e seu valor pode variar de 0 a 1. Na avaliação de um algoritmo de detecção de objeto, é necessário definir qual será o valor mínimo deste parâmetro para que uma caixa prevista que se sobrepõe parcialmente a uma caixa real possa ser considerada uma detecção positiva verdadeira.

- **Filtro *NMS (Non-Maximum Suppression)*** - é uma técnica de pós-processamento comumente utilizada em algoritmos de detecção de objeto [41], com objetivo de eliminar detecções duplicadas. Dadas duas ou mais caixas previstas que se interceptam, será mantida apenas a caixa com maior coeficiente de confiança. Existe a possibilidade de definir uma razão IoU mínima para que seja considerada a intersecção entre duas caixas previstas.

Serão consideradas predições positivas corretas (tp) aquelas que, após a aplicação de um Filtro NMS, possuírem razão IoU mínima com alguma caixa de seleção real. Da mesma forma, as predições positivas incorretas (fp) são as que não atingirem a razão IoU mínima com uma caixa de seleção real. As predições negativas incorretas (fn) serão representadas pelo número de caixas de seleção reais que não apresentaram sobreposição com nenhuma predição. As seguintes métricas serão então avaliadas sobre os conjuntos de validação e teste, com variação dos parâmetros NMS e IoU mínimos:

- **AP (*Average Precision*)** - para obter essa métrica, primeiramente são obtidos os valores de *precision* e *recall* ao longo de todo o conjunto de testes. As séries (que conterão o mesmo número de pontos) devem ser ordenadas pelo coeficiente de confiança da predição e plotadas no mesmo gráfico, com *recall* no eixo horizontal e *precision* no eixo vertical. O gráfico resultante contém a informação do desempenho do modelo para diversos valores de *threshold* de detecção. O valor AP corresponde à área sob a linha *precision-recall*, ou seja, a integral do gráfico obtido.
- ***Precision***.
- ***Recall***.
- **F1 *Score***.

3.5 Escolha da melhor imagem

Além dos algoritmos de aprendizado de máquina que esse trabalho envolve, também será necessário, ao fim, implementar um algoritmo para realizar a escolha da imagem que melhor representa um buraco na pista, dentre um conjunto de imagens obtidos em torno do instante do acionamento manual do sistema. Esse algoritmo precisará ser diferente para os processos de classificação e de detecção de objeto.

3.5.1 Classificação

Para o caso de classificação, será simplesmente escolhida a imagem que o algoritmo tiver classificado com maior *score*, ou seja, com maior chance de conter um buraco.

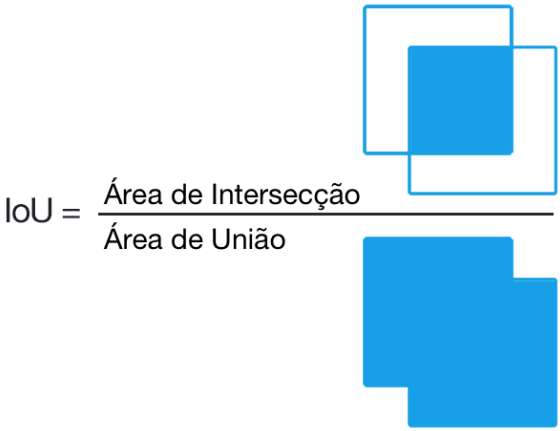


Figura 3.15: Demonstração de Intersecção sobre União.

3.5.2 Detecção de Objeto

O algoritmo implementado após a detecção de objeto seguirá os seguintes passos:

(*i*) Serão eliminadas as imagens para as quais não tiver sido encontrado nenhum objeto;

(*ii*) Dentre as imagens restantes, para as que contiverem mais de um objeto encontrado, será selecionado o objeto com maior *score*;

(*iii*) A partir do conjunto de objetos restantes, será escolhida a imagem que contém um objeto mais próximo do centro. Para tal, será calculada a distância entre o centro da caixa de detecção e o centro da imagem.

3.6 Testes a campo

Com a finalidade de avaliar, de maneira empírica, a aplicação dos algoritmos estudados nesse trabalho em situações reais, serão obtidas algumas imagens externas ao conjunto de dados utilizado no treinamento. Para tal, será utilizado um *smartphone* Apple iPhone 6S, fixado em um veículo de forma que a visão frontal a partir de dentro

do veículo seja semelhante àquela do conjunto de dados apresentado na Seção 3.2. A Figura 3.16 mostra em detalhe o suporte utilizado, com o *smartphone* em posição. Por tratar-se de uma avaliação empírica, não serão apresentados dados numéricos sobre o desempenho dos algoritmos sobre este conjunto de dados.

Figura 3.16: Detalhe do *smartphone* e suporte utilizado.

Para a captura das imagens, o *smartphone* conta com um recurso de câmera denominado *Live Photo* que captura, além da imagem em si, um vídeo de 3 segundos com os momentos anteriores e posteriores. Tal recurso atende perfeitamente o cenário estudado nesse trabalho.

Com a finalidade de facilitar a tarefa de captura das imagens, foi utilizado um disparador remoto *Bluetooth*, ilustrado na Figura 3.17.

Figura 3.17: Detalhe do disparador remoto utilizado.

CAPÍTULO 4

Resultados

Este capítulo é dedicado à apresentação dos resultados dos experimentos realizados no contexto do presente trabalho.

4.1 Classificação

As Tabelas 4.1 e 4.2 descrevem o desempenho de classificação atingida para o conjunto completo e para o subconjunto *Simplex*, respectivamente. As métricas apresentadas em tais tabelas são a acurácia balanceada (**bAcc**), *precision* (**Prc**), *recall* (**Rec**) e *F1 Score* (**F1**), as quais são descritas na Seção 3.4.1. Uma análise mais detalhada contendo a Curva ROC será feita apenas para o algoritmo que demonstrou melhor desempenho de acordo com as métricas da tabela. A primeira coluna contém um identificador da estratégia utilizada. Os identificadores que terminam com **-C** representam redes que possuem a adição das seguintes camadas ao final, em ordem:

- Convolucional - 64 unidades com *grid* 3x3,
- Batch Normalization,
- Ativação ReLU,

- Dropout 0.5,
- Convolucional - 1 unidade com *grid* 3x3 e função de ativação *sigmoid*.

As redes representadas por identificadores que terminam com "-D", por sua vez, possuem a adição das seguintes camadas ao final, em ordem:

- Densa - 4096 unidades com ativação ReLU,
- Densa - 4096 unidades com ativação ReLU,
- Densa - 1 unidade com função de ativação *sigmoid*.

Os primeiros três dígitos do identificador correspondem à topologia utilizada, de acordo com a descrição abaixo:

- **R50** - Topologia ResNet50;
- **V16** - Topologia VGG16 [23].

Adicionalmente, identificadores terminados na letra **A** correspondem a modelos que utilizaram *Data Augmentation* no processo de treinamento, conforme descrito na Seção 3.3.2.

Alg.	**Validação**				**Teste**			
	bAcc	***Prc***	***Rec***	***F1***	**bAcc**	***Prc***	***Rec***	***F1***
R50-C	0.883	0.858	0.859	0.858	0.538	0.455	0.876	0.599
R50-CA	0.888	0.987	0.613	0.756	0.537	0.290	0.726	0.414

Tabela 4.1: Desempenho de classificação - Conjunto completo

Os resultados apresentados nas Tabelas 4.1 e 4.2 mostram que não foi possível obter bons resultados a partir da utilização de todo o conjunto de dados. O único cenário onde foi obtida bom desempenho de classificação foi com a utilização do subconjunto de dados *Simplex*, que contém imagens mais simples, conforme detalhado na Seção 3.2. Este resultado pode ser atribuído ao fato de que as imagens do subconjunto *Simplex* são, em geral, bem iluminadas, e contém menos sombras de

Alg.	Validação				Teste			
	bAcc	***Prc***	***Rec***	***F1***	**bAcc**	***Prc***	***Rec***	***F1***
R50-C	0.862	0.826	0.917	0.869	0.737	0.491	0.959	0.650
R50-CA	0.921	0.880	0.903	0.891	0.778	0.959	0.580	0.723
V16-D	0.885	0.858	0.923	0.890	0.718	0.477	0.936	0.632
V16-DA	0.872	0.967	0.757	0.850	0.731	0.713	0.538	0.613

Tabela 4.2: Desempenho de classificação - Subconjunto *Simplex*

objetos externos sobre o pavimento. Fica assim evidenciado o impacto das condições ambientais no desempenho do sistema.

O algoritmo R50-CA aplicado ao subconjunto *Simplex* obteve a melhor desempenho no conjunto de teste. É importante visualizar a discrepância entre os valores de *precision* e *recall*, onde o alto valor do primeiro parâmetro indica pequena quantidade de predições positivas incorretas, porém o baixo valor do segundo parâmetro indica grande quantidade de predições negativas incorretas. Isso significa que as predições positivas do algoritmo são bastante confiáveis, mas muitas ocorrências positivas podem estar sendo classificadas como negativas. Tal comportamento é evidenciado na Curva ROC, apresentada na Figura 4.1, que mostra o desempenho do sistema para diferentes níveis de limiar de classificação. É possível visualizar que o limiar de classificação, inicialmente configurado para 0.5 (em uma escala de 0 a 1) e representado pelo círculo verde na Figura 4.1 está muito para o lado esquerdo da curva. Caso houvesse interesse em obter um sistema balanceado em termos de *precision* e *recall*, a Curva ROC é uma excelente ferramenta para a escolha do limiar de classificação. Nesse caso particular, o limiar de classificação que deixaria o sistema balanceado é o valor de 0.32, representado pelo círculo vermelho na Figura 4.1.

As Tabelas 4.3a e 4.3b apresentam a matriz de confusão para o algoritmo R50-CA com limiar de classificação de 0.5 e 0.31 (escolha balanceada), respectivamente. Com a escolha do novo limiar, atinge-se acurácia balanceada de 0.858, e *F1-Score* de 0.856, valores superiores aos obtidos com o limiar padrão de 0.5. É importante mencionar que a escolha do limiar de classificação para balanceamento entre *precision* e *recall* é ideal apenas para aplicações em que a sensibilidade e a

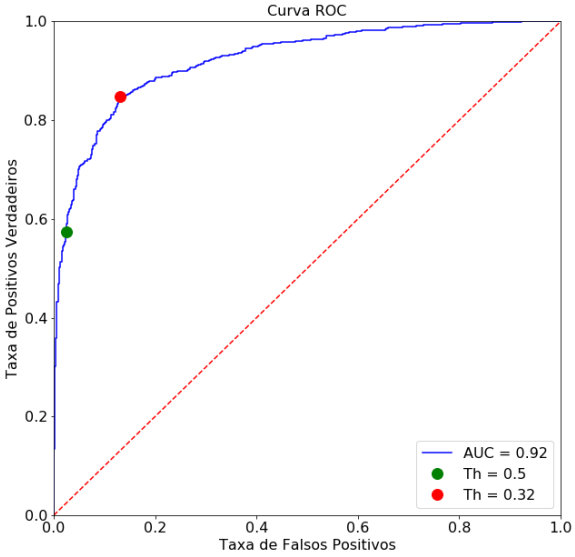


Figura 4.1: Curva ROC para o algoritmo de classificação R50-CA.

especificidade do sistema são igualmente importantes [42].

4.2 Detecção de Objeto

A Tabela 4.4 descreve o desempenho de detecção de objeto atingida para o conjunto completo. Adicionalmente às métricas de classificação, é apresentada a *Average Precision* (***AP***), que foi obtida de acordo com o procedimento apresentado na Seção 3.4.2. Uma análise mais detalhada contendo a Curva *Precision-Recall* que é utilizada no cálculo da *Average Precision* é apresentada para o algoritmo que demonstrou melhor desempenho. A primeira coluna contém o identificador do algoritmo dentre as topologias utilizadas, apresentadas na Seção 2.5.1.

Para uma avaliação empírica, a Figura 4.2 ilustra dois exemplos do conjunto de testes, onde as caixas azuis representam a patologia real, e as caixas vermelhas representam a detecção realizada pela rede.

		Predição		
		Negativo	Positivo	Total
Valor Real	Negativo	635	16	651
	Positivo	273	378	651
	Total	908	394	0.778

(a) Limiar de classificação = 0.5.

		Predição		
		Negativo	Positivo	Total
Valor Real	Negativo	566	85	651
	Positivo	100	551	651
	Total	666	636	0.858

(b) Limiar de classificação = 0.31.

Tabela 4.3: Matriz de confusão para o algoritmo de classificação R50-CA.

Alg.	**Teste**			
	Prc	***Rec***	***F1***	***AP***
YOLOv2	0.757	0.563	0.646	0.518
YOLOv3	0.522	0.947	0.674	0.722

Tabela 4.4: Desempenho de detecção de objeto

4.3 Escolha da melhor imagem

Os resultados para a escolha da melhor imagem serão dispostos de maneira empírica. A Figura 4.3 mostra 3 imagens sequenciais escolhidas dentro da divisão de testes do subconjunto *Simplex*. As Figuras 4.4 e 4.5 demonstram a imagem escolhida como melhor representante do defeito, pelos algoritmos de classificação e detecção de objeto, respectivamente.

A escolha da melhor imagem utilizando os algoritmos de detecção de objeto apresenta resultados superiores aos obtidos utilizando os algoritmos de classificação. Este fato pode ser explicado pela falta de informação disponível sobre a localização da irregularidade, logo o algoritmo analisa apenas a probabilidade de existência de uma irregula-

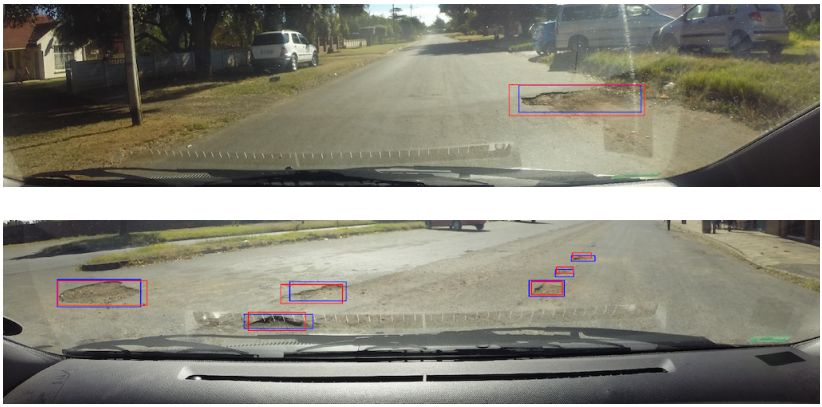


Figura 4.2: Exemplos do resultado de detecção de objeto.

Figura 4.3: Sequência de imagens do conjunto de dados.

Figura 4.4: Imagem escolhida pelo algoritmo de classificação.

Figura 4.5: Imagem escolhida pelo algoritmo de detecção de objeto.

ridade, mas não leva em consideração seu enquadramento na imagem.

Adicionalmente, foi realizado um teste com imagens externas ao conjunto de dados, obtidas na cidade de Florianópolis, através do método descrito na Seção 3.6. A Figura 4.6 mostra 5 imagens sequenciais após terem sido processadas pelo algoritmo de detecção de objetos, onde a terceira imagem foi escolhida como melhor representativa do buraco observado.

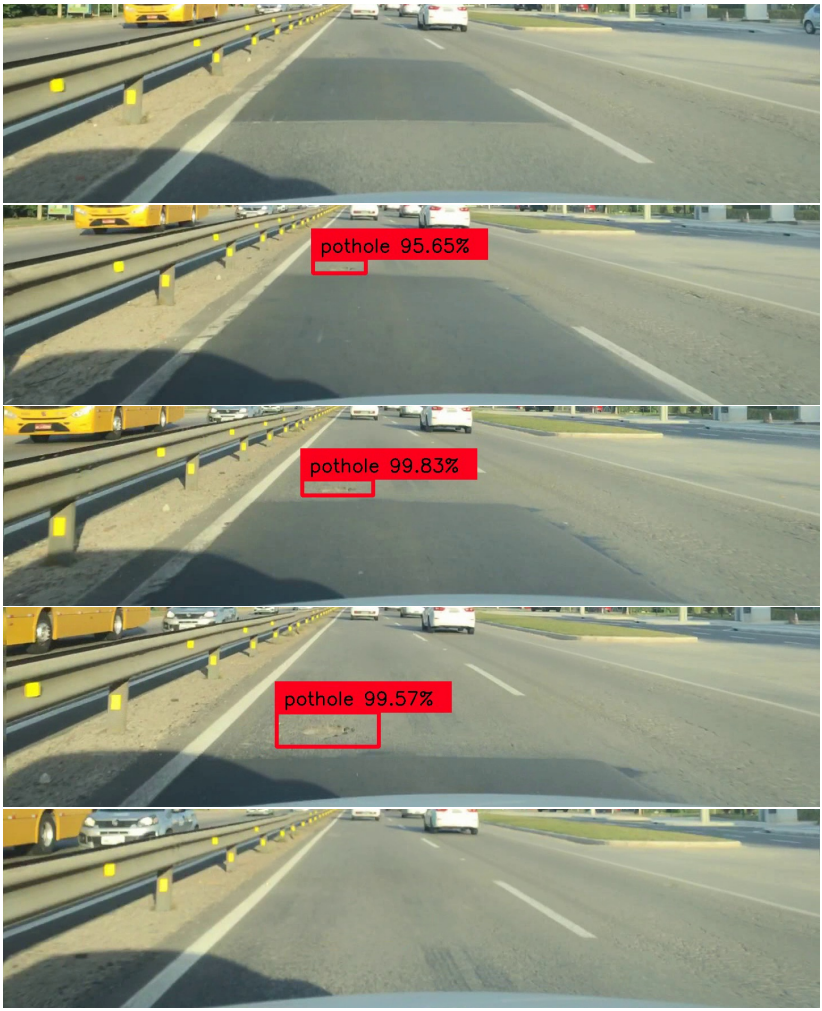


Figura 4.6: Sequência de imagens externa ao conjunto de dados.

CAPÍTULO 5

Análise de Viabilidade de Equipamento

Uma informação essencial na definição quanto à viabilidade do projeto, diz respeito aos custos e níveis de equipamento necessários para embutir esta tecnologia em um sistema embarcado. Este capítulo abordará, de forma breve, os requisitos de tal sistema quanto à câmera e capacidade computacional, fazendo paralelos com *smartphones* disponíveis no mercado e também com o computador *Raspberry Pi*, o qual possui baixo custo e dimensões reduzidas.

5.1 Câmeras Digitais

Para que o algoritmo possa funcionar corretamente, as imagens que o alimentam devem representar cenas estáticas, sem borrões. A existência de borrões pode acontecer, mesmo que a câmera esteja perfeitamente fixa, já que o carro estará em movimento relativo à estrada. Em fotografia digital, há três parâmetros básicos que podem ser configurados e definem o aspecto da foto:

- Velocidade do obturador - corresponde ao tempo pelo qual o sensor é exposto à luz para realizar a captura da imagem [43].

Quanto maior o tempo de exposição, mais clara fica a imagem, mas também há mais chance de ficar borrada;

- Abertura - também conhecido como *F-Number* corresponde à razão entre o diâmetro de entrada e saída da lente. Um valor numericamente menor (como F1.8) significa que mais luz está chegando ao sensor e que a profundidade de campo é reduzida. Um valor numericamente maior (como F10) significa que menos luz está chegando ao sensor, e que a profundidade de campo é mais ampla, ou seja, é possível capturar com nitidez elementos próximos e distantes;

- ISO - também conhecido como sensibilidade, corresponde ao ganho do sensor. Valores maiores permitem fotos mais claras, porém com mais ruído.

Além disso, a distância focal também é um parâmetro muito importante, porém costuma ser fixo nas câmeras mais populares.

Um bom ponto de partida para analisar esses parâmetros é o próprio conjunto de dados. Assim como a análise feita para data e hora das imagens na Seção 3.3.3, todos os parâmetros podem ser extraídos dos dados complementares em formato EXIF. No caso da câmera utilizada para obtenção do conjunto de dados, dois parâmetros estão fixos: a distância focal - fixa em 2,77mm (15,235mm em medida equivalente a um sensor de 35mm) - e a abertura - fixa em F2.8. A grande maioria das imagens (8571 - 99,15%) utilizou sensibilidade ISO 100, com as demais variando até ISO 400. O parâmetro que mais variou foi a velocidade do obturador, cuja distribuição pode ser visualizada na Figura 5.1. A velocidade do obturador costuma ser representada por uma fração cujo numerador sempre é 1.

O parâmetro que deve ser suficientemente pequeno para garantir que a foto não conterá borrões é a velocidade do obturador. A seguinte fórmula pode ser utilizada para obter o valor máximo para esse parâmetro [44]:

$$N \leq \frac{D * C}{F * V}, \tag{5.1}$$

onde N representa a velocidade do obturador (em segundos), D representa a distância até o objeto, C representa o círculo de confusão

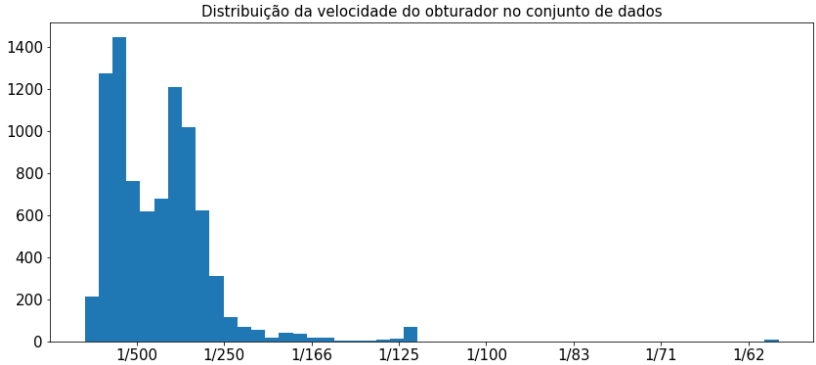


Figura 5.1: Distribuição da velocidade do obturador nas imagens do conjunto de dados.

(grandeza que representa o tamanho mínimo de um movimento para que seja considerado um borrão - será utilizado 0.1mm nesse trabalho [45]), F representa a distância focal da lente (em mm, equivalente a um sensor de 35mm), e V representa a velocidade relativa entre o objeto e a câmera.

Aplicando os parâmetros da câmera utilizada na obtenção do conjunto de dados à Equação 5.1, considerando velocidade relativa de 40 km/h e distância de 10m entre a câmera e o pavimento, encontramos que o valor máximo para a velocidade do obturador deve ser de 0,0059s. A grande maioria das imagens (99,04%) foi obtida com tempo inferior ao máximo, ou seja, as imagens podem ser consideradas estáticas.

Trazendo este cálculo para a realidade de um *smartphone* de boa qualidade, como o Apple iPhone 6S, o valor calculado máximo para a velocidade do obturador é de 0,003s. Considerando que tal equipamento possui abertura máxima de F2.2 (superior ao equipamento utilizado na obtenção do conjunto de dados) e obtém imagens com nível de ruído aceitável com sensibilidade ISO 800, é seguro afirmar que um *smartphone* pode ser utilizado para obtenção das imagens no contexto estudado nesse trabalho. A Figura 5.2 ilustra uma imagem obtida com tal *smartphone*, utilizando velocidade do obturador de 0,0027s.

Figura 5.2: Imagem obtida a partir de *smartphone* com velocidade do obturador de 1/371s.

5.2 Tempo de execução do algoritmo

Igualmente importante à obtenção das imagens é a execução do algoritmo de detecção. Com objetivo de obter uma estimativa do tempo de execução dos algoritmos estudados em uma plataforma embarcada, o processo de inferência dos mesmos sobre imagens do conjunto de dados foi executado em um computador Raspberry Pi 3. Tal dispositivo possui processador ARM *Quad-Core* com poder computacional similar a um *smartphone* de baixo custo. Os resultados de tempo de execução por imagem, para cada um dos algoritmos abordados, pode ser visualizado na Tabela 5.1.

Como demonstrado na Tabela 5.1, não foi possível executar os algoritmos de classificação no dispositivo em questão. Tal fato ocorreu devido ao limite na quantidade de memória RAM disponível. Os algoritmos de detecção de objeto, apesar de terem sido executados com sucesso, o tempo de execução dos mesmos inviabiliza soluções imple-

Algoritmo	**Tempo de Execução**
ResNet50	Não executou
VGG16	Não executou
YOLOv2	27.5s
YOLOv3	22.2s

Tabela 5.1: Tempo de execução dos algoritmos em plataforma embarcada

mentadas em tempo real.

CAPÍTULO 6

Conclusão

Os resultados apresentados no Capítulo 4 sustentam uma hipótese de que é possível utilizar visão computacional e aprendizado de máquina em sistemas de auxílio ao levantamento de defeitos em pavimento rodoviário. De acordo com os objetivos específicos do presente trabalho, apresentados na Seção 1.3.1:

- Foi encontrado um conjunto de dados que contém imagens de buracos no pavimento rodoviário. Este conjunto foi utilizado no treinamento de algoritmos de aprendizado de máquina. Apesar do treinamento ter sido efetivo na maioria dos casos, também foi visualizado na Seção 4.3 que o fato das imagens terem sido obtidas em outro país, cujo pavimento rodoviário tem aspecto diferente do brasileiro, pode impor uma restrição na utilização somente dessas imagens para o treinamento de um sistema comercial;

- Foram testados algoritmos populares de classificação e detecção de objeto. Os resultados a partir da detecção de objeto foram consideravelmente melhores, fato que pode ser atribuído à informação que é fornecida à rede no momento do treinamento. Enquanto no problema de classificação binária a rede treina apenas com a in-

formação da existência (ou não) de irregularidade na imagem, no problema de detecção de objeto, existe a informação do local e tamanho da irregularidade. Obviamente, o processo de obtenção de um conjunto de dados mais completo que seria necessário para a operação comercial do sistema é muito mais complexa para detecção de objetos, já que é necessário desenhar manualmente as caixas identificando cada irregularidade;

- A utilização de *Transfer Learning* certamente é um dos fatores que viabiliza o estudo de um problema como o abordado nesse trabalho, já que diminui consideravelmente a complexidade computacional do treinamento;

- Foi identificado que as condições climáticas e ambientais em que a foto é capturada influenciam fortemente os resultados, sendo que em alguns casos é impossível visualizar o pavimento corretamente a partir da imagem obtida. Esse fator consiste uma limitação operacional prática imposta pelo equipamento e pelas condições em que o mesmo é utilizado, não sendo diretamente relacionada aos algoritmos utilizados;

- Uma limitação relevante encontrada, como demonstrado na Seção 5.2, é o tempo de execução dos algoritmos estudados, que inviabiliza a execução dos mesmos em tempo real em um sistema embarcado. Uma possível solução seria o armazenamento dos dados para serem processados posteriormente com utilização de infraestrutura em nuvem, a utilização de redes neurais mais simples como a MobileNet [46], ou mesmo o uso de técnicas de compressão de redes [47].

Trabalhos futuros

Por fim, durante a realização desse trabalho, foram identificadas as seguintes linhas de estudo hipotéticas, que poderiam ser utilizadas em trabalhos futuros para obter resultados melhores que os demonstrados no presente estudo:

- Aplicar técnicas mais avançadas no pré-processamento das imagens, não apenas recortando uma área de interesse definida por

amostragem mas sim detectando a área ocupada pelo pavimento. Tal técnica poderia reduzir a incidência de falsos positivos na área exterior ao pavimento, e também eliminar a influência de outros atributos irrelevantes da imagem no processo de treinamento;

- Utilização de imagens tri-dimensionais para não apenas realizar a detecção das irregularidades, mas também estimar a distância entre o veículo e a irregularidade, e possivelmente algum parâmetro quantitativo sobre a dimensão da mesma;
- Obter um conjunto de dados mais extenso, e fazer a anotação do mesmo incluindo outros tipos de patologias rodoviárias.

Referências bibliográficas

[1] ASSOCIAÇÃO BRASILEIRA DE CONCESSIONÁRIAS DE RODOVIAS, "Relatório anual." `http://www.abcr.org.br/RelatoriosAnuais/RelatorioAnual2017.pdf`, 2017. Acesso em: 16 jun 2019.

[2] AGÊNCIA NACIONAL DE TRANSPORTES TERRESTRES, "Concessões rodoviárias." `http://www.antt.gov.br/rodovias/Concessoes_Rodoviarias/index.html`. Acesso em: 16 jun 2019.

[3] L. V. de Carvalho Almeida, M. G. Pignatti, and M. M. Espinosa, "Principais fatores associados à ocorrência de acidentes de trânsito na BR 163, Mato Grosso, Brasil," *Cadernos de Saúde Pública*, vol. 25, no. 2, pp. 303–312, 2009.

[4] DEPARTAMENTO NACIONAL DE INFRAESTRUTURA DE TRANSPORTES, "Norma 008/2003 - PRO - Levantamento visual contínuo para avaliação da superfície de pavimentos flexíveis e semi-rígidos," 2003.

[5] Road Labs, "Road Labs - Rodovias Conectadas e Inteligentes." `https://roadlabs.com.br/`, 2019. Acesso em: 16 jun 2019.

[6] S. Nienaber, M. T. Booysen, and R. Kroon, "Detecting potholes using simple image processing techniques and real-world footage,"

07 2015. Disponível em: https://scholar.sun.ac.za/handle/10019.1/97191. Acesso em: 16 jun 2019.

[7] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, "Road damage detection and classification using deep neural networks with smartphone images: Road damage detection and classification," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, 06 2018.

[8] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using android smartphones with accelerometers," pp. 1 – 6, 06 2011.

[9] P. Fornari, "Avaliacao de sensores e algoritmos para identificacao de irregularidades em rodovias," 2018.

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, vol. 82. 01 2003.

[11] D. H. D. H. Ballard and C. M. Brown, *Computer vision*. Englewood Cliffs, N.J. : Prentice-Hall, 1982.

[12] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210–229, July 1959.

[13] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. 01 2013.

[14] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, pp. 1–21, 01 2019.

[15] P. Sharma, "A practical guide to object detection using the popular yolo framework." https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python/. Acesso em: 16 jun 2019., 2018.

[16] F. I. B. Rodrigues, R. R. M. Carvalho, G. P. Gonçalves, and K. B. P. Barbosa, "Estudo de algoritmo mlp como aproximador de função," 2016.

[17] F. van Veen and S. Leijnen, "A mostly complete chart of neural networks." `http://www.asimovinstitute.org/wp-content/uploads/2019/04/NeuralNetworkZoo20042019.png`. Acesso em: 16 jun 2019., 2019.

[18] Data Science Academy, "Deep learning book." `http://www.deeplearningbook.com.br/`. Acesso em: 16 jun 2019., 2019.

[19] H. Habibi Aghdam and E. Jahani Heravi, *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. 01 2017.

[20] B. Rohrer, "How do convolutional neural networks work?." `https://brohrer.github.io/how_convolutional_neural_networks_work.html`. Acesso em: 16 jun 2019., 2019.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," vol. 7, 12 2015.

[22] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[24] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, Oct 2010.

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," pp. 779–788, 06 2016.

[27] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 04 2018.

[28] J.-F. Puget, "The most popular language for machine learning is ." https://medium.com/inside-machine-learning/the-most-popular-language-for-machine-learning-is-46e2084e851b. Acesso em: 16 jun 2019., 2017.

[29] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[30] M. Abadi, A. Agarwal, P. Barham, and E. B. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems." http://tensorflow.org/, 2015. Software available from tensorflow.org.

[31] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. http://arxiv.org/abs/1605.02688.

[32] Intel, "Plaidml." https://github.com/plaidml/plaidml.

[33] Google, "Google colaboratory." https://colab.research.google.com/.

[34] P. Jupyter, "Project jupyter." https://jupyter.org/.

[35] T. Kluyver, B. Ragan-Kelley, F. Perez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and J. development team [Unknown, "Jupyter notebooks a publishing format for reproducible computational workflows," 01 2016.

[36] S. Nienaber, M. T. Booysen, and R. Kroon, "Dataset of images used for pothole detection," 10 2015.

[37] A. Mikolajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," pp. 117–122, 05 2018.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, pp. 80–81. MIT Press, 2016. http://www.deeplearningbook.org. Acesso em: 16 jun 2019.

[39] D. Powers, "Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation," *Mach. Learn. Technol.*, vol. 2, 01 2008.

[40] A. Rosebrock, "Intersection over union (iou) for object detection." https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. Acesso em: 16 jun 2019., 11 2016.

[41] R. Rothe, M. Guillaumin, and L. Van Gool, "Non-maximum suppression for object detection by passing messages between windows," vol. 9003, 04 2015.

[42] R. Froud and G. Abel, "Using roc curves to choose minimally important change thresholds when sensitivity and specificity are valued equally: The forgotten lesson of pythagoras. theoretical considerations and an example application of change in health status," *PloS one*, vol. 9, p. e114468, 12 2014.

[43] R. Jacobson, S. Ray, G. Attridge, and N. Axford, *Manual of Photography*. Taylor & Francis, 2000.

[44] P. Facey, "How to freeze subject movement." http://www.brisk.org.uk/photog/subblur.html. Acesso em: 16 jun 2019.

[45] B. Atkins, "Digital depth of field." http://bobatkins.com/photography/technical/digitaldof.html. Acesso em: 16 jun 2019.

[46] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 04 2017.

[47] S. Han, H. Mao, and W. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 10 2016.