UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS SOBRAL
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

CARLOS DAVID BRAGA BORGES

APRENDIZADO DE ATRAVESSABILIDADE EM IMAGENS AÉREAS USANDO REDES TOTALMENTE CONVOLUCIONAIS

SOBRAL

2021

CARLOS DAVID BRAGA BORGES

APRENDIZADO DE ATRAVESSABILIDADE EM IMAGENS AÉREAS USANDO REDES TOTALMENTE CONVOLUCIONAIS

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da Universidade Federal do Ceará, campus Sobral, como requisito para obtenção do título de Mestre em Engenharia Elétrica e de Computação.

Orientador: Prof. Dr. Jarbas Joaci de Mesquita Sá Junior

SOBRAL

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B731a Borges, Carlos David Braga.
Aprendizado de atravessabilidade em imagens aéreas usando redes totalmente convolucionais / Carlos David Braga Borges. – 2021.
83 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Sobral, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Sobral, 2021.
Orientação: Prof. Dr. Jarbas Joaci de Mesquita Sá Junior.

1. Navegação Robótica. 2. Atravessabilidade. 3. Planejamento. 4. Imagens Aéreas. 5. Redes Totalmente Convolucionais. I. Título.

CDD 621.3

Carlos David Braga Borges

Aprendizado de atravessabilidade em imagens aéreas usando redes totalmente convolucionais

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da Universidade Federal do Ceará, campus Sobral, como requisito para obtenção do título de Mestre em Engenharia Elétrica e de Computação.

Trabalho aprovado em 05 de Fevereiro de 2021:

BANCA EXAMINADORA

Prof. Dr. Jarbas Joaci de Mesquita Sá Junior(Orientador)
Universidade Federal do Ceará –
PPGEEC/UFC – Campus de Sobral

Prof. Dr. Iális Cavalcante de Paula Júnior
Universidade Federal do Ceará –
PPGEEC/UFC – Campus de Sobral

Prof. Dr. Guilherme de Alencar Barreto
Universidade Federal do Ceará –
PPGETI/UFC

À vida, ao universo e a tudo mais.

AGRADECIMENTOS

Agradeço à minha mãe e ao meu pai, pelo apoio incondicional. Aos meus amigos e colegas de mestrado, por compartilharem comigo este percurso e todos os seus contratempos e sucessos, com especial atenção a Brena, Joniel, Márcio, Rafael e Syllas. Ao meu orientador, pelas sugestões e por estar sempre atento ao detalhes deste trabalho.

"Don't wait for a light to appear at the end of the tunnel, stride down there and light the bloody thing yourself."

(Sara Henderson)

RESUMO

Análise de atravessabilidade é essencial para operações de robôs terrestres, pois permite a incorporação de conhecimento acerca de terrenos atravessáveis e não-atravessáveis aos algoritmos de planejamento. É possível usar dados aéreos para calcular mapas de atravessabilidade para regiões amplas e usá-los para auxiliar a navegação de robôs em terra. A maioria dos métodos publicados para produção desses mapas a partir de imagens aéreas utiliza classificação de terrenos ou heurísticas para determinar atravessabilidade. No entanto, métodos baseados em classificação ou heurísticas delineadas manualmente exibem limitações quanto à qualidade das saídas e tempos de processamento. Com o propósito de aprimorar essas variáveis, o presente trabalho explora uma nova maneira de computar mapas de atravessabilidade a partir de imagens aéreas, em uma passagem direta por uma rede totalmente convolucional. Mostra-se que este método pode ser empregado para gerar mapas em ambientes diversos e demonstra-se que os mapeamentos proveem informações úteis para que um algoritmo de planejamento construa percursos seguros.

Palavras-chave: Navegação robótica. Atravessabilidade. Planejamento. Imagens aéreas. Redes totalmente convolucionais.

ABSTRACT

Traversability analysis is essential for ground robot operations because it allows the incorporation of knowledge about traversable and non-traversable terrain into the robot's planning algorithm. It is possible to use aerial data to compute traversability maps for large regions and use them to assist in ground robot navigation. Most published methods to generate these maps from aerial images use terrain classification or heuristics to determine traversability. However, methods based on classification or handcrafted heuristics exhibit limitations with regard to output quality and processing time. To improve on these variables, this work explores a new method to compute a traversability map from an aerial image in one forward pass through a fully convolutional network. It is shown that this method can be employed to generate traversability maps in diverse environments and demonstrated that it provides useful information for a planner algorithm to compute safe paths.

Keywords: Robot navigation. Traversability. Planning. Aerial images. Fully convolutional networks.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3D	tridimensional
ALE	do inglês, ***A**utomatic **L**abeling **E**nvironnment*
ATPD	do inglês, ***A**erial **T**raversability and **P**lanning **D**ataset*
CNN	do inglês, ***C**onvolutional **N**eural **N**etwork*
CE	do inglês, ***C**ross **E**ntropy*
DDR4	do inglês, ***D**ouble **D**ata **R**ate Type **4***
DEM	do inglês, ***D**igital **E**levation **M**odel*
DSM	do inglês, ***D**igital **S**urface **M**odel*
FCN	do inglês, ***F**ully **C**onvolutional **N**etwork*
GPU	do inglês, ***G**raphics **P**rocessing **U**nit*
HSV	do inglês, ***H**ue, **S**aturation and **V**alue*
LIDAR	do inglês, ***Li**ght **D**etection **A**nd **R**anging*
MAE	do inglês, ***M**ean **A**bsolute **E**rror*
MSE	do inglês, ***M**ean **S**quared **E**rror*
MLP	do inglês, ***M**ulti-**L**ayer **P**erceptron*
PRM	do inglês, ***P**robabilistic **R**oadmap **M**ethod*
RAM	do inglês, ***R**andom **A**ccess **M**emory*
ReLU	do inglês, ***Re**ctified **L**inear **U**nit*
RGB	do inglês, ***R**ed **G**reen **B**lue*
RRT	do inglês, ***R**apidly-Exploring **R**andom **T**ree*
SLAM	do inglês, ***S**imultaneous **L**ocalization **A**nd **M**apping*
SVM	do inglês, ***S**upport **V**ector **M**achine*
TFCN	do inglês, ***T**raversability **F**ully **C**onvolutional **N**etwork*

SUMÁRIO

1 INTRODUÇÃO

Um mapa de atravessabilidade é uma estrutura que permite distinguir terrenos atravessáveis de não-atravessáveis e aplicar esse conhecimento em estratégias de navegação autônoma (SERAJI, 1999; PAPADAKIS, 2013). Nesse mapa, cada região possui um valor de atravessabilidade, que é uma quantificação acerca do quanto ela é adequada à travessia por um robô móvel. Esse conceito foi extensivamente aplicado a tarefas de planejamento e pesquisadores do ramo utilizaram uma gama de diferentes sensores e métodos para estimativa de atravessabilidades a partir do chão (HOWARD; SERAJI; WERGER, 2003; CASTEJÓN et al., 2005; GUO et al., 2008; SHNEIER et al., 2008; LINHUI et al., 2013). Com a popularidade crescente de *drones* multirrotores equipados com câmeras e escaneadores a laser, uma ideia para aprimorar a navegação robótica no solo foi proposta: usar dados aéreos para incrementar a capacidade de planejamento a longas distâncias de veículos terrestres. Essa ideia foi explorada em diversos trabalhos, nos quais foram desenvolvidos novos métodos para construção de mapas de atravessabilidade a partir de múltiplas modalidades sensoriais (VANDAPEL; DONAMUKKALA; HEBERT, 2006; HUDJAKOV; TAMRE, 2013; YANG; TSENG; TSENG, 2015; DELMERICO et al., 2017; CHRISTIE et al., 2017; CHAVEZ-GARCIA et al., 2018; FEDORENKO; GABDULLIN; FEDORENKO, 2018; PETERSON et al., 2018; KIRAN; KUMAR; MOHAN, 2019).

De acordo com Papadakis (2013), pode-se distinguir três abordagens principais para análise de atravessabilidade: proprioceptiva, geométrica e visual. No domínio proprioceptivo, os dados de entrada são adquiridos de sensores acoplados ao veículo, como unidades de medição inercial, sensores de derrapamento, colisão e outros equipamentos capazes de inferir o estado do robô enquanto ele atravessa determinado terreno. Métodos geométricos são aqueles que usam dados obtidos de escaneadores tridimensionais como sensores LIDAR (do inglês, *Light Detection and Ranging*) ou reconstruções 3D advindas de visão estereoscópica (VANDAPEL; DONAMUKKALA; HEBERT, 2006; CHAVEZ-GARCIA et al., 2018; FEDORENKO; GABDULLIN; FEDORENKO, 2018; KIRAN; KUMAR; MOHAN, 2019). Os métodos visuais, por sua vez, usam atributos relacionados à aparência do terreno, seja através de imagens registradas por câmeras convencionais ou por sensores de imagem especializados, como câmeras infravermelhas ou termais (HUDJAKOV; TAMRE, 2013; YANG; TSENG; TSENG, 2015). Há também os métodos híbridos, que combinam duas ou até mesmo as três abordagens mencionadas (DELMERICO et al., 2017; CHRISTIE et al., 2017; PETERSON et al., 2018; GUO et al., 2011).

Concentrando-se nas abordagens visuais, pode-se identificar três estratégias para processar a imagem de entrada e derivar um mapa de atravessabilidade: heurísticas, regressão e classificação de terreno. Heurísticas normalmente usam algoritmos estatísticos,

critérios baseados nas características dos sensores ou conjuntos de regras para inferir a atravessabilidade de regiões. Embora haja a possibilidade de desenvolver esquemas de regressão para aferir custos de travessia, as técnicas mais difundidas são aquelas baseadas em classificação. Inicialmente, um classificador de terrenos é treinado para assinalar rótulos de classe a *pixels* ou regiões da imagem. Então, para construir o grafo em que será realizada a busca por caminhos, duas alternativas têm sido utilizadas. Na primeira, o grafo de busca é construído aplicando-se custos de travessia fixos a cada classe de terreno (e.g., estradas possuem menor custo que gramados, enquanto construções e demais obstáculos possuem custo infinito) (HUDJAKOV; TAMRE, 2013; DELMERICO et al., 2017; PETERSON et al., 2018). A segunda alternativa consiste em usar equações mais sofisticadas para atribuir pesos às arestas, considerando as classificações de regiões vizinhas e ponderando seus custos de forma correspondente (CHRISTIE et al., 2017). Trabalhos que utilizam classificadores normalmente empregam arquiteturas de CNN, por conta de seu excelente desempenho em tarefas de classificação e segmentação de terrenos (LÄNGKVIST et al., 2016). Como exemplos, pode-se citar Hudjakov e Tamre (2013), Linhui et al. (2013), Delmerico et al. (2017).

Um dos problemas nessas estratégias é que a maioria de seus exemplares utiliza métodos que requerem processamento individual de milhares de pequenas regiões da imagem, o que pode ser computacionalmente custoso, especialmente no caso de regressão ou classificação. No caso das heurísticas, mesmo que sua complexidade seja baixa, é difícil levar em conta toda a possível variabilidade das imagens usando apenas regras desenvolvidas manualmente. Uma heurística pode obter bom desempenho em um ambiente, mas necessitar ser reescrita para outro. Uma interessante solução para esses problemas é o uso de redes totalmente convolucionais (FCNs, do inglês *Fully Convolutional Networks*) para as estimativas de atravessabilidade ou segmentação de espaços livres. Tais arquiteturas são capazes de processar uma imagem inteira sem necessidade de particionamento explícito. Trabalhos como Schilling et al. (2017), Hamandi, Asmar e Shammas (2018), Ono et al. (2018), Yang et al. (2019), Zhou et al. (2019), Iwashita et al. (2019) e Martinez-Soltero et al. (2020) propõem usar FCNs para determinar atravessabilidade através de segmentação semântica de ambientes ao ar livre. Lüddecke, Kulvicius e Wörgötter (2019) utiliza FCNs para classificar acessibilidade para robôs em ambientes internos. Já Wulfmeier et al. (2017) usa aprendizado por reforço baseado em FCNs para aprender funções de custo de travessia para fundamentar o planejamento de caminhos.

O objetivo principal deste trabalho é desenvolver um método de análise de atravessabilidade visual para imagens aéreas que utilize aprendizado de máquina e seja capaz de realizar estimativas úteis para o cálculo de travessias de veículos autônomos terrestres. Tal método também precisa ter desempenho de tempo adequado, realizando estimativas na ordem de segundos ou microssegundos em áreas de centenas de metros quadrados. Este objetivo principal abrange um conjunto de objetivos específicos como:

- desenvolvimento de uma arquitetura de rede totalmente convolucional capaz de aprender a estimar atravessabilidades;
- integração de um algoritmo de planejamento de caminhos;
- avaliação do desempenho da rede convolucional na estimativa de atravessabilidades;
- avaliação do desempenho conjunto do método de análise de atravessabilidade e do planejador de caminhos;
- comparação com outros métodos de análise de atravessabilidade visual descritos na literatura;
- obtenção de desempenho similar ou superior a outros métodos presentes na comparação, considerando-se quesitos como qualidade dos mapeamentos produzidos, qualidade dos caminhos, tempo de processamento, adaptabilidade e robustez em diferentes ambientes.

O presente trabalho contribui para a área de pesquisa correspondente através da proposição e avaliação de um método de aprendizado supervisionado para cálculo de atravessabilidades na abordagem visual. A técnica proposta utiliza uma arquitetura totalmente convolucional para regredir atravessabilidades a partir de imagens aéreas, uma ideia pouco explorada na literatura. Esta dissertação apresenta duas melhorias em relação ao trabalho anteriormente publicado (BORGES et al., 2019). Os avanços estão no uso de uma FCN para cálculo de atravessabilidades, etapa que era antes realizada por heurísticas; e a determinação automática do limiar de corte do grafo de atravessabilidade, usado na fase de planejamento, valor que antes era definido manualmente. Essas melhorias estão descritas, respectivamente, nas seções 3.2 e 3.3.

Esta dissertação está dividida da seguinte maneira: no capítulo 2 são apresentados os fundamentos necessários à compreensão do trabalho, que incluem a navegação robótica, abordagens de cálculo de atravessabilidades e redes neurais convolucionais. A metodologia proposta, a arquitetura convolucional utilizada, o conjunto de dados, seu treinamento, validação e demais detalhes são expostos no capítulo 3. Avaliações quantitativas e qualitativas do método e também comparações com outras técnicas usadas na literatura, dentro do contexto da abordagem visual de atravessabilidades sobre imagens aéreas, são apresentadas no capítulo 4. Enfim, o capítulo 5 fecha a dissertação com as conclusões acerca do trabalho e ideias de futuros desenvolvimentos.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo explora os principais conceitos necessários à compreensão do presente trabalho: navegação robótica, análise de atravessabilidade e redes neurais convolucionais. Na seção 2.1, é feita uma revisão acerca da navegação robótica e seus subproblemas. A seção 2.2 contém uma definição de atravessabilidade e os principais métodos usados para seu cálculo. Enfim, na seção 2.3, faz-se uma revisão sobre os componentes da arquitetura de redes neurais convolucionais, suas camadas, técnicas de treinamento e o conceito de rede totalmente convolucional.

2.1 Navegação robótica

A habilidade de locomoção autônoma possui elevada importância para um robô, pois reduz em grande parte a necessidade de um controlador humano. O principal objetivo da navegação robótica é determinar uma sequência de operações que levarão o sistema de sua posição inicial a uma posição final designada (GUL; RAHIMAN; ALHADY, 2019). A realização dessa tarefa, por sua vez, requer conhecimento acerca do ambiente em que o robô está inserido. Também é necessário manter um conjunto de informações de estado e posicionamento. Nesta dissertação, divide-se o problema da navegação robótica em quatro subproblemas: mapeamento, localização, reconhecimento de objetivo e planejamento de rota. Apesar de algumas fontes descreverem o problema usando três questões principais: "onde estou?", "para onde vou" e "como chegar lá?", correspondentes, respectivamente, a localização, reconhecimento de objetivo e planejamento de rota, inclui-se neste trabalho o mapeamento, visto que é de fundamental importância para as demais atividades (LEONARD; DURRANT-WHYTE, 1991; NEGENBORN, 2003). Nesta seção, os quatro subproblemas da navegação robótica são apresentados.

2.1.1 Mapeamento

No contexto da navegação robótica, mapear uma região significa produzir uma representação do ambiente a partir de dados de sensores, sejam eles acoplados ao robô ou posicionados à distância. O mapa pode fornecer informações acerca da disposição ou natureza de obstáculos, variações de elevação do solo ou outras características que possam interferir na locomoção do sistema. Existem três abordagens convencionais para representação do mapa: métrica, topológica e semântica (YI; JEONG; CHO, 2012). Esses métodos podem também ser combinados para produzir um mapa mais informativo, o que torna possível o aproveitamento conjunto de suas vantagens, a depender dos requerimentos do sistema e da capacidade dos algoritmos de navegação para lidar com essas informações

(THRUN, 1998; LIM; FRAHM; POLLEFEYS, 2012; WALTER et al., 2013; BERNUY; SOLAR, 2015; DROUILLY; RIVES; MORISSET, 2015).

2.1.1.1 Abordagem métrica

Na representação métrica, o ambiente é descrito através de uma grade igualmente espaçada, na qual a presença de obstáculos é indicada pelo conteúdo de cada uma de suas células (THRUN, 1998). Nessa abordagem, os mapas são de fácil compreensão, construção e manutenção, visto que basta dividir o ambiente em sub-regiões e armazenar suas características em uma matriz ou estrutura de dados similar. Outra vantagem concentra-se em sua capacidade de prover informações de localização sem ambiguidade ou dependência de ponto de vista, porque uma coordenada no mapa tem relação unívoca com uma posição no ambiente. O planejamento de caminhos também é facilitado, pois a conversão da grade em um grafo pode ser realizada de forma simples, através do pareamento entre células da grade e vértices, sendo a existência e os pesos das arestas atribuídos a partir das características de cada célula e de suas distâncias entre si. Uma desvantagem desse método é o alto consumo de espaço de armazenamento, uma vez que é necessário manter em memória todas as regiões do ambiente, mesmo que não contenham informações relevantes. Também faz-se necessário obter uma acurácia elevada por parte dos algoritmos de localização do robô, visto que suas ações dependem fortemente do conhecimento correto de sua posição.

2.1.1.2 Abordagem topológica

Na representação topológica, a descrição do ambiente é feita através de grafos em que os vértices representam pontos de referência e a presença de arestas é determinada pela existência de caminhos diretos entre esses pontos (THRUN, 1998). A utilização de mapeamento topológico proporciona uma utilização mais eficiente da memória, pois apenas os pontos de referência e suas conexões precisam ser armazenados. Esse mesmo fator permite que o planejamento de caminhos seja realizado de maneira mais rápida em comparação com a abordagem métrica. As principais desvantagens desse método são a possível ambiguidade de localização e a susceptibilidade à produção de caminhos subótimos. O motivo encontra-se no fato de que, nessa abordagem, o robô determina sua localização de maneira relativa aos pontos de referência nas suas proximidades. Caso haja pontos com propriedades similares, torna-se incerta a real localização do robô, visto que, a princípio, ele poderia estar próximo a qualquer um desses locais.

2.1.1.3 Abordagem semântica

O método semântico funciona como um complemento às abordagens métrica e topológica, com o propósito de facilitar a interação entre humanos e máquinas. A ideia é assinalar rótulos semânticos a coordenadas, objetos ou pontos de referência dentro

do mapa, de maneira que o robô possa relacioná-los com termos usados na linguagem humana (WALTER et al., 2013). Essa é uma abordagem mais bem adaptada ao uso de algoritmos de planejamento simbólicos ou baseados em linguagem natural. Por exemplo, comandos expressos por um humano, como "Prossiga pelo corredor, vire à direita e entre no laboratório." possuem significado nítido dentro de um mapa com informações semânticas.

2.1.2 Localização

O problema da localização consiste em definir o posicionamento do robô em relação ao ambiente. Esse é um passo importante, pois permite avaliar o estado atual do sistema e determinar suas possibilidades de ação. A localização é considerada fácil para situações em que o ambiente é estático, estruturado, de tamanho limitado ou caso tenha sido previamente mapeado. Para essas condições, existem diversos métodos capazes de localizar o sistema robótico com alta acurácia e precisão (HUANG; DISSANAYAKE, 2016). No entanto, a complexidade do problema cresce de maneira significativa em ambientes desconhecidos, dinâmicos, não-estruturados e de larga escala (SICILIANO; KHATIB, 2008). Nos casos em que o ambiente é desconhecido, não existe inicialmente um mapa ou sistema de referência para guiar os algoritmos de posicionamento, sendo, portanto, necessário realizar simultaneamente mapeamento e localização, o que deu origem ao paradigma denominado SLAM (do inglês, *Simultaneous Localization and Mapping*). A literatura que concerne a localização e suas variantes mais complexas, como o SLAM, é vasta e encontra-se em desenvolvimento ativo. Para aprofundamento no tema, recomenda-se a leitura de Siciliano e Khatib (2008), Huang e Dissanayake (2016), Saeedi et al. (2016) e Bresson et al. (2017).

2.1.3 Reconhecimento de objetivo

O problema geral de determinar um objetivo de maneira automática, embora tenha grande importância para a implementação de robôs completamente autônomos, ainda é um campo pouco explorado na literatura. É possível que parte do motivo para essa lacuna seja o fato de que grande parte das pesquisas em navegação robótica concentra-se em desafios cujos objetivos são bem estabelecidos. Por exemplo, alcançar um ponto específico no mapa, encontrar a melhor rota entre múltiplos pontos pré-determinados, buscar um objeto ou seguir um alvo móvel. Em todos esses casos, a definição do objetivo primário é realizada diretamente pelo desenvolvedor ou operador do sistema. Essa característica define o paradigma da inteligência artificial fraca, que consiste no desenvolvimento de sistemas com foco em apenas um problema. Nesse caso, a melhor aproximação é o reconhecimento de sub-objetivos, ou seja, pré-requisitos para a completude do objetivo estabelecido ou busca por estados intermediários entre o estado atual e o desejado (YE; WEBB, 2009). Visto que a natureza dos sub-objetivos é fortemente dependente da aplicação, é comum que essa etapa seja implementada utilizando-se estratégias especializadas. Um sistema robótico autônomo para vigilância, por exemplo, precisa de algoritmos específicos que o instruam a

localizar objetos perdidos e movidos ou detectar e seguir pessoas (PAOLA et al., 2010). No entanto, há trabalhos que buscam otimizar a comunicação de objetivos entre humanos e robôs, utilizando linguagem natural (WALTER et al., 2013) ou até mesmo interfaces cérebro-máquina (KUHNER et al., 2019). Outro ramo crescente na literatura robótica é o desenvolvimento de arquiteturas cognitivas, cujo propósito é estabelecer plataformas de processamento adequadas a objetivos e ambientes de alta complexidade (SALGADO et al., 2016; BUSTOS et al., 2019).

2.1.4 Planejamento de rota

Planejar uma rota ou caminho consiste em determinar uma cadeia de ações que conduzam o veículo robótico de sua posição inicial até a final. A depender da aplicação e dos recursos disponíveis, também é importante estabelecer esse caminho de maneira a otimizar a distância percorrida, tempo de locomoção, gasto de combustível, segurança da travessia ou outros critérios de eficiência. Essa tarefa é normalmente dividida em duas etapas: planejamento global e planejamento local (MARÍN et al., 2018). As diferenças entre as etapas concentram-se na natureza e na escala dos obstáculos considerados durante o processo. Ambas as modalidades de planejamento e alguns dos métodos utilizados na literatura são expostos brevemente nas seguintes subseções. Para mais detalhes, recomenda-se a leitura de Souissi et al. (2013), Yang et al. (2016) e González et al. (2016).

2.1.4.1 Planejamento global

O planejador global trabalha em uma escala consideravelmente maior do que as dimensões do veículo robótico, sendo, portanto, capaz de antever obstáculos a longas distâncias e produzir uma primeira aproximação do caminho. Uma das técnicas mais comumente aplicadas para essa fase é a busca em grafos, na qual o espaço de localizações é representado por vértices cujas arestas são possíveis travessias. Os pesos associados às arestas representam os custos de transitar entre dois estados e seu valor é determinado por uma função que inclui as variáveis que se deseja otimizar, como distância, tempo, uso de combustível e segurança. Tal abordagem permite o uso de algoritmos clássicos como Dijkstra (DIJKSTRA, 1959) e A* (HART; NILSSON; RAPHAEL, 1968). Uma alternativa que busca solucionar restrições temporais é o uso de planejadores baseados em amostragem. Nesse método, os estados do mapa são explorados randomicamente. Exemplos bastante empregados na literatura são PRM (do inglês, *Probabilistic Roadmap Method*) (KAVRAKI et al., 1996) e RRT (do inglês, *Rapidly-Exploring Random Tree*) (LAVALLE; KUFFNER, 2001). Cada um dos algoritmos citados possui diversas variantes, desenvolvidas com o propósito de solucionar defeitos ou aplicar extensões ao método base. O trabalho de Souissi et al. (2013) aborda um amplo conjunto de extensões do algoritmo A*, dentro as quais destacam-se D* (STENTZ, 1994), Field D* (FERGUSON; STENTZ, 2007) e Theta* (DANIEL et al., 2014). Para aprofundamento em relação a técnicas baseadas em

PRM e RRT, recomenda-se a leitura de Geraerts e Overmars (2004) e Véras, Medeiros e Guimaráes (2019), respectivamente.

2.1.4.2 Planejamento local

O planejador local, por sua vez, recalcula variações da rota em uma escala menor, nas proximidades do caminho proposto pelo planejador global, com o propósito de suavizar o trajeto e evitar obstáculos de dimensões similares ou inferiores ao robô. É também nessa fase que ocorre maior ênfase nas restrições cinemáticas do sistema e o uso de mecanismos proprioceptivos como acelerômetros, giroscópios e sensores táteis. Isso implica que os algoritmos de planejamento local são mais dependentes da topologia do robô e da disposição de seus sensores do que os de planejamento global. A consequência disso é uma dificuldade de generalização, visto que cada algoritmo precisa ser adaptado aos recursos disponíveis. Apesar disso, pode-se apontar métodos que servem como fundamento para a implementação da fase local de planejamento. Uma das principais categorias de técnicas usadas nessa fase é a família de planejadores por interpolação de curvas, que inserem novos pontos sobre o caminho global para melhorar a continuidade da trajetória, considerando características do veículo e a possível natureza dinâmica do ambiente. Alguns exemplos dessa família são linhas e círculos (HORST; BARBERA, 2006), clotoides (FRAICHARD; SCHEUER, 2004), curvas polinomiais (PIAZZI et al., 2002), curvas de Bézier (HAN et al., 2010) e *splines* (BERGLUND et al., 2010). O planejamento local através de otimização numérica apresenta-se como outra opção. Nesse método, é definida uma função de otimização que leva em conta diversos atributos do trajeto, como velocidade, frenagem, restrições de rolamento do veículo, entre outras variáveis. Faz-se, então, a seleção da trajetória que minimiza essa função, de acordo com estimativas realizadas com métodos numéricos (GONZÁLEZ et al., 2016).

2.2 Análise de atravessabilidade

No contexto da navegação robótica, a análise de atravessabilidade é o processo que determina o quanto uma região é adequada à passagem do sistema. Esse processo faz parte da fase de mapeamento, visto que é uma maneira de aprimorar o conhecimento do sistema acerca do ambiente. Em suma, os métodos responsáveis por essa análise utilizam um conjunto de atributos da região para estabelecer um valor numérico que representa seu grau de atravessabilidade. A natureza desses atributos apresenta grande variabilidade, no entanto, é possível delimitar três principais abordagens: proprioceptiva, geométrica e visual. Cada abordagem apresenta vantagens e limitações. Por isso, é também comum a hibridização dos métodos, com uso de duas ou até mesmo as três categorias. Esta seção contém descrições das abordagens para análise de atravessabilidade. Para uma cobertura mais aprofundada desse tópico, recomenda-se a leitura de Papadakis (2013).

2.2.1 Abordagem proprioceptiva

Segundo Taylor (2009), a propriocepção é o mecanismo que nos permite perceber a localização, o movimento e as ações de partes do nosso próprio corpo e é possibilitada por um conjunto de receptores sensoriais nos músculos, pele e articulações. Na robótica, denominam-se sensores proprioceptivos aqueles que são instalados no próprio sistema e possuem a função de avaliar sua posição e estado. Exemplos são acelerômetros, giroscópios, sensores de vibração, derrapagem e colisão, entre outros (SICILIANO; KHATIB, 2008).

Na abordagem proprioceptiva, a atravessabilidade é estimada utilizando as informações captadas a partir desses sensores, conforme o veículo trafega no ambiente. Essa é uma verificação empírica acerca do quanto as regiões são adequadas à travessia. Por exemplo, em uma heurística simples, ambientes nos quais se medem consideráveis ruídos nos sinais dos giroscópios e sensores de vibração podem indicar que a região apresenta empecilhos à passagem. Contudo, se esses sinais são suaves, pode-se avaliar que a região é apropriada para a navegação. Os métodos para essa avaliação variam desde a estimativa de parâmetros do solo com o método dos mínimos quadrados (IAGNEMMA; SHIBLY; DUBOWSKY, 2002), até a classificação com SVMs (GUO et al., 2011; BERMUDEZ et al., 2012) e redes neurais convolucionais (BIJO, 2019). Recomenda-se o estudo de Papadakis (2013) para uma lista mais aprofundada de métodos proprioceptivos.

A abordagem proprioceptiva é fundamental nos algoritmos de planejamento local, pois fornece informações importantes acerca das proximidades imediatas do sistema e de seus requisitos cinemáticos. Apesar disso, uma limitação significativa dessa abordagem é sua incapacidade de observar obstáculos e condições de terrenos além da atual posição. Por isso, é muito comum adicionar métodos exteroceptivos, ou seja, abordagens que utilizam informações externas ao sistema para avaliar a atravessabilidade do ambiente. Esse incremento possibilita que o robô estime atravessabilidades com antecedência, antes de realmente navegar sobre os terrenos (PAPADAKIS, 2013).

2.2.2 Abordagem geométrica

A abordagem geométrica é composta por métodos exteroceptivos que buscam construir uma representação tridimensional (3D) do ambiente para guiar a análise de atravessabilidade. Essa é a abordagem mais amplamente explorada, visto que mapas 3D proporcionam um conhecimento detalhado acerca do posicionamento de obstáculos, irregularidades do solo, profundidade de vales, entre outros possíveis empecilhos à navegação. O fato de métodos geométricos trabalharem com a forma tridimensional do ambiente garante também invariância a condições de iluminação, fumaça e sombras. Em compensação, o aparato sensorial necessário à implementação de métodos geométricos é mais caro e consome mais energia em relação aos sensores utilizados em outras abordagens.

As principais bases para análise de atravessabilidade dentro dessa abordagem são o modelo digital de elevação (DEM, do inglês, *Digital Elevation Model*) e o modelo digital da superfície (DSM, do inglês, *Digital Surface Model*). DEMs representam a forma 3D do terreno como uma função $z = f(x, y)$, que mapeia pontos (x, y) em um plano para um valor de elevação z (FLORIANI; MAGILLO, 2018). Os DSMs são representados da mesma maneira, mas incluem também informações de objetos físicos na superfície do terreno, como árvores, construções e outros obstáculos. Há uma enorme variedade de métodos para produção de DEMs e DSMs (FLORINSKY, 2016), entretanto, os mais comumente utilizados no contexto da navegação robótica são (i) fotogrametria (SHU et al., 2018), (ii) reconstrução de estrutura a partir do movimento (GUASTELLA. et al., 2017) e (iii) altimetria a laser, com uso de sensores LIDAR (FOROUTAN, 2020). As técnicas para inferir atravessabilidades a partir dos DEMs e DSMs também apresentam grande variedade e são, na maioria das vezes, utilizadas em conjunto, mas podem ser categorizadas em (i) processamento de sinais, (ii) estatísticas e (iii) aprendizado de máquina. Uma revisão mais completa acerca deste tema pode ser encontrada em Papadakis (2013).

2.2.3 Abordagem visual

Os métodos visuais ou, como chama Papadakis (2013), baseados em aparência, são aqueles que projetam o problema para os campos de processamento de imagens e visão computacional. Os sensores utilizados para reconhecimento do ambiente são câmeras, sendo mais comuns aquelas que varrem o espectro visível ou infravermelho. Imagens ou vídeos das regiões são captados e passam por fases de pré-processamento e extração de atributos. Esses atributos, por sua vez, são utilizados para realizar as estimativas de transitabilidade.

A ideia mais utilizada nessa abordagem é a conversão da análise de atravessabilidades em um problema de classificação de terrenos, cujas classes possuem custos de travessia previamente definidos. Exemplos de trabalhos que seguem essa metodologia são Hudjakov e Tamre (2013), Yang, Tseng e Tseng (2015), Delmerico et al. (2017), Christie et al. (2017) e Peterson et al. (2018). No entanto, há na literatura alguns métodos que realizam regressão, estimando uma medida de atravessabilidade diretamente a partir dos atributos visuais da região, como Guo et al. (2011) e Borges et al. (2019).

Em relação aos métodos geométricos, o uso de imagens tem a vantagem de economia de custo e energia. Além disso, câmeras podem ser facilmente ajustadas para focar em áreas de interesse e fornecer melhores resoluções de maneira dinâmica. Outra situação em que o uso da abordagem visual é mais adequado acontece no problema de estabelecer se determinada superfície é capaz de suportar o peso do robô. Nesse caso, é importante identificar o material da superfície ou analisar sua resposta térmica, características que somente podem ser extraídas a partir de métodos visuais (PAPADAKIS, 2013).

2.3 Redes neurais convolucionais

O advento da aprendizagem profunda impulsionou consideráveis avanços em diversas áreas de aplicação, com especial impacto na classificação, segmentação e detecção de objetos em imagens (DHILLON; VERMA, 2019). A ideia da aprendizagem profunda é construir sucessivas representações dos dados através do empilhamento de camadas de computação. Cada uma das camadas realiza uma transformação no espaço de seus atributos de entrada e fornece uma nova caracterização na saída. Essa técnica vem sendo empregada com sucesso incremental desde as proposições do perceptron multicamadas (MLP, do inglês *multilayer perceptron*) e de seu algoritmo de treinamento, conhecido como retropropagação (do inglês, *backpropagation*) (WASSERMAN; SCHWARTZ, 1988). No entanto, foram as redes neurais convolucionais (CNNs, do inglês, convolutional neural networks) que de fato revolucionaram os campos do processamento de imagens e visão computacional. Sua proposta inicial foi feita por Fukushima (1980), a incorporação do algoritmo de retropropagação foi feita por Waibel et al. (1989) e sua principal aplicação no princípio, dentro do contexto da visão, foi o reconhecimento automático de caracteres (DENKER et al., 1988; LECUN et al., 1989; LECUN et al., 1998). A utilização de CNNs teve um forte crescimento após a proposta do uso de unidades de processamento gráfico (GPUs, do inglês, *graphics processing units*) como *hardware* preferencial, fator que acelerou consideravelmente o tempo de treinamento, possibilitou o aumento de seu número de camadas e, consequentemente, de seu poder de computação (CHELLAPILLA; PURI; SIMARD, 2006). Desde então, diversas arquiteturas de CNN foram propostas, superando continuamente o estado da arte em diversas tarefas de visão computacional (DHILLON; VERMA, 2019). A implementação de *hardware* especializado para treinamento de CNNs, suas variantes e demais modelos de redes neurais também são ramos ativos de desenvolvimento (JOUPPI et al., 2017).

Uma CNN é composta por sequências de camadas de convolução, *pooling*, normalização e, em alguns problemas, um conjunto de camadas totalmente conectadas. Em muitas tarefas de visão computacional, a informação a ser processada é uma imagem representada por uma matriz. Contudo, a camada de entrada da CNN pode ser arquitetada para processar tensores de qualquer dimensionalidade. O tensor introduzido na camada de entrada é subsequentemente processado pelas camadas internas, nas quais são produzidas diferentes representações desse valor. Durante o treinamento, são fornecidos exemplos de tensores de entrada e do resultado esperado na camada de saída. Os algoritmos de otimização, normalmente baseados na retropropagação e no método do gradiente (RUMELHART; HINTON; WILLIAMS, 1986; LECUN et al., 1998), são responsáveis por ajustar os parâmetros das camadas internas de maneira que o resultado do processamento pelo conjunto de camadas do modelo aproxime-se do valor desejado. Os parâmetros internos podem, então, ser armazenados para posterior aplicação direta ou servir de base para treinar outros modelos, aproveitando-se do conhecimento já obtido, uma técnica denominada

transferência de aprendizagem (THRUN, 1998).

Esta seção contém breves apresentações das principais camadas e algoritmos utilizados na construção de CNNs. Também é apresentado um conjunto de CNNs especializadas para problemas cujas entradas e saídas são imagens ou tensores, denominadas redes totalmente convolucionais (FCNs, do inglês *fully convolutional networks*).

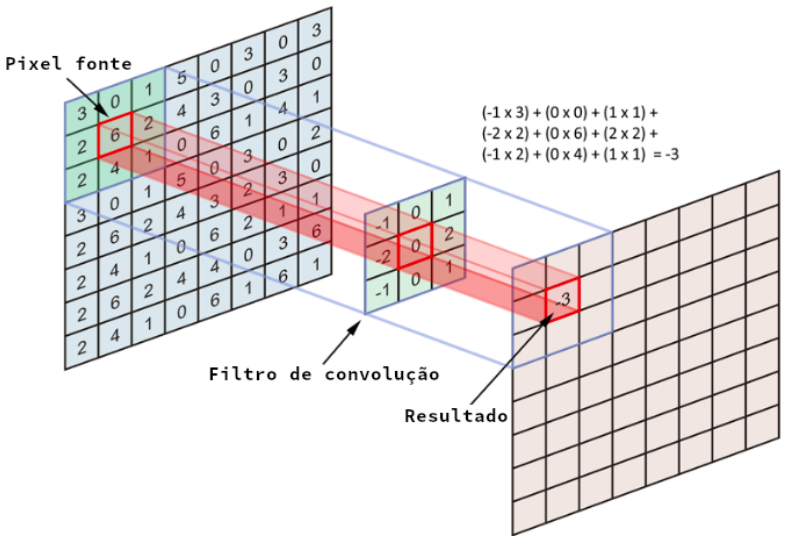
2.3.1 Camadas de convolução

No contexto do processamento de sinais, a convolução é uma operação genérica que determina a aplicação de um filtro sobre um sinal de entrada. Considerando uma imagem como um sinal bidimensional, temos que a convolução 2D pode ser definida como

$$(f * g)[x, y] = \sum_{m_1=-\infty}^{\infty} \sum_{m_2=-\infty}^{\infty} f[m_1, m_2] g[x - m_1][y - m_2] \tag{2.1}$$

Na Equação 2.1, f é a imagem de entrada, g o filtro de convolução e x e y são as coordenadas espaciais da imagem. Essa expressão descreve o processo de deslizar o filtro de convolução sobre a imagem de entrada e obter o produto escalar entre os elementos sobrepostos da imagem e do filtro para cada pixel, como exemplificado na Figura 1. A saída desse processo é denominada mapa de ativação.

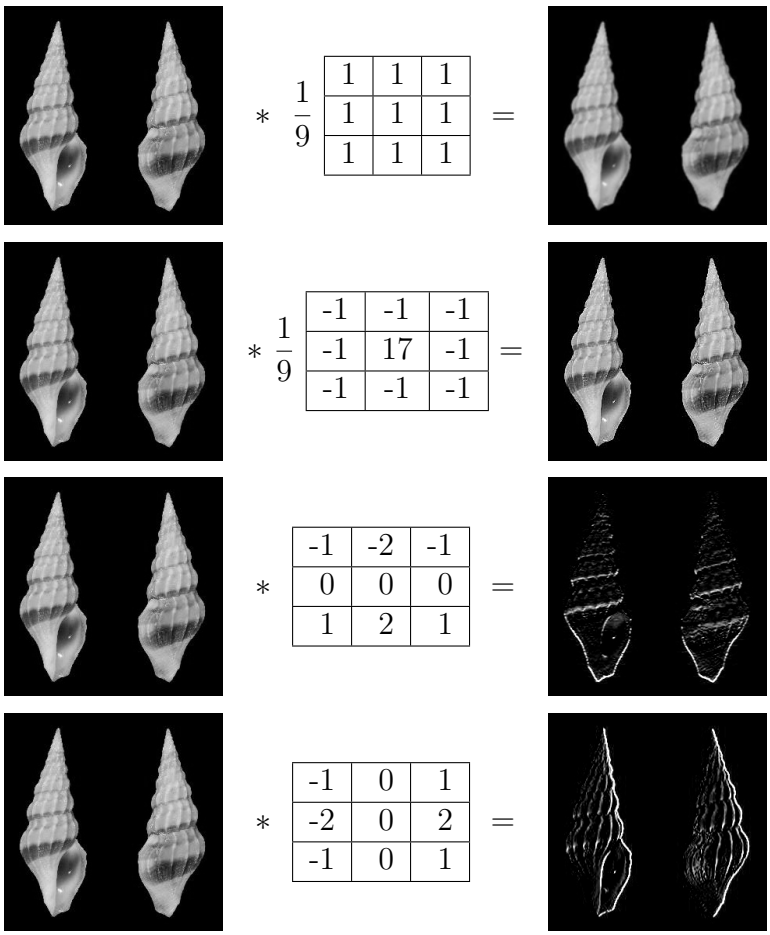
Figura 1 – Convolução entre a imagem e um filtro e o mapa de ativação resultante



Fonte: Adaptado de FreeCodeCamp (2018)

Na convolução, a natureza do filtro é capaz de definir operações sobre o conteúdo da imagem, portanto, essa é uma técnica bastante utilizada para realizar transformações. As operações de suavização, aguçamento e detecção de bordas, por exemplo, podem ser definidas através de convoluções com seus respectivos filtros (Figura 2).

Figura 2 – Suavização, aguçamento e detecção de bordas através de convoluções



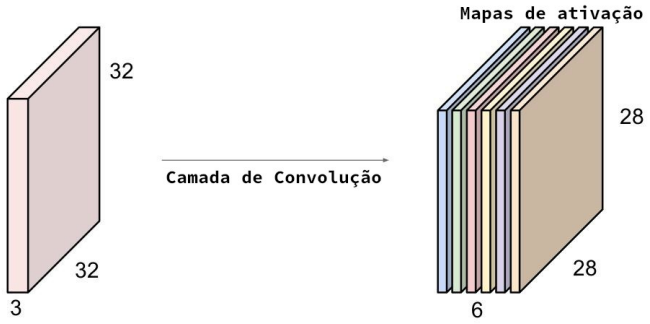
Fonte: Imagem das conchas de Kilburn e Stahlschmidt (2012), filtros aplicados pelo Autor (2021)

Os principais componentes de uma CNN são as camadas convolucionais, estruturas responsáveis por aplicar convoluções aos atributos de entrada, extrair informações relevantes e obter representações internas desses valores conforme o fluxo de passagem pela rede. A extração de informações e a construção de novas representações é dependente da natureza do filtro, portanto, o objetivo do treinamento nas camadas de convolução é definir automaticamente os valores de seus filtros de acordo com o contexto do problema. As camadas de convolução possuem outros parâmetros, por exemplo:

- Dimensões dos filtros: o tamanho dos filtros define a largura do campo receptivo de cada pixel e influencia na escala das características extraídas a partir dos dados de entrada. Filtros menores podem ser capazes de capturar detalhes, enquanto filtros maiores extraem informações genéricas, mais adequadas a identificar os componentes básicos da imagem;
- Quantidade de filtros: o número de filtros aplicados na camada de convolução define a profundidade do tensor de saída. Caso a camada possua N filtros, a passagem dos atributos pela camada produzirá N mapas de ativação, uma para cada filtro e, dessa maneira, o tensor de saída terá profundidade N (Figura 3). Cada filtro

é capaz de identificar uma característica em particular da imagem. Caso sejam usados poucos filtros pode ocorrer perda de informação, no entanto, uma grande quantidade de filtros pode aumentar demasiadamente a complexidade da rede e diminuir seu desempenho. Portanto, essa quantidade deve ser balanceada de acordo com o problema, consideração que em muitos casos é feita experimentalmente;

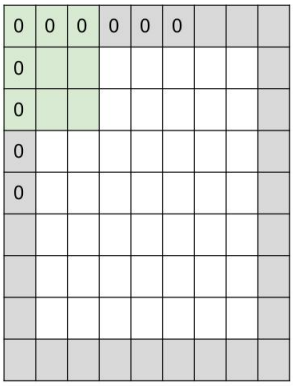
Figura 3 – Camada convolucional com $N = 6$ filtros



Fonte: Traduzido de Li, Johnson e Yeung (2019)

- Técnica de preenchimento: o processo de convolução requer que a imagem e o filtro sejam sobrepostos de maneira que seus elementos centrais estejam alinhados. Por conta disso, os *pixels* das bordas da imagem são perdidos, visto que é impossível centralizar o filtro sobre esses elementos. Os mapas de ativação, portanto, possuem dimensões menores que os tensores de entrada. Caso a rede seja muito profunda, é possível que ocorra perda de informação. Para solucionar esse problema, pode ser feito um preenchimento nas extremidades da imagem, o que aumenta seu tamanho e permite a centralização dos *pixels* de borda (Figura 4). Esse preenchimento pode ser feito de diversas maneiras, sendo comum a adição de zeros, replicação ou espelhamento de textura.

Figura 4 – Preenchimento das bordas da imagem com zeros

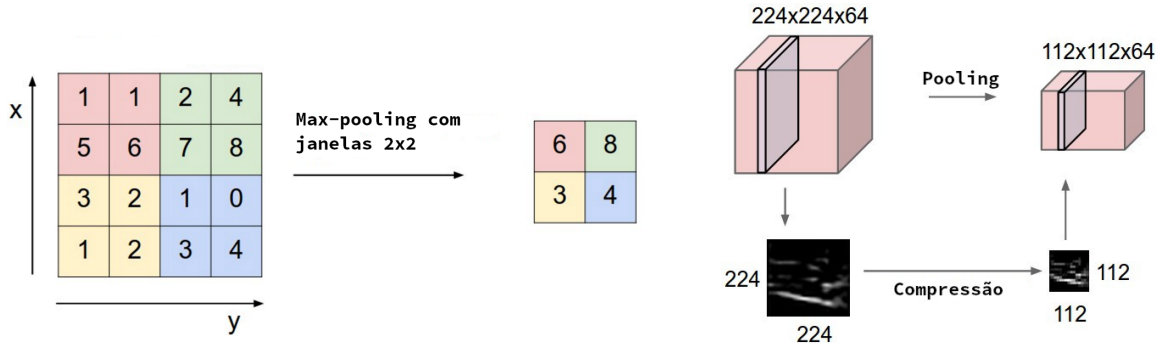


Fonte: Li, Johnson e Yeung (2019)

2.3.2 Camadas de pooling

A função das camadas de *pooling* é reduzir as dimensões espaciais dos mapas de ativação. Isso auxilia na redução de parâmetros da rede, torna o processamento e uso de memória mais eficiente, contribui para a prevenção do sobreajuste em modelos preditivos e acelera a convergência do treinamento (NAGI et al., 2011; DHILLON; VERMA, 2019). As camadas de *pooling* realizam agregação de atributos locais da imagem contidos em pequenas janelas. As dimensões dessa janela podem ser configuradas conforme a necessidade e definem o grau de compressão aplicado aos mapas de ativação. Existem diversas técnicas para a operação de agregação, sendo mais comum a utilização do *max-pooling*, cuja saída é o elemento de valor máximo dentro da janela. O processo que ocorre na camada de *pooling* é exemplificado na Figura 5. Pode-se observar no exemplo a aplicação da técnica *max-pooling* utilizando uma janela 2×2. O resultado é um mapa de ativação cujas dimensões de altura e largura foram reduzidas pela metade em relação ao mapa de entrada.

Figura 5 – Funcionamento da camada de *pooling*



(a) Técnica *max-pooling* (b) Compressão

Fonte: Adaptado de Li, Johnson e Yeung (2019)

2.3.3 Camadas de não-linearidade

Muitos problemas de interesse encontrados na natureza e na sociedade possuem características não-lineares, ou seja, não é possível modelar seu comportamento de maneira adequada usando apenas funções lineares. Logo, é importante que modelos de predição sejam capazes de contabilizar essa característica. No contexto das redes neurais, a não-linearidade é inserida no modelo através da utilização de funções não-lineares entre as camadas internas da rede. Esse ingrediente é um fator fundamental para a capacidade de generalização das redes neurais tradicionais (CYBENKO, 1989) e também das redes neurais convolucionais (ZHOU, 2020).

A camada de não-linearidade é a estrutura responsável por aplicar uma função de ativação a cada elemento de seu mapa de entrada, o que é equivalente a realizar transformações não-lineares no espaço de atributos. Há uma grande diversidade de funções de ativação utilizadas em CNNs. Alguns exemplos são as funções sigmoide, tangente

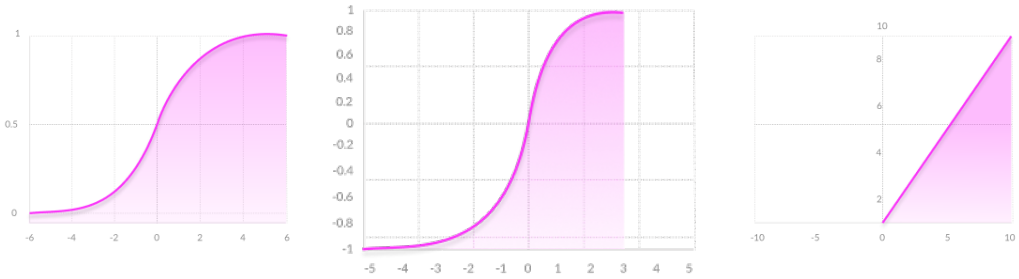
hiperbólica e ReLU (do inglês, *rectified linear unit*) (Equações 2.2, 2.3 e 2.4 e Figura 6). Apesar da multiplicidade de opções, verifica-se na literatura uma especial preferência pela função ReLU e suas variantes (RAMACHANDRAN; ZOPH; LE, 2017). A popularidade da ReLU deve-se principalmente ao seu mínimo custo computacional e sua propriedade de não-saturação, que levam a um treinamento mais ágil e mitigação do problema de desaparecimento do gradiente (GLOROT; BORDES; BENGIO, 2011).

$$f(x)_{sig} = \frac{1}{1 + e^{-x}} \tag{2.2}$$

$$f(x)_{tanh} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.3}$$

$$f(x)_{ReLU} = \max(0, x) \tag{2.4}$$

Figura 6 – Funções de ativação não-lineares



(a) Sigmoide (c) Tangente hiperbólica (b) ReLU

Fonte: Missing Link AI (2019)

2.3.4 *Normalização de lotes*

A ampliação do número de camadas de uma rede neural torna sua estrutura mais complexa e isso leva ao surgimento de novos problemas e preocupações. Um desses problemas é o que se chama de deslocamento interno de covariável (tradução do inglês, *internal covariate shift*). Ioffe e Szegedy (2015) definem o deslocamento interno de covariável como a flutuação de distribuição das entradas em diferentes camadas da rede conforme seus parâmetros são atualizados durante o treinamento. Conforme os parâmetros das camadas precedentes mudam, a distribuição de suas saídas sofre alterações, o que força a camada atual a reajustar-se à nova distribuição. O efeito dessa variação é o aumento do tempo de treinamento e a elevada sensibilidade da estrutura à inicialização dos parâmetros, o que produz, por sua vez, dificuldades na elaboração de novos modelos. A proposta de resolução desse problema, segundo os autores, reside na técnica de normalização de lotes.

A camada de normalização de lotes é responsável por fixar a média e o desvio padrão de suas entradas e prover mapas de ativação com distribuição estável para as

camadas posteriores. Detalhes de sua implementação podem ser explorados no trabalho de Ioffe e Szegedy (2015). Uma revisão acerca de seus efeitos pode ser encontrada na publicação de Bjorck, Gomes e Selman (2018). Apesar de a normalização de lotes ter sido desenvolvida para de reduzir o deslocamento interno de covariável, o real motivo de seu sucesso ainda é um tópico bastante discutido entre especialistas, com diversas hipóteses tendo sido propostas (SANTURKAR et al., 2018; KOHLER et al., 2019).

2.3.5 *Dropout*

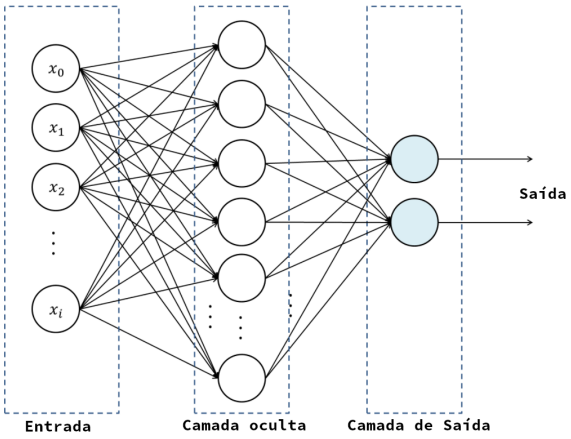
O fenômeno do sobreajuste (tradução do inglês para o termo *overfitting*) ocorre quando pequenas variações na distribuição de dados de teste causam uma queda de desempenho significativa no modelo (HINTON et al., 2012), uma preocupação comum no ramo da aprendizagem de máquina. A técnica de *dropout* é uma maneira de combater esse problema em redes neurais artificiais. Durante o treinamento do modelo, os parâmetros das unidades neuronais são afinados para a detecção de um certo tipo de padrão. No entanto, é possível que unidades vizinhas tornem-se dependentes entre si na realização de suas ativações. Essa conjuntura resulta na ultraespecialização de grupos de neurônios, para os quais uma pequena variação em relação aos dados de treinamento pode prejudicar sua acurácia local e, por conseguinte, afetar o desempenho da rede como um todo. O *dropout* é responsável por desativar aleatoriamente uma parcela dos neurônios em uma camada durante o treinamento, o que torna a aprendizagem de um neurônio menos dependente de seus vizinhos. A incerteza sobre quais unidades estarão presentes na operação força que uma unidade em particular aprenda uma maneira independente de representar os dados e fornecer ativações, fator capaz de impulsionar a robustez do modelo (SRIVASTAVA et al., 2014). Deve-se lembrar que a técnica de *dropout* é uma ferramenta utilizada apenas no momento de treinamento para reforçar o aprendizado de representações independentes e que todos os neurônios disponíveis são utilizados para realizar predições na fase de teste.

2.3.6 Camadas totalmente conectadas

Essas camadas são reminiscentes da tradicional arquitetura de redes neurais do tipo MLP. Seu nome advém da característica de que, nesse modelo, cada neurônio de uma camada precedente possui uma conexão com todos os neurônios da camada posterior (Figura 7). As intensidades dessas conexões são denominadas pesos e constituem os parâmetros treináveis da rede. Sua operação é equivalente a uma transformação linear de um vetor de entrada por uma matriz contendo os pesos das conexões entre as camadas.

Camadas totalmente conectadas são notáveis por sua simplicidade e capacidade de generalização, visto que, quando utilizadas em conjunto com camadas de não-linearidade, podem aproximar qualquer função ou curva de separação (CYBENKO, 1989). No contexto de redes neurais convolucionais, o uso de camadas totalmente conectadas ocorre nos

Figura 7 – Sequência de camadas totalmente conectadas



Fonte: Adaptado de An (2017)

estágios finais do processamento do modelo, principalmente em problemas de classificação e regressão (DHILLON; VERMA, 2019). A ideia é que as camadas anteriores sejam responsáveis pela extração automática de atributos, que, por sua vez, são repassados a um grupo de camadas totalmente conectadas, no qual é realizada a categorização ou predição.

2.3.7 *Redes totalmente convolucionais*

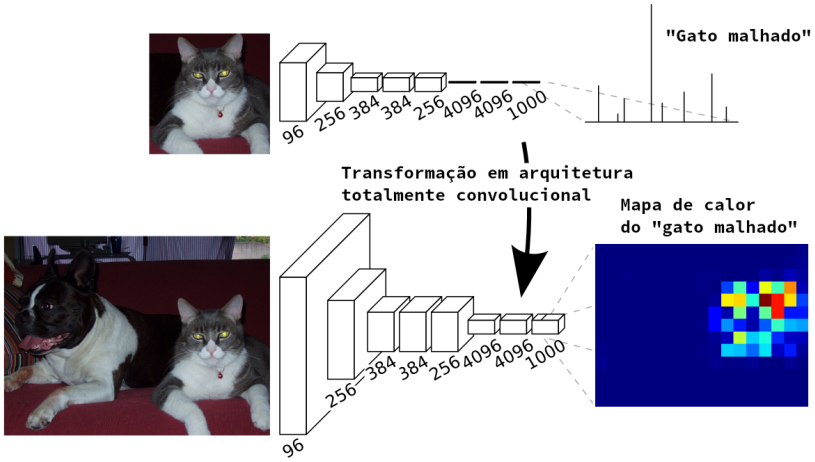
No modelo tradicional de CNN para classificação ou regressão, a imagem de entrada passa pelas camadas iniciais de convolução, *pooling*, não-linearidade e de normalização; nessas camadas são construídos mapas internos de representações em forma de tensor. Ao final, a representação tensorial é achatada e torna-se um vetor de atributos, uma representação unidimensional da imagem, que é fornecida a um conjunto de camadas totalmente conectadas para que seja feita a predição de um valor de saída. O uso de camadas totalmente conectadas ao final da arquitetura impõe certas restrições ao modelo:

- Entradas de dimensões fixas: a quantidade de neurônios na primeira camada totalmente conectada é fixa, portanto, é necessário que a representação tensorial da camada imediatamente anterior seja achatada para um comprimento pré-determinado. Isso só é possível se a imagem de entrada possuir, também, um tamanho fixo, para que sua passagem pelas camadas anteriores produza mapas de ativação com as dimensões requeridas pelas camadas posteriores. Sendo assim, um modelo tradicional só é capaz de processar imagens com dimensões pré-especificadas.
- Perda de informações espaciais: o achatamento implica na perda de informações de localização dos objetos da imagem. Isso ocorre porque um *array* multidimensional é reduzido a um vetor. As camadas totalmente conectadas, portanto, nada podem predizer acerca da posição de objetos na imagem ou da natureza de *pixels* individuais.

A principal característica das FCNs é a substituição das estruturas totalmente

conectadas por camadas convolucionais e de redimensionamento. Essa reestruturação resolve os problemas mencionados anteriormente, permitindo a entrada de imagens com dimensões variadas e habilitando as camadas finais da arquitetura a realizar predições sobre *pixels* individuais da imagem de entrada (Figura 8). Primeiramente, camadas convolucionais não estabelecem requisitos sobre as dimensões espaciais da imagem. Além disso, a exclusão da operação de achatamento possibilita a propagação das informações espaciais até as últimas camadas do modelo. Por fim, representações internas de dimensões arbitrárias podem ser submetidas a camadas de redimensionamento, que normalmente operam utilizando interpolações para reconstruir uma saída com o mesmo tamanho da imagem de entrada. Arquiteturas totalmente convolucionais são componentes que figuram nos modelos mais avançados de segmentação semântica (GUO et al., 2018) e detecção de objetos (SULTANA; SUFIAN; DUTTA, 2020).

Figura 8 – Transformação de uma CNN tradicional em uma FCN



Fonte: Traduzido de Shelhamer, Long e Darrell (2017)

2.3.8 Treinamento

O treinamento de uma CNN é um problema de otimização cujo propósito é determinar o conjunto de parâmetros W que minimiza a estimativa de erro do modelo. O conjunto W inclui os filtros das camadas convolucionais, os pesos de suas camadas totalmente conectadas e quaisquer outros parâmetros treináveis incluídos na arquitetura. A estimativa de erro é dada por uma função de perda $\mathcal{L}(y, \hat{y})$, que atribui um erro a cada dupla de saída real do modelo y e saída esperada $\hat{y}$. A literatura contém uma grande variedade de propostas de funções de perda, algoritmos de otimização e maneiras de combiná-los para resolução de problemas específicos. Esta subseção possui como objetivo apresentar de forma breve os conceitos por trás desses componentes do treinamento e apontar quais deles são mais frequentemente utilizados.

2.3.8.1 Funções de perda

No contexto de redes neurais, uma função de perda $\mathcal{L}(y, \hat{y})$ é uma operação matemática que determina uma distância entre o resultado na saída do modelo (y) e o valor esperado de acordo com os dados de treinamento ($\hat{y}$). Indiretamente, seu valor também depende dos parâmetros treináveis da arquitetura (W), pois esses são utilizados no cálculo de y. Devido a essa dependência, funções de perda podem ser usadas para indicar a qualidade do modelo, fornecendo uma maneira de diferenciar quais conjuntos de parâmetros são mais adequados ao tratamento do problema. Naturalmente, os valores de W que resultam em uma maior proximidade entre y e $\hat{y}$ são os mesmos que minimizam a função de perda e, portanto, tendem a produzir resultados mais acurados dentro do conjunto de treinamento.

As funções de perda podem ser categorizadas de acordo com o tipo de problema em que são aplicadas. Dentro do contexto do aprendizado supervisionado, a distinção feita por Wang et al. (2020) cita duas categorias: regressão e classificação. Em problemas de regressão, destacam-se o erro absoluto médio (MAE, do inglês, *mean absolute Error*) e o erro quadrático médio (MSE, do inglês, *mean squared error*). Já em problemas de classificação, pode-se enfatizar a entropia cruzada (CE, do inglês, *cross entropy*). As expressões dessas funções de perda são exibidas nas Equações 2.5, 2.6 e 2.7, respectivamente. O espectro de possibilidade, no entanto, é vasto. Para melhor explorar o arsenal de funções de perda já desenvolvidas, recomenda-se a leitura de Jadon (2020) e Wang et al. (2020).

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|. \tag{2.5}$$

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{2.6}$$

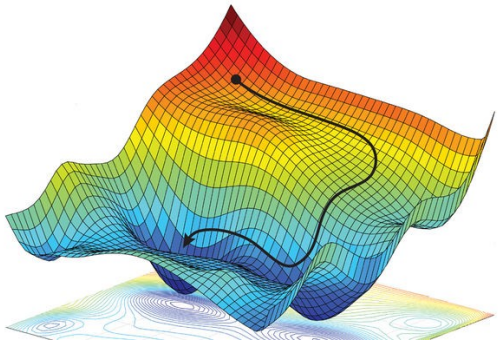
$$\text{CE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\epsilon + \hat{y}_i) \tag{2.7}$$

2.3.8.2 Algoritmos de otimização

Os algoritmos de otimização realizam a busca pelos valores dos parâmetros treináveis que minimizam a função de perda no conjunto de treinamento. Devido ao grande número de parâmetros envolvidos, é inviável fazer uma busca exaustiva e não há técnicas para cálculo direto. A estratégia utilizada nesse caso é a aplicação de métodos de aproximação iterativa, nos quais os parâmetros são continuamente refinados até a satisfação de um critério de parada. É possível visualizar o processo como um passeio pela superfície de perda, em que a direção do movimento se dá na direção dos mínimos (Figura 9).

A principal base dos algoritmos de otimização para redes neurais tradicionais e redes convolucionais é a mesma: ambas fundamentam-se na técnica de retropropagação

Figura 9 – Exemplo visual de processo de otimização em baixa dimensionalidade



Fonte: Towards Data Science (2018)

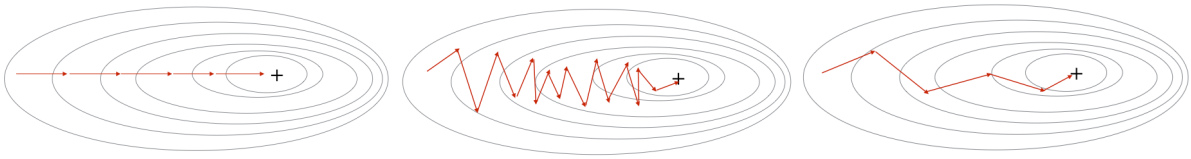
e no método do gradiente (RUMELHART; HINTON; WILLIAMS, 1986; LECUN et al., 1998). De maneira simplificada, a retropropagação é responsável por calcular o gradiente da função de perda em relação a cada parâmetro do conjunto W, ou seja, a taxa de variação da intensidade e da direção do erro em relação à mudança de cada parâmetro. O método do gradiente especifica que as taxas de variação obtidas na retropropagação sejam utilizadas para atualizar os valores dos parâmetros no sentido inverso ao do gradiente, operação que, por sua vez, tende a reduzir a perda obtida com os novos parâmetros calculados (GOODFELLOW; BENGIO; COURVILLE, 2016). Conforme as iterações são executadas, espera-se que a função de perda seja minimizada e o desempenho do modelo na tarefa seja otimizado até que ocorra a estagnação da curva de perda, situação denominada convergência.

No método do gradiente, há três maneiras de determinar a frequência de atualização dos parâmetros conforme a iteração pelas amostras de treinamento (Figura 10):

- *Batch*: atualização dos parâmetros ocorre a cada época;
- Estocástico: atualização ocorre a cada nova amostra;
- *Mini-Batch*: atualização ocorre a cada grupo de K amostras.

Nesse contexto, uma época refere-se a uma passagem completa pelo conjunto de treino. O algoritmo *batch* apresenta convergência e variações de erro mais estáveis, no entanto, devido a usar todas as amostras de treino a cada atualização, sua convergência é lenta em conjuntos de dados de grande porte. O método do gradiente estocástico agiliza o processo de convergência ao atualizar os parâmetros a cada nova amostra observada. Apesar disso, a variação dos parâmetros e do erro durante o processo é instável. Em muitos casos, essa instabilidade é irrelevante, pois o objetivo é alcançar um valor de mínimo, independentemente do caminho. Contudo, é possível balancear as vantagens dos algoritmos *batch* e estocástico através do uso de mini-lotes, como feito no método *Mini-batch*. Nessa implementação, as amostras de treinamento são subdivididos em lotes de tamanho K e os parâmetros são atualizados com o gradiente de erro calculado a cada lote.

Figura 10 – Variações básicas do método do gradiente



(a) Algoritmo *batch* (b) Algoritmo estocástico (c) Algoritmo *mini-batch*

Fonte: Adaptado de Deep Learning AI (2016)

Além da velocidade de convergência, outro desafio na execução do método do gradiente é a escapada de mínimos locais. Devido à natureza complexa da superfície de perda, é possível, e bastante provável, que o algoritmo eventualmente convirja para um mínimo local. No entanto, é importante que o método de otimização seja capaz de escapar de vales menos promissores e explorar uma área maior da superfície de perda. Isso é possível através da regulação da intensidade e direção de atualização dos parâmetros. Inicialmente, por exemplo, pode ser interessante realizar passos largos na superfície de erro, aplicando atualizações maiores a cada iteração, o que permite a exploração do espaço de possibilidades. Todavia, conforme o treino progride, essa intensidade de atualização precisa ser reduzida, de maneira a possibilitar a convergência. As variáveis que governam o grau de modificação dos parâmetros são a taxa de aprendizado α e o fator de momento μ:

$\alpha \rightarrow$ taxa de aprendizado: influencia a escala das atualizações dos parâmetros;
$\mu \rightarrow$ fator de momento: influencia a direção das atualizações dos parâmetros.

Existem diversas variações mais complexas do método do gradiente, capazes de alterar dinamicamente os valores de α e μ conforme a natureza dos dados e o progresso do treinamento. Dentre esses algoritmos adaptativos, pode-se citar Adagrad (DUCHI; HAZAN; SINGER, 2011), Adadelta (ZEILER, 2012) e Adam (KINGMA; BA, 2015).

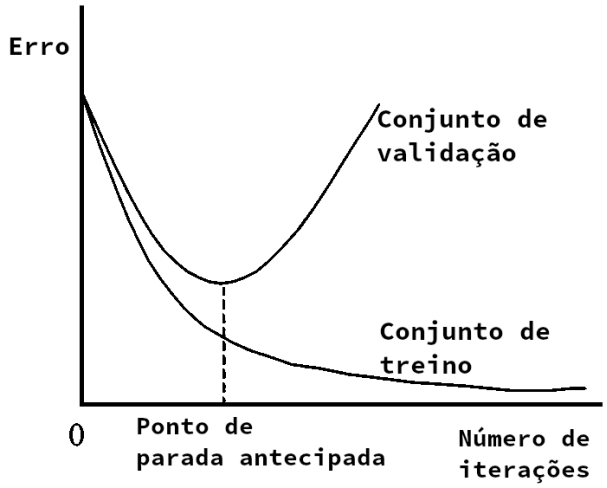
2.3.8.3 Critério de parada

O término do treinamento ocorre de acordo com um critério de parada, que pode estabelecer um número de máximo de iterações a realizar, um nível de desempenho a ser alcançado ou ainda a detecção de convergência para um mínimo local. Alguns critérios comumente usados são:

- Número máximo de épocas: o treinamento é prolongado até que seja atingido um determinado número de épocas. Nesse ponto, finaliza-se o treino. É um método simples, mas pode levar a desempenho ruim caso a variação de erro seja instável, pois a parada ocorre independentemente da avaliação do modelo;

- Limiar de perda: define-se um limiar máximo aceitável para a perda de treinamento ou validação interna e o treinamento ocorre até que esse limiar seja atingido. Nesse caso, indica-se o uso da perda de validação, visto que esse cálculo é realizado sobre amostras não apresentadas ao modelo durante o treino, o que a torna uma avaliação de qualidade mais confiável;
- Parada antecipada: é um método de detectar o momento de convergência do treinamento, ou seja, o ponto em que o modelo atinge a estabilidade em um mínimo local. Funciona através da análise da perda de validação após cada época e comparação com valores de perda anteriores. Se não é registrada uma melhoria na perda em um intervalo de p épocas, variável denominada "paciência", assume-se que o mínimo local foi atingido no início desse intervalo. O treinamento é então finalizado e os parâmetros obtidos no mínimo local são utilizados como resultado final da otimização. Esse método possui a vantagem de evitar divergências da perda prejudiciais, que podem ocorrer em caso de sobreajuste, como exibido na Figura 11.

Figura 11 – Gráficos de perda durante o processo de otimização com parada antecipada

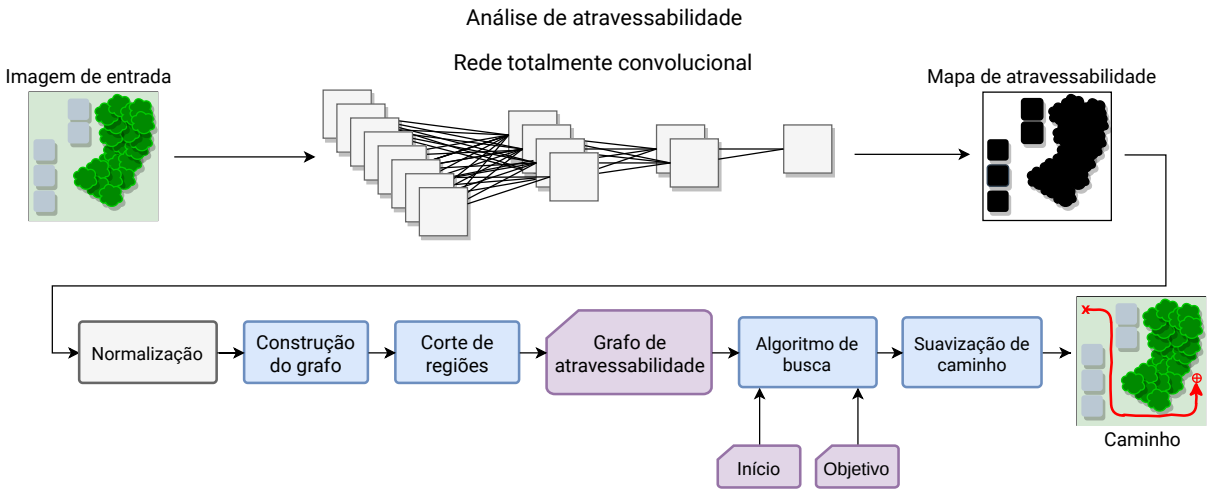


Fonte: Traduzido de Gençay e Qi (2001)

3 O MÉTODO PROPOSTO

A proposta do presente trabalho é descrita neste capítulo. Primeiramente, na seção 3.1, são apresentados os recursos necessários à implementação do método, com atenção ao conjunto de dados utilizado para treinamento e teste. O processo de análise de atravessabilidade e planejamento é mostrado na Figura 12. A imagem de entrada, uma vista aérea da região, é submetida a uma FCN treinada para estimar custos de travessia. A saída da rede consiste em um mapa de atravessabilidade, que é usado para construir uma representação em grafo das regiões presentes na imagem original. Nesse grafo, os vértices representam as regiões e arestas indicam possibilidades de transição. Os pesos assinalados às arestas são inversamente proporcionais às atravessabilidades das regiões adjacentes. Dessa maneira, otimizar uma rota e minimizar o risco de travessia entre duas regiões significa encontrar o caminho mais curto entre dois vértices do grafo. Para isso, diversos algoritmos de menor caminho podem ser utilizados. Após a definição da rota inicial, o caminho é submetido a um pós-processamento, que tem como principal objetivo a suavização do percurso. O cerne da proposta é a análise de atravessabilidade por FCN, tema exposto na seção 3.2, em que apresenta-se o modelo TFCN (do inglês, *traversability fully convolutional network*), sua metodologia de treinamento e validação. Por fim, a seção 3.3 contém uma explanação acerca da estratégia de cálculo de caminhos utilizada.

Figura 12 – Diagrama do processo de análise de atravessabilidade e planejamento



Fonte: O Autor (2021)

3.1 Recursos

A preparação de um modelo de aprendizado supervisionado requer um conjunto de dados de treinamento específico para o problema. As amostras desse conjunto são

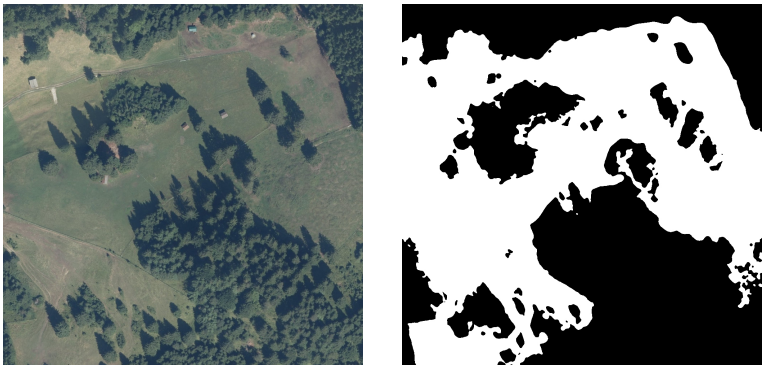
exemplos do tipo de informação que será apresentado ao modelo durante sua operação e da maneira como o modelo deve responder ao estímulo. O presente trabalho utiliza a base de dados denominada ATPD (do inglês, *aerial traversability and planning dataset*), usada no trabalho de Borges et al. (2019), especificamente organizada para análise de atravessabilidade e planejamento de caminhos. A validação cruzada também faz uso do mesmo *dataset*. Neste trabalho, foi empregado o método *leave-one-out*. Técnicas de aumento de dados foram aplicadas durante o treinamento com o propósito de apresentar ao modelo uma maior variedade de exemplos e fortificar seu aprendizado. Além disso, para avaliar quão bem o modelo lida com amostras de baixa frequência no *dataset*, foram realizados experimentos múltiplos, com e sem a presença de *outliers*. Detalhes acerca do *dataset*, método de validação cruzada, aumento de dados e do tratamento de *outliers* são expostos nesta seção.

3.1.1 Dataset

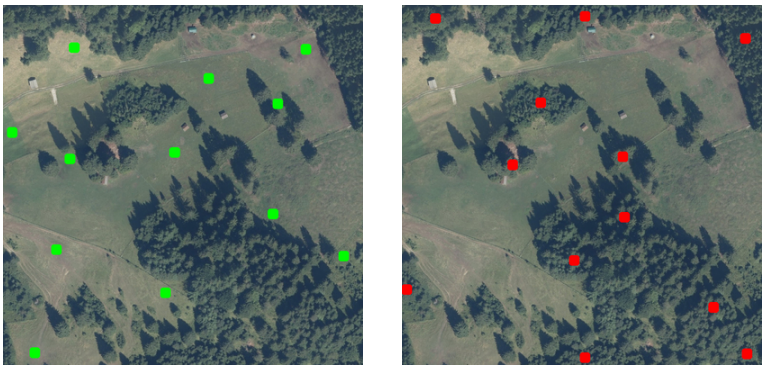
O conjunto de dados ATPD contém 8 imagens aéreas ortorretificadas de tamanho 1000×1000 *pixels*, com resolução espacial de aproximadamente $0.3 \times 0.3\ m^2/px$. As imagens aéreas para construção desse conjunto foram obtidas a partir do *Inria Image Labeling Dataset* (MAGGIORI et al., 2017) e do portal de dados aéreos da instituição americana *National Ecological Observatory Network* (NEON, 2013). Os rótulos de atravessabilidade e a marcação de pontos de interesse foram desenvolvidos por Borges et al. (2019). Cada imagem possui um rótulo de atravessabilidade correspondente, no qual são assinalados valores binários de atravessabilidade a todos os *pixels* da imagem original: 1, se o *pixel* original pertence a uma subregião atravessável e 0 caso contrário. Essas imagens e rótulos são usadas para treinamento, validação interna e teste. Uma amostra do ATPD é exibida na Figura 13. O *dataset* também inclui dois conjuntos de pontos de interesse para cada imagem. O primeiro conjunto contém 12 coordenadas alcançáveis e é usado para verificar se uma abordagem de planejamento de caminhos pode produzir trajetórias seguras. O segundo conjunto possui 12 coordenadas inalcançáveis e é usado para testar se a abordagem é capaz de detectar a impossibilidade do trajeto e rejeitar a proposta de missão.

O conjunto ATPD foi desenvolvido para treinamento e teste de modelos de regressão de atravessabilidade, técnica que tem por objetivo determinar diretamente o valor ou custo das travessias. No entanto, para experimentos de comparação com modelos que realizam classificação de terrenos, foi feita uma extensão para incluir rótulos de classes. Foram identificados 6 tipos principais de terreno ou objetos presentes nas imagens no *dataset*: estradas, construções, grama, vegetação, veículos e terreno arenoso. Para abranger os elementos não presentes nessas categorias, foi adicionada uma classe para objetos desconhecidos. Portanto, na extensão do ATPD para classificação, os rótulos estão distribuídos em 7 classes e são assinalados *pixel* a *pixel* (Figura 14). O conteúdo do ATPD é exposto no Apêndice A.

Figura 13 – Exemplo de amostra do conjunto de dados ATPD



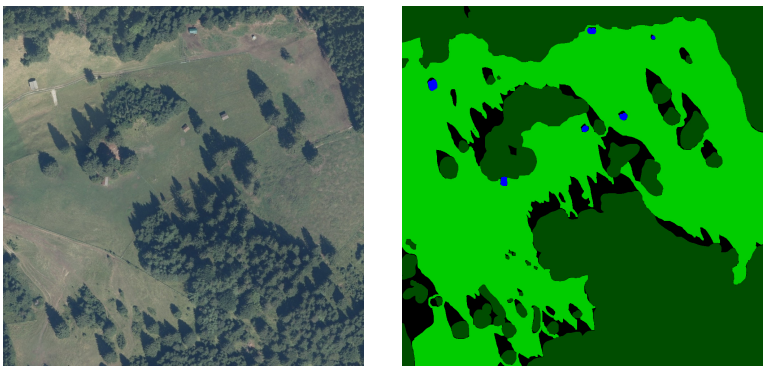
(a) Imagem aérea (b) Rótulos de atravessabilidade



(c) Coordenadas alcançáveis (d) Coordenadas inalcançáveis

Fonte: Imagens aéreas de Maggiori et al. (2017), com rótulos e marcações feitas por Borges et al. (2019)

Figura 14 – Exemplo de extensão do conjunto ATPD para problemas de classificação



(a) Imagem aérea (b) Rótulos de classificação

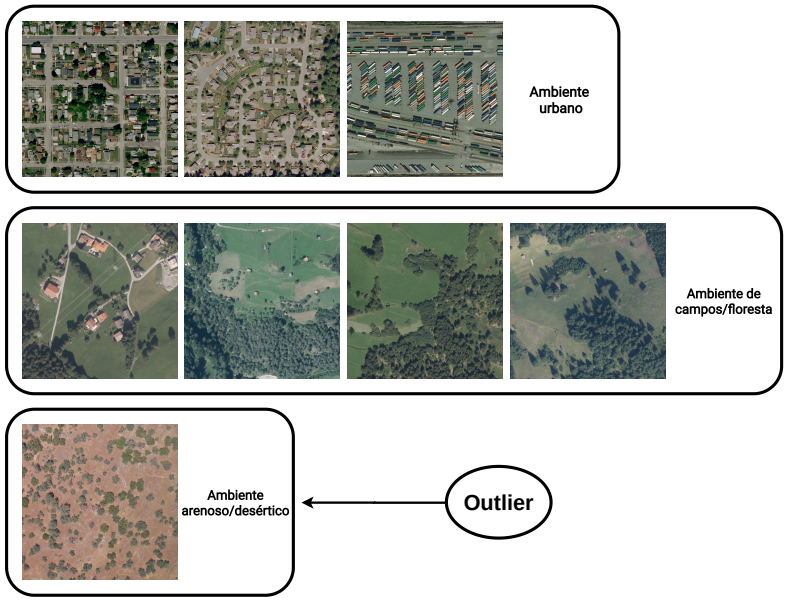
Fonte: Imagens aéreas de Maggiori et al. (2017), com rótulos feitos pelo Autor (2021)

3.1.2 Tratamento de outliers

As imagens aéreas presentes no ATPD são expostas na Figura 15, de acordo com o tipo de ambiente que representam. Pode-se observar que o ambiente arenoso/desértico contém diferenças significativas na aparência do terreno em relação às demais e possui apenas uma amostra. Além disso, essa imagem exibe um cenário arenoso com vegetação esparsa, no entanto, terrenos arenosos não estão presentes em nenhuma outra imagem

no ATPD. Levando em consideração que essa amostra representa um tipo único de terreno no *dataset*, pode ser considerado injusto usá-la como elemento de teste durante a validação cruzada, pois nenhuma das amostras de treino poderia preparar o modelo para prever atravessabilidades corretamente neste novo ambiente. Essa observação é ainda mais relevante se o modelo realiza predições usando apenas os canais RGB da imagem. Dessa maneira, decidiu-se realizar dois tipos de experimentos. No primeiro experimento, utilizou-se todo o *dataset*. No segundo experimento, a amostra de ambiente arenoso/desértico foi ignorada, sendo o treinamento, a validação interna e o teste realizados apenas com as amostras restantes. A análise dos resultados desses experimentos permite tirar conclusões acerca da capacidade de generalização do modelo e sua robustez à presença de *outliers*.

Figura 15 – Imagens do ATPD, organizadas segundo seus tipos de ambiente



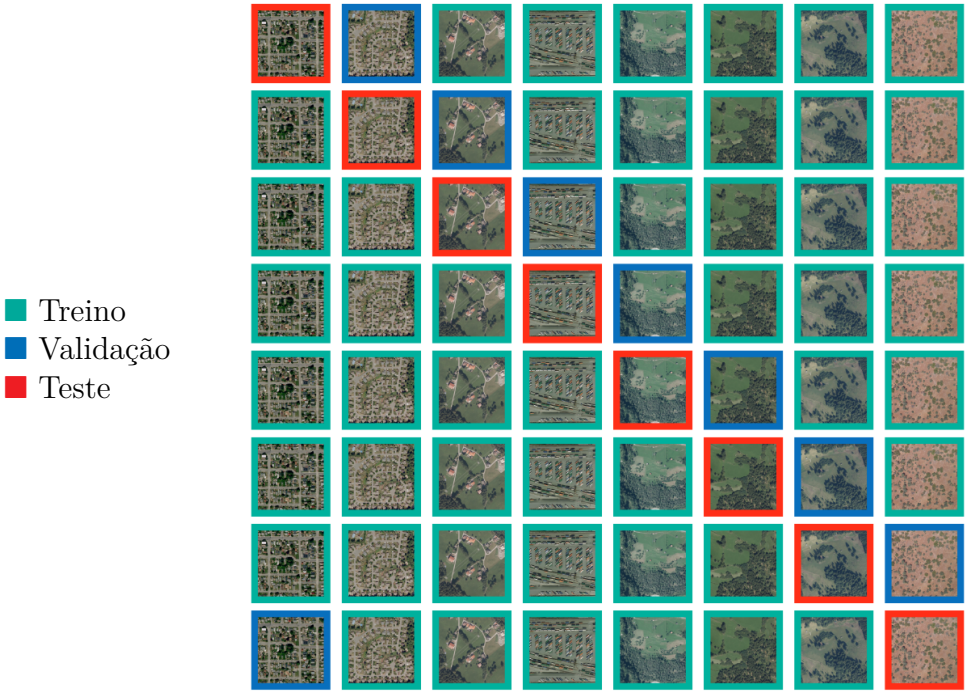
Fonte: Imagens aéreas de Maggiori et al. (2017) e NEON (2013), com categorização feita pelo Autor (2021)

3.1.3 Separação de dados para validação cruzada

A validação cruzada foi feita com o método *leave-one-out*. Nessa abordagem, cada amostra do conjunto ATPD foi usada uma vez para teste, enquanto as demais foram usadas para treinamento. O resultado final da validação cruzada foi a média dos desempenhos obtidos após a iteração completa pelos conjuntos de teste. A cada iteração, as amostras de treinamento foram ainda subdivididas em conjuntos de seis amostras para treino e uma amostra para validação interna. Nos experimentos sem a presença de *outliers*, foram usadas cinco amostras para treino e uma para validação interna. A Figura 16 contém uma representação visual dos conjuntos de treino, validação interna e teste utilizados em cada iteração do *leave-one-out* para os experimentos com o *dataset* completo. Na figura, cada

linha representa uma iteração e os conjuntos são indicados pelas cores nas bordas das amostras.

Figura 16 – Visualização dos conjuntos de treino, validação e teste



Fonte: Imagens aéreas de Maggiori et al. (2017) e NEON (2013), com categorização feita pelo Autor (2021)

É importante distinguir os conceitos de validação cruzada, conjunto de teste e conjunto de validação interna. A validação cruzada refere-se ao processo de avaliar a qualidade do modelo utilizando múltiplas iterações de treinamento e teste com diferentes conjuntos de dados. Os conjuntos de treino e validação interna, por sua vez, estão ligados ao processo de otimização dos parâmetros treináveis. O conjunto de treino propriamente dito contém as amostras que são apresentadas ao modelo e cujos valores de perda são usados diretamente para atualizar os parâmetros da arquitetura. Já o conjunto de validação interna é usado para estimar o desempenho da rede durante treinamento. Os valores de perda calculados sobre esse conjunto são necessários para a detecção de sobreajuste e realização da parada antecipada.

Considera-se que o método de validação cruzada escolhido é justo, pois garante que, em cada iteração, amostras do conjunto de teste não são usadas de maneira alguma durante o treinamento. Além disso, a escolha da melhor configuração de parâmetros, feita usando a técnica de parada antecipada, baseia-se apenas na perda sobre o conjunto de validação interna, que não possui interseção com os conjuntos de treino e teste.

3.1.4 Aumento dos dados de treinamento

Durante essa pesquisa, houve dificuldade em encontrar *datasets* que contivessem rótulos para treinar modelos de regressão de atravessabilidade em imagens aéreas. Devido a esse contratempo, o conjunto ATPD foi desenvolvido através de recortes de outros *datasets* de imagens aéreas e rotulação manual de atravessabilidades, um processo bastante laborioso. A versão inicial do conjunto, apresentada por Borges et al. (2019), incluía apenas oito imagens, seus rótulos de atravessabilidade e pontos de interesse. Na segunda versão, foram adicionados rótulos de sete classes para experimentos com métodos de classificação. Apesar disso, pode-se considerá-lo um *dataset* extremamente pequeno para os padrões usados em redes neurais convolucionais. Por esse motivo, foram usadas técnicas de aumento de dados para incrementar artificialmente a quantidade e variedade de amostras na fase de treinamento. Esse aumento foi realizado através de transformações geométricas e alterações no espaço de cores das amostras. Para detalhes acerca de técnicas de aumento de dados e sua abrangência no aprendizado de máquina, recomenda-se a leitura de Shorten e Khoshgoftaar (2019).

Um segundo tipo de aumento realizado sobre o ATPD foi a alteração do espaço de cores das imagens de entrada. É um fato amplamente aceito que a natureza e dimensionalidade do espaço de entrada pode alterar o desempenho de um modelo (DOMINGOS, 2012). Com isso em mente, a proposta foi validada em dois experimentos com diferentes espaços de cores: escala de cinza e RGB. O restante deste trabalho se refere ao modelo testado com entradas em escala de cinza como TFCN-G e ao modelo testado com entradas RGB como TFCN-RGB. Na Tabela 1, especificam-se as transformações realizadas neste trabalho, que foram aplicadas diretamente aos conjuntos de treino. O processo de treinamento é detalhado na subseção 3.2.1.

Tabela 1 – Transformações realizadas para aumento de dados

Transformação	**Parâmetros**
Rotação	0º a 360º
Deslocamento vertical	Entre 0 e 30%
Deslocamento horizontal	Entre 0 e 30%
Giros	Vertical e Horizontal
Cisalhamento	Entre 0 e 30%
Ampliação	Entre 0 e 20%
Preenchimento de bordas	Reflexão
Alteração no espaço de cores	Conversões para escala de cinza

3.1.5 Implementação

O método proposto e as demais técnicas utilizadas para comparação foram implementadas usando a linguagem Python 3. As arquiteturas de redes neurais e o processo de

aumento de dados foram desenvolvidos com auxílio dos pacotes Keras e Tensorflow. Outros algoritmos de aprendizado, como SVM, foram implementados com o pacote Scikit-learn. Os algoritmos e visualizações dos grafos foram feitos com NetworkX e Graph-tool. Outros pacotes importantes na implementação do projeto foram OpenCV, Scikit-image e Numpy.

3.2 Mapas de atravessabilidade

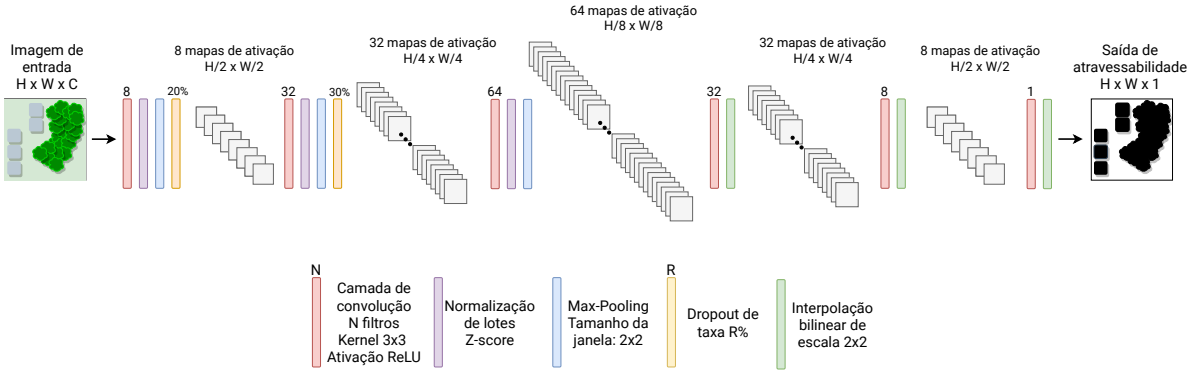
A proposta deste trabalho é o uso de uma arquitetura compacta e totalmente convolucional para análise de atravessabilidade, denominada TFCN. Esse modelo foi idealizado para computar valores de atravessabilidade diretamente a partir da imagem original, sem uso de mecanismos de pré-processamento, heurísticas ou classificação de terrenos para a estimativa de custos de travessia. Isso é diferente do que já foi realizado em trabalhos anteriores como Hudjakov e Tamre (2013) e Delmerico et al. (2017), que usaram CNNs e camadas totalmente conectadas para classificar regiões das imagens aéreas e só então assinalar atravessabilidades. No método aqui proposto, esses valores são calculados de forma direta. Também pode-se diferenciar o presente trabalho de outros que usaram FCNs no contexto de atravessabilidade para robôs, como Schilling et al. (2017), Hamandi, Asmar e Shammas (2018), Ono et al. (2018), Yang et al. (2019), Zhou et al. (2019), Iwashita et al. (2019) and Martinez-Soltero et al. (2020), pois esses métodos também dependem de classificação de terreno prévia. O uso de FCNs para regressão de funções de custo foi feito antes por Wulfmeier et al. (2017). O método aqui descrito, no entanto, diferencia-se do trabalho mencionado pelo cenário de aplicação e natureza dos dados de entrada. Enquanto a técnica descrita por Wulfmeier et al. (2017) utiliza informações geométricas capturadas por sensores LIDAR acoplados ao veículo, o presente trabalho segue uma abordagem visual para regredir atravessabilidades a partir de imagens capturadas por drones ou satélites. Esta seção explora a arquitetura da TFCN, os detalhes de seu treinamento e do método usado para avaliar as saídas de atravessabilidade em relação a outras técnicas.

3.2.1 Análise de atravessabilidade usando o modelo TFCN

Em alto nível, a arquitetura foi desenvolvida utilizando dois componentes: o codificador e o decodificador. O codificador realiza a extração de características da imagem de entrada e cria uma representação tensorial interna de seus atributos. O decodificador, por sua vez, processa a representação gerada pelo codificador e determina o grau de atravessabilidade de cada região da imagem original. Um diagrama do modelo é exibido na Figura 17.

O codificador da TFCN é composto por três fases de convolução e *pooling*. Normalização de lotes e regularização L2 com peso igual 10^{-3} são realizadas após cada camada convolucional com o propósito de acelerar o treinamento, reduzir a sensibilidade à inicialização de parâmetros e diminuir a complexidade do modelo. A técnica de *dropout* é usada

Figura 17 – Diagrama da TFCN



Fonte: O Autor (2021)

durante o treinamento ao final de cada fase desse componente e seu objetivo é reduzir o risco de sobreajuste. O propósito do codificador é extrair automaticamente atributos relevantes da imagem de entrada através da operação dos filtros de convolução, normalizações e *pooling*. A passagem pelas camadas desse componente reduz sucessivamente as dimensões espaciais de altura e largura da imagem e cria uma representação cada vez mais profunda. Pode-se observar no diagrama que, após a primeira fase do codificador, a representação possui dimensões $H/2 \times W/2 \times 8$, em que H e W são, respectivamente, a altura e a largura da imagem de entrada. A profundidade, por sua vez, é determinada pela quantidade de filtros usados na camada convolucional da fase em questão. Já o grau de compressão de altura e largura são especificados pelo tamanho da região de agregação usada na camada de *pooling*. Vê-se, então, que após a segunda fase do codificador, a representação assume o tamanho $H/4 \times W/4 \times 32$. E ao término da terceira fase, obtém as dimensões $H/8 \times W/8 \times 64$. Espera-se que a arquitetura composta por fases seja capaz de aprender características progressivamente mais complexas conforme os dados de entrada fluam para o seu interior, resultando em uma representação a partir da qual seja possível extrair informações de atravessabilidade no segundo componente do modelo.

O decodificador, inversamente, realiza três fases de convolução e interpolação. Cada camada de *pooling* no codificador possui uma contraparte de interpolação no decodificador, de tal maneira que a saída da TFCN possua as mesmas dimensões de altura e largura que a imagem de entrada. Efetivamente, o modelo assinala um valor de atravessabilidade a cada *pixel* da imagem original. Pode-se verificar esse comportamento através do diagrama da TFCN. Depois da primeira fase do decodificador, obtém-se uma representação de tamanho $H/4 \times W/4 \times 32$. Na segunda fase, o resultado tem dimensões $H/2 \times W/2 \times 8$. Por fim, após a última fase, tem-se o mapa de atravessabilidade, com dimensões $H \times W \times 1$. Os graus de interpolação aplicados no decodificador são correspondentes aos níveis de compressão impostos pelo codificador, ou seja, a cada fase da decodificação, duplicam-se

as dimensões de altura e largura. Simultaneamente, as quantidades de filtros nas camadas de convolução do decodificador foram definidas de maneira simétrica às do codificador. A exceção encontra-se na última camada, que possui apenas um filtro, de forma a garantir que a saída seja um único valor por *pixel* da imagem de entrada.

A TFCN foi treinada com o esquema de mini-lotes, otimização Adam (KINGMA; BA, 2015), perda MSE e parada antecipada (PRECHELT, 2012). O conjunto de treino propriamente dito foi constituído de amostras geradas através do aumento de dados do ATPD, como descrito na subseção 3.1.4. O modelo foi alimentado com 40 lotes de 16 amostras por época, sendo todos os lotes diferentes entre si. A parada antecipada foi aplicada nos casos em que não foi observado decréscimo de no mínimo 0,001 na perda de validação por 20 épocas consecutivas. Os parâmetros obtidos na época de menor perda de validação foram selecionados como resultado do treinamento.

3.2.2 Metodologia de validação dos mapas de atravessabilidade

A avaliação dos mapas de atravessabilidade foi feita separadamente para cada etapa do *leave-one-out*. Os mapas foram analisados visualmente e também através das perdas de treinamento, validação e teste a cada etapa em quatro experimentos, usando:

1. entradas RGB com inclusão de *outliers*;
2. entradas RGB com exclusão de *outliers*;
3. entradas em escala de cinza com inclusão de *outliers*;
4. entradas em escala de cinza com exclusão de *outliers*.

As saídas da TFCN também foram avaliadas em relação a outros trabalhos que tratam de atravessabilidade dentro da abordagem visual. Para realizar essa comparação, é necessário que seja possível obter uma medida de erro das saídas de atravessabilidade. Tal erro deve ser obtido por comparação da saída com seus valores esperados em um conjunto de teste. Essa comparação requer, também, que os métodos calculem atravessabilidades via regressão, assim como a TFCN. Neste trabalho, foram reproduzidos dois métodos que atendem a essa descrição. O primeiro método é uma versão adaptada de Guo et al. (2011), que usa regressão SVM para predizer a atravessabilidade de uma região da imagem usando como entrada um vetor de nove descritores de cor e textura. Em seu trabalho, os autores usam sensores inerciais para medir os ângulos de arfagem e rolamento do robô enquanto atravessa uma região do terreno. Esses valores são posteriormente usados para estimar os níveis de atravessabilidade. Na implementação desse experimento, no entanto, foram usados os níveis providos pelo ATPD. Essa modificação não altera as características do modelo SVM utilizado. O segundo método foi extraído de uma publicação anterior, resultante da pesquisa (BORGES et al., 2019), na qual as atravessabilidades são calculadas por uma heurística baseada no inverso do desvio padrão das intensidades dos *pixels* dentro da região (denominada aqui como heurística σ). Esse método possui determinados

parâmetros que devem ser selecionados de acordo com experimentação sobre o *dataset*. Nas comparações aqui apresentadas, foram usados os parâmetros $r = 8$ e $c = 0,4$. Esses parâmetros representam, respectivamente, a dimensão em *pixels* das regiões quadradas processadas pelo algoritmo e o limiar de corte de vértices do grafo de atravessabilidade. O valor de r foi escolhido de maneira a combinar com os tamanhos das regiões utilizadas no planejamento da TFCN, enquanto o valor de c foi reportado no artigo como o melhor limiar correspondente ao valor de r já selecionado, de acordo com experimentos no mesmo *dataset*. Os resultados obtidos são descritos na seção 4.1.

3.3 Caminhos

O procedimento de geração de caminhos pode ser dividido em duas etapas. A primeira etapa consiste na preparação de uma estrutura de dados para representar o mapa de atravessabilidades de uma maneira que seja possível delinear trajetórias. A estrutura escolhida para essa tarefa foi um grafo orientado e ponderado, cujos vértices representam regiões e arestas indicam possibilidades de transição. A segunda etapa é a busca pelo melhor caminho, que começa com a seleção dos pontos de partida e chegada, passa por um algoritmo de busca de caminhos e termina com um processo de suavização. O método utilizado aqui possui similaridades com o que foi já exposto em uma publicação anterior (BORGES et al., 2019). A principal alteração foi realizada na fase de corte de arestas, para a qual foi desenvolvida uma técnica de cálculo automático do limiar de corte. Esta seção trata da construção do grafo de atravessabilidade e demais definições e procedimentos de preparação e validação relativos ao cálculo dos caminhos. A principal mudança em relação ao artigo mencionado é detalhada na subseção 3.3.5.

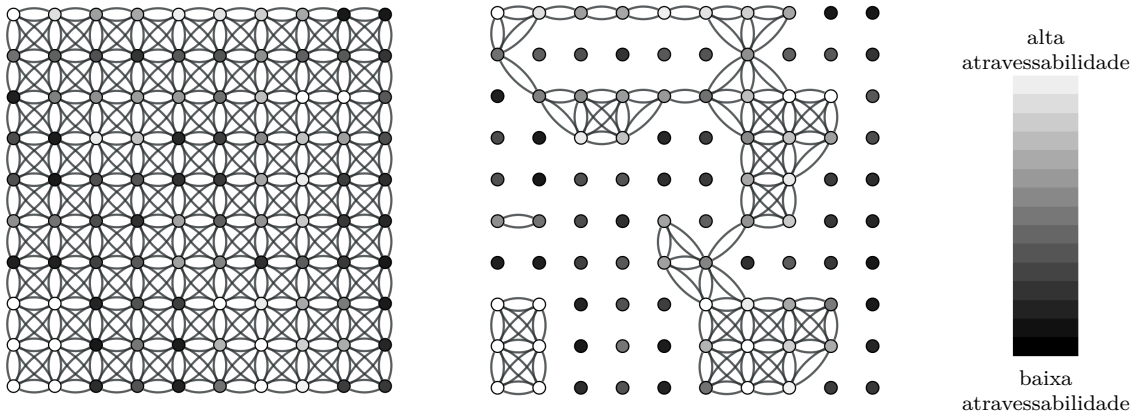
3.3.1 O grafo de atravessabilidade

A construção do grafo de atravessabilidade visa a facilitar a determinação de trajetórias com uso de algoritmos pré-existentes para busca por caminhos mais curtos. Escolheu-se representar o mapa como um grafo orientado e ponderado, pois essa estrutura permite a existência de caminhos de duas vias entre seus vértices e também possibilita estimar a qualidade de um caminho através dos pesos assinalados às suas arestas constituintes.

O primeiro passo do processo de construção do grafo de atravessabilidade é o pareamento entre regiões da imagem e vértices. Neste trabalho, o particionamento do mapa é feito por uma grade igualmente espaçada contendo células de tamanho $r \times r\ px$. Conhecendo-se a resolução espacial da imagem $r_s \times r_s\ m^2/px$, pode-se inferir que a área de uma célula ou região é igual a $r \cdot r_s \times r \cdot r_s\ m^2$. Portanto, modulando-se o valor de r, pode-se definir a abrangência métrica das regiões avaliadas. Nos experimentos, utilizou-se $r = 8$. No ATPD, o valor de r_s é aproximadamente $0,3$, portanto, cada vértice do grafo representa uma área de $2,4 \times 2,4\ m^2$.

No mapa, existem regiões intransponíveis, correspondentes a obstáculos ou demais áreas de baixa atravessabilidade. É importante que o método de cálculo de caminhos seja capaz de eliminar rotas que passem por essas regiões. Além disso, um problema relacionado é identificar quando não existe caminho possível entre duas regiões selecionadas. Para lidar com esses requerimentos, o grafo de atravessabilidade passa por uma fase de corte. Nessa fase, são removidas as arestas incidentes a vértices com atravessabilidade inferior a um limiar de corte c. Essa operação impede a produção de caminhos que contenham regiões consideradas ruins para a travessia, portanto, resolvendo os problemas mencionados anteriormente. Um exemplo dessa operação pode ser visualizado na Figura 18, em que são exibidas a configuração inicial de um grafo e sua estrutura após a fase de corte. Pode-se observar na configuração inicial que entre cada vértice adjacente existem duas arestas, representando ida e volta. A atravessabilidade de cada vértice é indicada pela sua coloração em escala de cinza. Na estrutura final do grafo, as arestas incidentes a vértices de baixa atravessabilidade foram removidas, o que impossibilita a passagem por essas regiões. A qualidade do corte e dos caminhos produzidos depende da escolha do limiar c. Neste trabalho, a seleção do limiar é realizada automaticamente por um processo de otimização especificado em detalhes na seção 3.3.5.

Figura 18 – Exemplo da operação de corte de arestas



(a) Estrutura inicial do grafo (b) Estrutura do grafo após fase de corte

Fonte: O Autor (2021)

Observa-se que, na fase de mapeamento, os valores de atravessabilidade são crescentes conforme aumenta a facilidade de travessia. No entanto, modelar o problema de forma a maximizar a atravessabilidade cria uma série de dificuldades. Por exemplo, maximizar a atravessabilidade e reduzir o comprimento da travessia seria equivalente a encontrar o caminho que resulta na maior soma dos pesos de suas arestas ao mesmo tempo em que se minimiza o número de vértices visitados. Embora inicialmente intuitiva, a resolução desse problema possui uma formulação mais complexa, pois exige a otimização de múltiplas variáveis. A abordagem escolhida foi transformar o problema em uma tradicional busca

por caminho mais curto. Com esse fim, os pesos das arestas foram determinados com base nas atravessabilidades de seus vértices adjacentes através da Equação 3.1, que foi proposta em Borges et al. (2019).

$$w(u,v) = \left(\frac{\kappa_u}{\tau_u}\right)^2 + \left(\frac{\kappa_v}{\tau_v}\right)^2 \tag{3.1}$$

Na expressão, $w(u,v)$ é o peso assinalado à aresta que liga u a v; τ_u e τ_v são, respectivamente as atravessabilidades de u e v; κ_u e κ_v são variáveis que permitem modular a contribuição de cada vértice para o valor final do custo de travessia de suas arestas. Neste trabalho, para simplificar a metodologia, utilizou-se $\kappa_u = \kappa_v = 100$, indicando que ambos os vértices contribuem sempre com a mesma intensidade para a atribuição dos pesos. No entanto, a definição dinâmica desses valores é uma estratégia a ser explorada. A característica mais importante dessa equação é sua capacidade de transformar os valores de atravessabilidade dos vértices em custos de travessia das arestas, sendo o custo de travessia inversamente proporcional às atravessabilidades. Dessa maneira, pode-se empregar algoritmos de caminho mais curto para a resolução do problema.

3.3.2 *Busca pelo melhor caminho*

O algoritmo A* (HART; NILSSON; RAPHAEL, 1968) foi usado neste trabalho para buscar pelo caminho de menor custo entre dois vértices s e t. O princípio de funcionamento do A* é similar ao clássico algoritmo de Dijkstra (1959), com a adição de uma heurística à função de custo que determina a ordem de exploração dos vértices. O algoritmo explora o grafo a partir da fonte s e determina um conjunto de predecessores e custos de alcance para cada vértice v da estrutura. O procedimento mantém uma fila de vértices a visitar e, a cada iteração, explora as imediações do elemento que possui o menor valor da função $f(s,t) = c(s,v) + h(v,t)$. A função $c(s,v)$ indica o custo do caminho mais curto encontrado até o momento entre s e v. A diferença implementada no A* é o uso de $h(v,t)$, a função heurística que calcula uma estimativa da distância entre v e o objetivo t. O término da execução ocorre quando encontra-se o vértice t ou esvazia-se a fila de elementos a visitar. Nesse momento, o custo do caminho pode ser obtido pelo valor da função $c(s,t)$ e o caminho de s a t pode ser reconstruído através de uma visitação recursiva pela estrutura de predecessores, iniciando-se a partir de t. Neste trabalho, utilizou-se a heurística da distância euclidiana, $h(v,t) = \sqrt{(x_v - x_t)^2 + (y_v - y_t)^2}$, em que (x_v, y_v) e (x_t, y_t) são, respectivamente, as coordenadas de v e t na matriz de atravessabilidade. Durante o desenvolvimento, não foram realizados experimentos com outras heurísticas. Apesar de sua praticidade e aplicabilidade a uma grande variedade de cenários, entende-se que a distância euclidiana pode não ser a heurística mais adequada ao problema. Um estudo acerca do desempenho de diferentes heurísticas ou até outros algoritmos de planejamento pode ser base de futura pesquisa.

3.3.3 Suavização do caminho

A busca com o algoritmo A* muitas vezes resulta em caminhos descontínuos que levariam o veículo robótico a realizar trajetórias sinuosas, gerando dispêndios extras de energia e possíveis desgastes físicos. É importante que essa particularidade do algoritmo de busca seja resolvida de maneira eficiente. Com esse propósito, foi utilizada uma técnica de planejamento local denominada suavização de caminho. O método aplicado consiste na subamostragem das coordenadas dos vértices do caminho utilizando o algoritmo de Ramer (1972), Douglas e Peucker (1973) e posterior reconstrução de uma representação da curva usando *B-splines* (DIERCKX, 1995). O processo de subamostragem reduz as componentes ruidosas do caminho, mas produz alterações bruscas do trajeto a cada região. A reconstrução com *B-splines*, por sua vez, elimina a maior parte dessas mudanças bruscas e fornece uma trajetória mais suave ao veículo. Um exemplo da suavização aqui aplicada pode ser visualizado na Figura 19.

Figura 19 – Exemplo de suavização de trajetória

(a) Trajetória produzida pelo algoritmo A*

(b) Trajetória suavizada com subamostragem e reconstrução por *B-splines*

Fonte: Imagens aéreas de Maggiori et al. (2017), com caminhos gerados pelo algoritmo de planejamento

3.3.4 Parâmetros de desempenho dos caminhos

É importante avaliar quão bem uma proposta de algoritmo é capaz de satisfazer as necessidades de planejamento do sistema. Com esse objetivo, foram utilizados quatro parâmetros para estimar o desempenho dos caminhos produzidos. Essas medidas foram originalmente descritas em (BORGES et al., 2019). Tais parâmetros são computados sobre um conjunto de propostas de trajeto denominadas missões. Uma missão é realizada sobre um mapa específico e inclui a posição inicial e o objetivo a ser alcançado. A missão é viável quando o objetivo é alcançável a partir da posição inicial; caso contrário, diz-se que a missão é inviável. A tarefa do método de planejamento é determinar a viabilidade da missão e, se possível, calcular um trajeto seguro entre os pontos selecionados. Dentro desse

contexto, é possível extrair quatro medidas, que poderão ser usadas tanto para otimização do método quanto para sua validação:

- Taxa de detecção de caminhos viáveis (f^+): é equivalente à fração de caminhos encontrados pelo método dentro do conjunto total de missões viáveis inferidas. Esse valor indica a capacidade do algoritmo de determinar a existência de um caminho entre dois pontos quando esse trajeto realmente existe.
- Taxa de detecção de caminhos inviáveis (f^-): equivale à fração de missões rejeitadas pelo método dentro do conjunto de missões inviáveis analisadas. Esse valor aponta quão bem o algoritmo detecta a inexistência de caminhos plausíveis quando realmente não é possível delinear um trajeto.
- Taxa de detecção de viabilidade (f): é a fração de missões corretamente categorizadas de acordo com sua viabilidade. Essa medida agrega f^+ e f^- e revela, de maneira geral, a capacidade do método para distinguir missões viáveis e inviáveis. A avaliação dessa variável é importante, pois a análise dos valores de f^+ e f^- individualmente pode levar a conclusões errôneas. Por exemplo, um algoritmo que aceita realizar todas as missões propostas teria um elevado f^+, mas seu f^- seria nulo. O valor de f captura o balanço entre f^+ e f^- em uma única medida.
- Qualidade do caminho (p): é uma estimativa de desempenho que mede a atravessabilidade geral do caminho, computada por uma agregação das atravessabilidades médias das regiões que o constituem. O intervalo de valores de p é $[0, 1]$, em que 0 indica um caminho que atravessa muitas regiões não atravessáveis e 1 sinaliza uma trajetória sem erros de atravessabilidade. A qualidade é calculada a partir dos rótulos disponíveis no *dataset*. Para um caminho $\mathcal{C}$ qualquer, formado pela sequência de coordenadas centrais $(c_1, c_2, c_3, ..., c_N)$, seja $\mathcal{R}^{\mathcal{L}} = (R_1, R_2, R_3, ..., R_N)$ a sequência de regiões do rótulo de atravessabilidades centradas em cada $c_i \in \mathcal{C}$, com área de $r \times r$ px. Define-se então uma estimativa das atravessabilidades de cada $R_i \in \mathcal{R}^{\mathcal{L}}$, representadas por $\mathcal{L} = (\lambda_1, \lambda_2, \lambda_3, ..., \lambda_N)$, em que λ_i é calculado pela Equação 3.2. Nessa expressão, $\ominus$ é o operador morfológico de erosão e S um elemento estruturante quadrado de tamanho $r/2 \times r/2$ px. A barra superior representa a operação de média aritmética sobre uma matriz. A ideia principal é agregar os valores de atravessabilidade dos *pixels* internos à região através da média. A erosão é aplicada de forma a incrementar a contribuição negativa de pequenos obstáculos.

$$\lambda_i = \overline{R_i \ominus S} \tag{3.2}$$

O valor p de qualidade do caminho é então determinado pela Equação 3.3, em que n é a quantidade de regiões do caminho com λ_i inferior a um limiar ℓ e δ é o fator de penalidade imposto para travessias sobre terrenos de baixa atravessabilidade. Em suma, a equação descreve o processo de percorrer as regiões do caminho e aplicar

uma penalidade δ a cada região cuja estimativa de atravessabilidade é menor que ℓ.

$$p = \max(0,\ 1 - \delta n(1 - \lambda_i)) \tag{3.3}$$

3.3.5 Estimativa do limiar de corte

No trabalho de Borges et al. (2019), o limiar de corte c era fixo e definido manualmente. Aqui, um método foi elaborado para calcular de forma automática o valor de c, com base nos dados de treinamento do modelo TFCN. O limiar é escolhido para maximizar a qualidade média dos trajetos (p) e as taxas de detecção de caminhos viáveis (f^+) e inviáveis (f^-). A otimização de c é importante, pois caso seu valor seja muito baixo, o algoritmo de planejamento poderá falhar no reconhecimento de regiões não alcançáveis, resultando em um baixo f^- e alto f^+. De maneira inversa, um valor de c muito elevado fará com o que planejador seja incapaz de encontrar caminhos atravessáveis, implicando em um baixo f^+ e alto f^-. A qualidade dos caminhos, nos dois casos, decresce de maneira considerável.

Fundamentando-se nessas observações acerca de p, f^+ e f^-, após o treinamento da TFCN, utilizou-se o método de Brent (1971) para estimar o valor de c que minimiza uma determinada função de perda $L(p, f^+, f^-)$. O método de Brent é uma heurística de minimização de rápida convergência que combina três outras técnicas de otimização: o método da bisseção, da secante e da interpolação inversa quadrática. A função de perda desenvolvida é especificada na Equação 3.4, em que α, β e γ representam a importância relativa que se deseja aplicar a cada um dos parâmetros de desempenho. Foram utilizados $\alpha = 2$, $\beta = 1$ e $\gamma = 3$. Isso significa que é dada prioridade à detecção e prevenção de passagens por regiões consideradas de baixa atravessabilidade.

$$L(p, f^+, f^-) = 1 - \frac{\alpha p + \beta f^+ + \gamma f^-}{\alpha + \beta + \gamma} \tag{3.4}$$

A cada iteração do processo, o método de planejamento usa uma estimativa de c para determinar viabilidade e trajetórias sobre 132 missões de travessia para cada imagem. Essas missões são predefinidas no conjunto ATPD. Os parâmetros de desempenho p, f^+ e f^- são calculados sobre as 132 missões e usados para encontrar a próxima estimativa de c usando o método de Brent, que prossegue com o propósito de minimização da função de perda até que seja atingida a convergência. Em testes exploratórios, foi possível melhorar p e f^+ sem alterações significativas em f^- através da divisão do valor de c encontrado no processo de otimização por dois. O limiar de corte determinado com esse método é então usado pelo algoritmo de planejamento para gerar caminhos, sempre utilizando o valor de c obtido sobre o conjunto de treinamento correspondente.

3.3.6 Metodologia de validação dos caminhos

O desempenho conjunto do mapeamento com a TFCN e do algoritmo de planejamento foi avaliado pelas métricas descritas na subseção 3.3.4 e comparado com uma série de métodos propostos na literatura. Foram selecionados métodos que usam a abordagem visual para análise de atravessabilidade, incluindo os artigos de Guo et al. (2011), Hudjakov e Tamre (2013), Christie et al. (2017) e Peterson et al. (2018). A técnica da heurística σ, desenvolvida em Borges et al. (2019), também está presente nas comparações. Nos experimentos, cada método foi usado para estimar atravessabilidades e calcular caminhos utilizando as informações disponíveis no ATPD. Todos foram submetidos ao mesmo tratamento de *outliers*, explanado na subseção 3.1.2, e processo de validação cruzada, exposto na subseção 3.1.3. As medidas de desempenho p, f^+, f^- e f foram compiladas separadamente para dois casos: primeiramente, usando o *dataset* completo e, em seguida, removendo-se os *outliers*. Também foram avaliadas métricas de tempo de processamento em relação às principais fases do processo de planejamento, que são o mapeamento de atravessabilidades e cálculo do caminho. Os resultados e discussões acerca desses experimentos estão disponíveis na seção 4.2

Deve-se acrescentar que não foi possível executar métodos de Hudjakov e Tamre (2013), Christie et al. (2017) e Peterson et al. (2018) no *dataset* completo. Isso deve-se ao fato de que essas técnicas baseiam-se em classificação de *pixels* ou regiões, cujas classes precisam estar presentes no conjunto de treinamento. No entanto, quando se realiza o experimento com o *dataset* completo, durante a etapa de teste no ambiente desértico, não há nenhuma outra imagem no conjunto de treinamento com a presença da classe de terreno arenoso. Dessa forma, o classificador seria incapaz de realizar qualquer predição para essa classe. Então, para manter a consistência dos experimentos, decidiu-se testar os métodos supracitados apenas no experimento com remoção de *outliers*. O restante desta subseção é dedicado a esclarecer os detalhes de implementação dos métodos de Hudjakov e Tamre (2013), Christie et al. (2017) e Peterson et al. (2018). As observações acerca dos demais métodos (GUO et al., 2011; BORGES et al., 2019) foram sumarizados na subseção 3.2.2.

No trabalho de Hudjakov e Tamre (2013), os autores treinam uma CNN para realizar classificação de terreno sobre regiões da imagem aérea. A saída da rede é um vetor que estabelece as probabilidades da presença de cada tipo de terreno. O custo de travessia assinalado a cada região é dado pelo produto escalar entre o vetor de probabilidades e um vetor de pesos que indica o custo associado a cada categoria. O mapa de custos é transformado em um grafo e então fornecido a um algoritmo de planejamento baseado em A*, processo descrito com mais detalhes em um de seus trabalhos anteriores (HUDJAKOV; TAMRE, 2011).

O método descrito por (CHRISTIE et al., 2017) também tem como base a classi-

ficação de terrenos. No entanto, tal categorização é realizada ao nível de *pixels*, através da segmentação semântica da imagem aérea. Para esta tarefa, os autores empregam uma biblioteca de segmentação denominada *automatic labeling environment* (ALE) (LADICKY, 2011). O resultado é corrigido através do uso de um modelo digital de elevação (DEM, do inglês *digital elevation model*), que é uma representação 3D da forma do terreno (FLORIANI; MAGILLO, 2018). Esse processo é usado como medida de segurança para detectar possíveis obstáculos não encontrados durante a segmentação. Nos experimentos do presente trabalho não há DEMs disponíveis, portanto, os testes aqui realizados tratam apenas da abordagem visual descrita pelos autores. A segmentação é usada para calcular o custo de travessia entre vértices do grafo, dando preferência a movimentações sobre estradas em vez de grama. Os autores também empregam o algoritmo A* para busca.

O procedimento de segmentação semântica usado por Peterson et al. (2018) é um simples classificador HSV com limiares fixos. Os autores, no entanto, reconhecem que a abordagem não é particularmente robusta e mencionam que tal classificador é apenas uma implementação temporária. Eles recomendam o uso de um método mais eficaz e citam que o processo descrito por Christie et al. (2017) é um exemplo que poderia ser empregado. Nos experimentos realizados aqui, o classificador HSV foi substituído pelo segmentador recomendado pelos autores. Após a categorização dos *pixels* da imagem aérea, os custos de travessia foram atribuídos às classes com base no dispêndio de energia de um veículo robótico ao atravessar o terreno correspondente. Os autores realizaram este experimento e assim determinaram os custos para cada classe. Neste trabalho, foram utilizados os mesmos custos descritos no artigo mencionado (PETERSON et al., 2018). Os autores mencionam o uso do algoritmo D* para o cálculo de caminhos. Tal algoritmo é uma extensão do A* para o caso de grafos dinâmicos. No entanto, devido à natureza estática dos grafos aqui abordados, não há diferença entre o uso das duas técnicas, pois o D* reduz-se ao A* em grafos estáticos. Deve-se mencionar que o artigo de Guo et al. (2011) utiliza uma técnica alternativa para executar os caminhos após o cálculo de atravessabilidades. Nos experimentos realizados aqui, no entanto, a técnica de estimativa de atravessabilidades de Guo et al. (2011) foi usada como suporte para o planejamento com o algoritmo A*. É importante ter em mente que o propósito deste experimento é fazer uma comparação entre as técnicas de cálculos de atravessabilidades e preparação da estrutura de grafos, não necessariamente do algoritmo de cálculo de caminhos. Devido a isso, todos os experimentos aqui descritos utilizam o A* como planejador de caminhos.

4 RESULTADOS

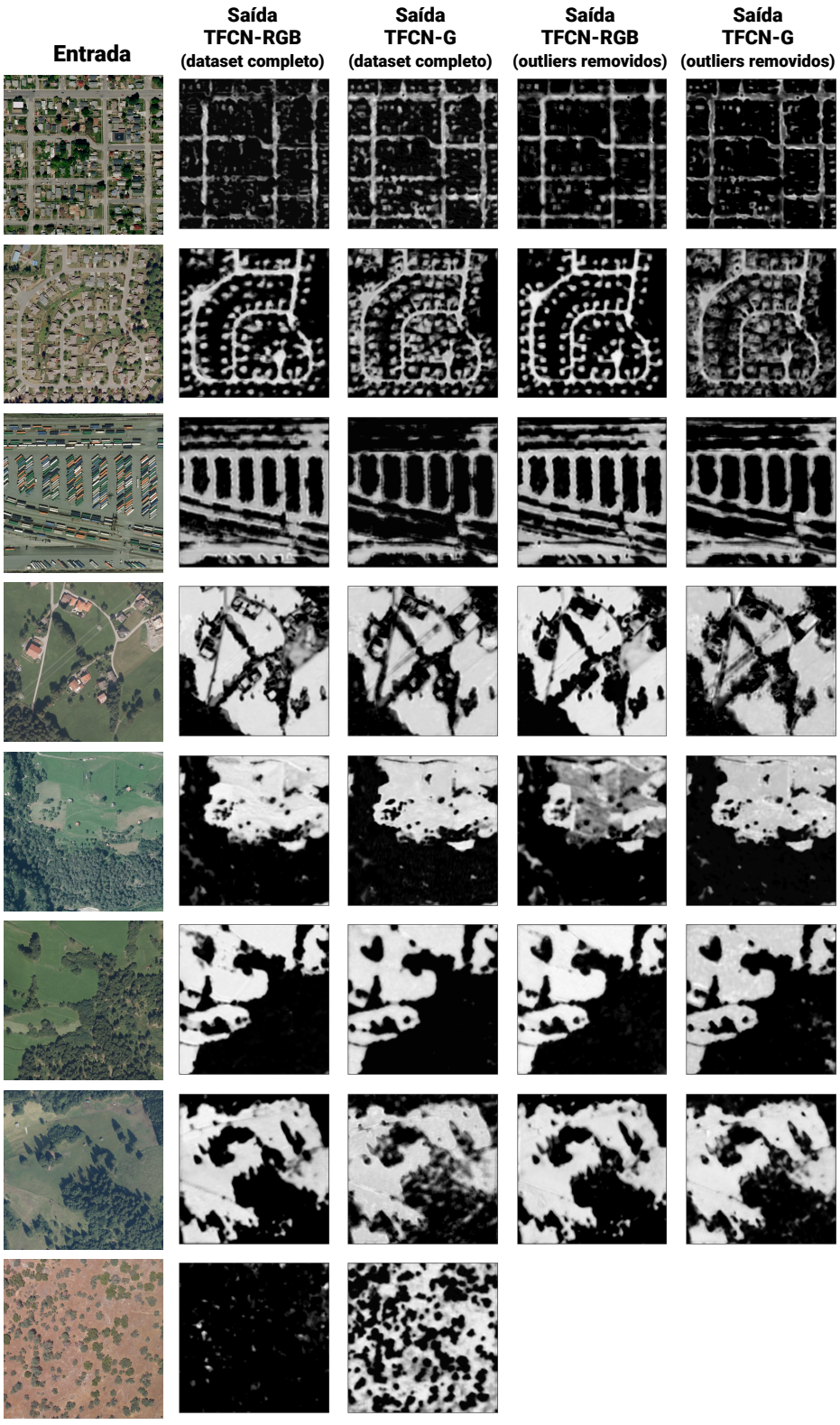
Neste capítulo são apresentados os resultados dos experimentos descritos nas subseções 3.2.2 e 3.3.6. Cada apresentação é acompanhada por discussões e comparações com outros métodos para análise de atravessabilidade visual encontrados na literatura. Os resultados são divididos em duas seções: mapas de atravessabilidade e planejamento de caminhos. Na primeira, o foco é o processo de mapeamento. Na segunda, trata-se dos caminhos planejados com uso dos mapas de atravessabilidade produzidos na primeira etapa do método, portanto, faz-se uma avaliação da metodologia como um todo e sua comparação com outras propostas.

4.1 Acerca dos mapas de atravessabilidade

Os mapas de atravessabilidade foram avaliados de duas maneiras. A primeira avaliação concentra-se no exame visual dos mapas obtidos e das perdas de treinamento, validação e teste dos modelos TFCN-RGB e TFCN-G em dois experimentos. Primeiro com o *dataset* completo e em seguida com *outliers* removidos. A segunda avaliação trata de uma comparação quantitativa com outros métodos de regressão de atravessabilidade.

Nas Tabelas 2, 3, 4 and 5, são apresentadas as perdas de treino, validação e teste obtidas com a melhor configuração de pesos em cada etapa do *leave-one-out*. Na Figura 20, são exibidas as entradas e saídas para cada amostra de teste. Gráficos que ilustram os comportamentos das perdas durante o treinamento, para todos os experimentos com a TFCN, podem ser encontrados no Apêndice B. Vê-se nas primeiras sete amostras que a TFCN foi capaz de reconhecer corretamente os níveis de atravessabilidade da maioria das regiões. Essa afirmação tem como suporte os baixos valores de erro de teste exibidos nas tabelas de perda. Apesar disso, ocorre uma exceção para o modelo TFCN-RGB no tratamento da última amostra, no qual pode-se observar que a maioria das áreas foi marcada como altamente não-atravessável, mesmo que algumas regiões do terreno arenoso possam ser transitadas. De maneira correspondente, a perda de teste dessa amostra é consideravelmente superior às demais (observar Tabela 2, Teste 8). Tal resultado não surpreende, pois o ambiente arenoso/desértico é uma amostra muito diferente das outras no quesito visual, especialmente pelos tipos de terreno e suas colorações. O experimento reforça, portanto, que uma forte mudança de coloração entre os conjuntos de treino e teste pode produzir um grande impacto negativo no desempenho do modelo RGB. Quando a mesma arquitetura é treinada com entradas em escala de cinza, os resultados não indicam tamanha sensibilidade, como pode ser visto na Figura 20 e nas tabelas de perda (observar Tabela 3, Teste 8). Isso é um indício de que trabalhar sem informações de cor e dar maior atenção a texturas em escala de cinza pode prover melhor capacidade de generalização para novos terrenos em *datasets* contendo *outliers*.

Figura 20 – Saídas da TFCN sobre as amostras do ATPD



Fonte: Imagens aéreas de Maggiori et al. (2017) e NEON (2013), com mapas gerados pela TFCN

Apesar da dificuldade de generalização para novos tipos de terreno, a arquitetura TFCN-RGB obteve desempenho médio superior à TFCN-G nas demais amostras, indicando

Tabela 2 – Validação cruzada do modelo TFCN-RGB (*dataset* completo)

Imagem Teste	**Melhor Época**	**Perda de Treino**	**Perda de Validação**	**Perda de Teste**
1	11	0,096	0,101	0,154
2	79	0,060	0,068	0,120
3	99	0,058	0,096	0,091
4	53	0,056	0,049	0,103
5	45	0,063	0,029	0,053
6	50	0,062	0,068	0,045
7	7	0,122	0,501	0,136
8	72	0,057	0,052	0,587

Tabela 3 – Validação cruzada do modelo TFCN-G (*dataset* completo)

Imagem Teste	**Melhor Época**	**Perda de Treino**	**Perda de Validação**	**Perda de Teste**
1	22	0,086	0,130	0,115
2	86	0,056	0,072	0,184
3	42	0,064	0,099	0,216
4	89	0,059	0,043	0,117
5	76	0,069	0,031	0,053
6	89	0,063	0,060	0,042
7	13	0,124	0,130	0,130
8	42	0,064	0,070	0,089

Tabela 4 – Validação cruzada do modelo TFCN-RGB (*outliers* removidos)

Imagem Teste	**Melhor Época**	**Perda de Treino**	**Perda de Validação**	**Perda de Teste**
1	28	0,062	0,069	0,105
2	26	0,066	0,072	0,103
3	17	0,067	0,086	0,090
4	95	0,047	0,047	0,092
5	54	0,057	0,030	0,087
6	22	0,075	0,079	0,050
7	35	0,057	0,061	0,080

Tabela 5 – Validação cruzada do modelo TFCN-G (*outliers* removidos)

Imagem Teste	**Melhor Época**	**Perda de Treino**	**Perda de Validação**	**Perda de Teste**
1	26	0,072	0,102	0,141
2	48	0,060	0,075	0,139
3	53	0,057	0,088	0,173
4	89	0,059	0,045	0,118
5	28	0,072	0,033	0,054
6	56	0,063	0,070	0,040
7	32	0,068	0,070	0,075

que se os padrões de coloração entre treino e teste são correspondentes, a informação de cor é ainda muito relevante para a qualidade da análise. Dessa maneira, o problema de generalização da TFCN-RGB pode ser contornado através de uma preparação adequada dos conjuntos de treino e teste, aliada a uma definição concreta sobre quais tipos de terreno o modelo estará apto a trabalhar. Essa asserção pode ser verificada na Tabela 6, em que observa-se o impacto positivo da remoção do *outlier* no desempenho da TFCN-RGB. Na tabela estão marcados em azul os melhores valores de MSE obtidos em cada experimento, calculados sobre o conjunto de teste. Vê-se que remover o *outlier* produz melhora da qualidade das saídas na maioria das amostras. Essa observação indica que o modelo RGB pode prover resultados superiores, mas é sensível a *outliers*. O mesmo não ocorre na TFCN-G, que não apresenta essa discrepância. Isso pode ser visto na Tabela 7, na qual os melhores valores de perda estão igualmente distribuídos entre os experimentos com e sem a presença de *outliers*.

Tabela 6 – Comparação do MSE em dois experimentos com a TFCN-RGB

Imagem Teste	**TFCN-RGB (*dataset* completo)**	**TFCN-RGB (*outliers* removidos)**
1	0,119	0,094
2	0,088	0,080
3	0,107	0,096
4	0,100	0,089
5	0,045	0,096
6	0,033	0,027
7	0,072	0,072

Tabela 7 – Comparação do MSE em dois experimentos com a TFCN-G

Imagem Teste	**TFCN-G (*dataset* completo)**	**TFCN-G (*outliers* removidos)**
1	0,087	0,116
2	0,113	0,087
3	0,236	0,192
4	0,121	0,127
5	0,051	0,055
6	0,042	0,042
7	0,086	0,067

Não obstante, a boa performance na atribuição de atravessabilidades para as primeiras sete amostras do ATPD, alguns problemas ainda podem ser observados. Por exemplo, no primeiro teste, algumas regiões de estrada foram marcadas como de difícil travessia, o que impede a busca por caminhos que passem por essas regiões. No segundo teste, telhados foram, por vezes, definidos como atravessáveis, possivelmente por suas cores e texturas parecerem muito similares a estradas. Os cabos de energia e trilhas do quarto teste foram marcados com baixas atravessabilidades, mas esses objetos não representam reais obstáculos para um veículo em terra. Essas pequenas falhas podem levar a uma queda

de desempenho no planejamento de caminhos e eventualmente conduzir o veículo robótico a uma trajetória perigosa ou subótima, como será mostrado na seção 4.2.

Tabela 8 – MSE dos mapas de atravessabilidade via regressão (*dataset* completo)

Imagem Teste	Guo et al. (2011) MSE	Borges et al. (2019) MSE	TFCN-RGB MSE	TFCN-G MSE
1	0,209	0,157	0,119	0,087
2	0,331	0,207	0,088	0,113
3	0,213	0,171	0,107	0,236
4	0,222	0,127	0,100	0,121
5	0,214	0,139	0,045	0,051
6	0,192	0,142	0,033	0,042
7	0,164	0,176	0,072	0,086
8	0,237	0,149	0,572	0,081

Tabela 9 – MSE do mapas de atravessabilidade via regressão (*outliers* removidos)

Imagem Teste	Guo et al. (2011) MSE	Borges et al. (2019) MSE	TFCN-RGB MSE	TFCN-G MSE
1	0,207	0,157	0,094	0,116
2	0,329	0,207	0,080	0,087
3	0,214	0,171	0,096	0,192
4	0,219	0,127	0,089	0,127
5	0,215	0,139	0,096	0,055
6	0,199	0,142	0,027	0,042
7	0,166	0,176	0,072	0,067

As Tabelas 8 e 9 contêm os valores de MSE calculados sobre as amostras de teste. Os mapas de atravessabilidade foram redimensionados para o tamanho padrão de 125×125 px, o que corresponde a regiões de área $2,4 \times 2,4$ m^2. As melhores medidas em cada caso são realçadas em azul. Pode-se ver pelas tabelas que em todos os casos a TFCN supera os demais métodos por uma margem significativa.

4.2 Acerca do planejamento de caminhos

O desempenho do sistema conjunto, unindo a TFCN para análise de atravessabilidade e o método de planejamento proposto, foi avaliado e comparado com os trabalhos de Guo et al. (2011), Hudjakov e Tamre (2013), Christie et al. (2017), Peterson et al. (2018) e Borges et al. (2019). Neste experimento, cada método foi usado para estimar atravessabilidades sobre as imagens do ATPD, seguindo a mesma abordagem de validação cruzada dos testes anteriores (subseções 3.1.3 and 3.1.2). Em seguida, sobre cada imagem de teste, foram executadas 132 missões, igualmente distribuídas entre viáveis e inviáveis. No total, para cada método, foram realizadas 1056 missões nos experimentos com *dataset* completo e 924 nos testes feitos com remoção de *outliers*. Os parâmetros de desempenho, como especificados na subseção 3.3.4, estão compilados nas Tabelas 10 e 11.

Vê-se na tabela 10 que a arquitetura TFCN-G supera as demais técnicas no caso em que o dataset completo é *utilizado*. A TFCN-G obteve maiores taxas de detecção de caminhos inviáveis, detecção de viabilidade e qualidade dos caminhos. A única exceção ocorreu na detecção de percursos viáveis, na qual foi superada pela heurística σ. Mesmo assim, a TFCN-G atinge a segunda posição nesse critério. No entanto, quando os *outliers* foram removidos, a TFCN-RGB demonstrou o melhor desempenho em todos os critérios, segundo a Tabela 11. O planejamento utilizando os mapas produzidos pela TFCN-RGB foi capaz de definir corretamente a possibilidade de 100% dos caminhos viáveis e sinalizou a impossibilidade de 99,8% dos trajetos inviáveis, o que levou a 99,9% de acurácia na determinação de viabilidade das rotas. A qualidade dos caminhos atingiu 0,968 ± 0,096, uma indicação de que a maioria dos caminhos são válidos e não apresentam desvio significativo do valor médio.

Tabela 10 – Validação com planejamento de caminhos (*dataset* completo)

Método de cálculo de atravessabilidades	**Parâmetros de desempenho**			
	f^+	f^-	f	p
Regressão SVM (GUO et al., 2011) + A*	0,860	0,714	0,787	0,610 (± 0,428)
Heurística σ (BORGES et al., 2019)	0,979	0,924	0,952	0,906 (± 0,239)
TFCN-RGB	0,854	0,841	0,848	0,809 (± 0,359)
TFCN-G	0,958	0,964	0,961	0,929 (± 0,219)

Tabela 11 – Validação com planejamento de caminhos (*outliers* removidos)

Método de cálculo de atravessabilidades	**Parâmetros de desempenho**			
	f^+	f^-	f	p
Regressão SVM (GUO et al., 2011) + A*	0,857	0,816	0,837	0,659 (± 0,423)
Classificação CNN (HUDJAKOV; TAMRE, 2013)	0,911	0,978	0,945	0,878 (± 0,280)
Segmentação ALE 1(CHRISTIE et al., 2017)	0,935	0,827	0,881	0,680 (± 0,406)
Segmentação ALE 2(PETERSON et al., 2018)	0,935	0,818	0,877	0,582 (± 0,440)
Heurística σ (BORGES et al., 2019)	0,976	0,935	0,956	0,912 (± 0,233)
TFCN-RGB	1,000	0,998	0,999	0,968 (± 0,096)
TFCN-G	0,976	0,991	0,984	0,963 (± 0,145)

O método de classificação de regiões por CNN funcionou bem, foi capaz de reconhecer viabilidade em 94,5% dos testes e produzir caminhos de boa qualidade na maioria dos casos. Contudo, as abordagens baseadas em segmentação com a biblioteca ALE tiveram desempenho ruim. Pela Tabela 11, vê-se que esses métodos, mesmo que tenham corretamente apontado a possibilidade de 93,5% dos caminhos viáveis, não foram capazes de consistentemente gerar caminhos seguros, fato deduzido a partir do baixo valor de p e elevado desvio padrão. Após análise dos mapeamentos, verificou-se que o segmentador ALE não foi capaz de classificar corretamente regiões nos ambientes urbanos do ATPD, o que levou a péssimos resultados nessas situações. Em contrapartida, nos ambientes de

campos/florestas, ALE teve bom desempenho e os caminhos produzidos alcançaram melhor qualidade. Deve-se lembrar que os experimentos realizados no trabalho de Christie et al. (2017) trataram de ambientes abertos contendo principalmente campos gramados e estradas, um cenário para o qual ALE demonstrou-se suficiente. Em ambientes mais complexos, no entanto, surge a necessidade de um segmentador ou regressor de atravessabilidades mais robusto.

Comparando-se ambas as Tabelas 10 e 11, vê-se que quase não há variação entre os *scores* obtidos pela heurística σ e TFCN-G nos dois diferentes experimentos. Isso é uma evidência de que esses métodos são robustos à presença de *outliers*, visto que a existência desse tipo de amostra não interfere de forma significativa em seus desempenhos. A regressão SVM e a TFCN-RGB, no entanto, não demonstraram tal robustez. Ambas sofreram quedas de desempenho quando o *dataset* completo foi usado e demonstraram grande melhora nos testes sem *outliers*. Especial atenção deve ser dada à TFCN-RGB, cujos parâmetros de desempenho superaram os demais métodos, em todos os quesitos, no experimento sem *outliers*. Isso significa que arquitetura RGB, com o devido cuidado para com o conjunto de treinamento e área de aplicação, pode ser um recurso eficaz no cálculo de atravessabilidades sobre imagens aéreas. Sugere-se pelas evidências que a principal preocupação no treinamento desse modelo é garantir que o conjunto de treino contenha amostras suficientes em ambientes similares às imagens de teste ou terreno de aplicação, em uma paleta de cores equivalente. Tal preocupação é bastante comum no ramo do aprendizado de máquina. Caso não seja possível garantir essas condições, a TFCN-G pode ser uma melhor escolha, visto que ela não depende de informações de cor para predizer atravessabilidades.

4.3 Acerca dos tempos de processamento

Os métodos também foram analisados de acordo com seus tempos de processamento. Os resultados são exibidos na Tabela 12, com valores em segundos. A primeira coluna mede o tempo médio necessário para processar a imagem de entrada, computar atravessabilidades ou classificar *pixels*/regiões e construir o grafo de busca. A segunda coluna é o tempo médio para planejar um caminho entre dois pontos do mapa. Pode-se observar que as TFCNs apresentam uma clara vantagem nesse critério, sendo capazes de mapear áreas de $300 \times 300\ m^2$ e resolução espacial de $0,3 \times 0,3\ m^2/px$ em menos de um segundo, o que é, no mínimo, quatro vezes mais rápido que os outros métodos. A razão para essa vantagem é o fato de que as TFCNs produzem todo o mapa de atravessabilidade em apenas uma passagem direta pela arquitetura. Os outros métodos apresentados precisam avaliar milhares de subregiões da imagem individualmente para realizar regressão ou classificação, fator que eleva o custo de processamento.

Tabela 12 – Comparação de tempo para mapeamento e planejamento *

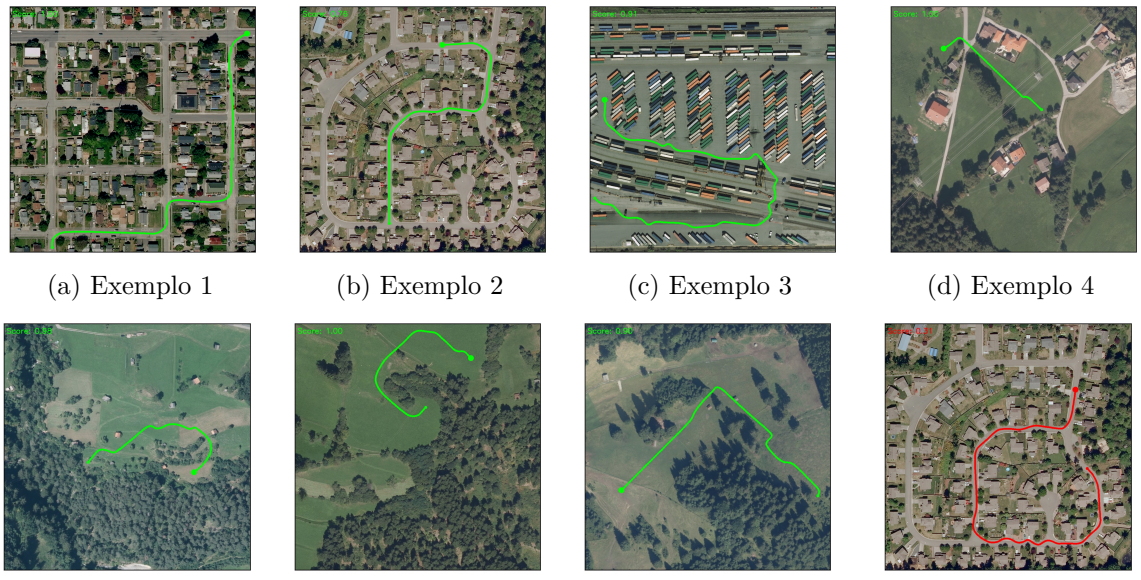
Método de cálculo de atravessabilidades	**Tempo (s)**	
	Mapeamento	**Planejamento**
Regressão SVM (GUO et al., 2011) + A*	43,151 (± 0,727)	0,032 (± 0,036)
Classificação SVM (HUDJAKOV; TAMRE, 2013)	8,678 (± 0,184)	0,022 (± 0,028)
Segmentação ALE 1(CHRISTIE et al., 2017)	123,666 (± 56,485)	0,099 (± 0,169)
Segmentação ALE 2(PETERSON et al., 2018)	36,256 (± 47,110)	0,044 (± 0,043)
Heurística σ (BORGES et al., 2019)	3,329 (± 0,806)	0,228 (± 0,279)
TFCN-RGB	0,716 (± 0,027)	0,022 (± 0,027)
TFCN-G	0,720 (± 0,066)	0,024 (± 0,030)

* Experimentos realizados com processador AMD Ryzen 5 2600X e memória RAM 16 GB DDR4 2666 Mhz.

4.4 Análise qualitativa dos caminhos

Na Figura 21, são exibidos diversos exemplos de caminhos calculados usando os mapas produzidos pela TFCN sobre o *dataset* sem *outliers*. Pode-se observar que a técnica é capaz de produzir trajetos de boa qualidade mesmo em ambientes cheios de obstáculos como cidades e locais de armazenamento, mas também em áreas rurais e ambientes florestais.

Figura 21 – Exemplos de planejamento nas amostras do ATPD



(a) Exemplo 1 (b) Exemplo 2 (c) Exemplo 3 (d) Exemplo 4

(e) Exemplo 5 (f) Exemplo 6 (g) Exemplo 7 (h) Exemplo 8

Fonte: Imagens aéreas de Maggiori et al. (2017), com caminhos gerados pelo algoritmo de planejamento

Não obstante, existem casos em que o método falha no cálculo do caminho. Um exemplo disso é mostrado na Subfigura 21h. Nesse caso, a TFCN-RGB reconhece a sombra de uma árvore como um obstáculo que parece bloquear a estrada e assinala a essa região um baixo valor de atravessabilidade. O algoritmo de planejamento recebe essa informação

e produz uma trajetória subótima, contornando todo o quarteirão para atingir o objetivo, que, na realidade, estava muito mais próximo. Esse exemplo realça um dos principais problemas de usar apenas imagens para computar atravessabilidades de uma perspectiva aérea: é difícil distinguir entre sombras ou outras superfícies incomuns e obstáculos. A presença de descontinuidades de elevação também é difícil de perceber sem informações tridimensionais. Idealmente, a técnica aqui apresentada deve ser usada para melhorar a fase de análise visual de atravessabilidade em métodos híbridos que combinem entradas visuais, geométricas e até proprioceptivas.

5 CONCLUSÃO

A presente dissertação explorou o uso de uma arquitetura totalmente convolucional para calcular atravessabilidades a partir de imagens aéreas. Esta tarefa é uma necessidade no campo da navegação robótica autônoma, visto que a mobilidade de um robô autônomo depende de sua capacidade de compreender quais terrenos são transitáveis e quais devem ser evitados. A arquitetura aqui exposta, denominada TFCN, foi treinada e avaliada usando o método *leave-one-out* sobre um pequeno conjunto de dados contendo áreas urbanas, rurais, florestais e desérticas. Os resultados obtidos mostram que a TFCN é capaz de gerar mapas de atravessabilidade que podem ser empregados com sucesso na fase de planejamento. Através desses mapas e do processo de construção do grafo de busca delineado, até mesmo o simples algoritmo A* pode ser usado para detectar a viabilidade de caminhos com alta acurácia e produzir bons percursos em ambientes variados.

Umas das principais vantagens do modelo apresentado é sua velocidade, pois o mapa de atravessabilidade pode ser gerado em uma única passagem direta pela rede. Apesar disso, verificou-se nos experimentos uma forte sensibilidade à presença de *outliers* quando o treinamento é executado com informações de cor, mas que esse problema pode ser mitigado com o uso de entradas em escala de cinza. Outra questão que requer cuidado durante o uso da arquitetura é a possibilidade de objetos serem erroneamente identificados como obstáculos (e.g. sombras, cabos de energia, etc.) e obstáculos tidos como caminhos transitáveis (e.g. telhados de casas, visto sua similaridade de textura com estradas). Esses dois problemas são comuns dentro da abordagem visual, considerando-se que não há informações geométricas acerca do ambiente. Acredita-se que a integração dessa nova modalidade de dados poderia ser de grande auxílio na desambiguação de determinados terrenos. Sendo assim, recomenda-se o uso da presente técnica em métodos híbridos para análise de atravessabilidade, que podem vir a tratar também da abordagem geométrica e proprioceptiva. O futuro dessa pesquisa concentra-se na integração dessas abordagens à arquitetura aqui proposta para proposição de um método híbrido, capaz de lidar com ambientes mais complexos.

REFERÊNCIAS

AN, S. **Feedforward Neural Networks**. 2017. Disponível em: <https://www.cc.gatech.edu/~san37/post/dlhc-fnn/>. Acesso em: 03 set 2020. Citado na página 31.

BERGLUND, T.; BRODNIK, A.; JONSSON, H.; STAFFANSON, M.; SODERKVIST, I. Planning Smooth and Obstacle-Avoiding B-Spline Paths for Autonomous Mining Vehicles. **IEEE Transactions on Automation Science and Engineering**, v. 7, n. 1, p. 167–172, 2010. Disponível em: <https://doi.org/10.1109/TASE.2009.2015886>. Citado na página 21.

BERMUDEZ, F. L. G.; JULIAN, R. C.; HALDANE, D. W.; ABBEEL, P.; FEARING, R. S. Performance analysis and terrain classification for a legged robot over rough terrain: **IEEE/RSJ International Conference on Intelligent Robots and Systems**. [s.n.], 2012. p. 513–519. Disponível em: <https://doi.org/10.1109/IROS.2012.6386243>. Citado na página 22.

BERNUY, F.; SOLAR, J. R. D. Semantic Mapping of Large-Scale Outdoor Scenes for Autonomous Off-Road Driving: **IEEE International Conference on Computer Vision Workshop (ICCVW)**. [s.n.], 2015. p. 124–130. Disponível em: <https://doi.org/10.1109/ICCVW.2015.26>. Citado na página 18.

BIJO, S. **Traversability Estimation Techniques for Improved Navigation of Tracked Mobile Robots** . Tese (Doutorado) — Virginia Tech, 2019. Disponível em: <https://vtechworks.lib.vt.edu/handle/10919/94629>. Citado na página 22.

BJORCK, J.; GOMES, C. P.; SELMAN, B. Understanding Batch Normalization. **arXiv**, 2018. Disponível em: <http://arxiv.org/abs/1806.02375>. Citado na página 30.

BORGES, C. D. B.; ALMEIDA, A. M. A.; JÚNIOR, I. C. P.; JUNIOR, J. J. M. S. A strategy and evaluation method for ground global path planning based on aerial images. **Expert Systems with Applications**, v. 137, p. 232 – 252, 2019. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2019.06.067>. Citado 14 vezes nas páginas 16, 23, 38, 39, 42, 45, 46, 48, 49, 51, 52, 58, 59 e 61.

BRENT, R. P. An algorithm with guaranteed convergence for finding a zero of a function. **The Computer Journal**, v. 14, n. 4, p. 422–425, 1971. Disponível em: <https://doi.org/10.1093/comjnl/14.4.422>. Citado na página 51.

BRESSON, G.; ALSAYED, Z.; YU, L.; GLASER, S. Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving. **IEEE Transactions on Intelligent Vehicles**, v. 2, n. 3, p. 194–220, 2017. Disponível em: <https://doi.org/10.1109/TIV.2017.2749181>. Citado na página 19.

BUSTOS, P.; MANSO, L.; BANDERA, A.; BANDERA, J.; GARCÍA-VAREA, I.; MARTÍNEZ-GÓMEZ, J. The CORTEX cognitive robotics architecture: Use cases. **Cognitive Systems Research**, v. 55, p. 107 – 123, 2019. ISSN 1389-0417. Disponível em: <https://doi.org/10.1016/j.cogsys.2019.01.003>. Citado na página 20.

CASTEJÓN, C.; BOADA, B. L.; BLANCO, D.; MORENO, L. Traversable region modeling for outdoor navigation. **Journal of Intelligent and Robotic Systems**, v. 43, n. 2, p. 175–216, Aug 2005. ISSN 1573-0409. Citado na página 14.

CHAVEZ-GARCIA, R. O.; GUZZI, J.; GAMBARDELLA, L. M.; GIUSTI, A. Learning ground traversability from simulations. **IEEE Robotics and Automation Letters**, Institute of Electrical and Electronics Engineers (IEEE), v. 3, n. 3, p. 1695–1702, jul 2018. Disponível em: <https://doi.org/10.1109/lra.2018.2801794>. Citado na página 14.

CHELLAPILLA, K.; PURI, S.; SIMARD, P. High Performance Convolutional Neural Networks for Document Processing: Université de Rennes 1. **International Workshop on Frontiers in Handwriting Recognition**. 2006. Disponível em: <https://hal.inria.fr/inria-00112631>. Citado na página 24.

CHRISTIE, G.; SHOEMAKER, A.; KOCHERSBERGER, K.; TOKEKAR, P.; MCLEAN, L.; LEONESSA, A. Radiation search operations using scene understanding with autonomous UAV and UGV. **Journal of Field Robotics**, v. 34, n. 8, p. 1450–1468, 2017. Disponível em: <https://doi.org/10.1002/rob.21723>. Citado 9 vezes nas páginas 14, 15, 23, 52, 53, 58, 59, 60 e 61.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of Control, Signals and Systems**, v. 2, n. 4, p. 303–314, 1989. Disponível em: <https://doi.org/10.1007/BF02551274>. Citado 2 vezes nas páginas 28 e 30.

DANIEL, K.; NASH, A.; KOENIG, S.; FELNER, A. Theta*: Any-Angle Path Planning on Grids. **Journal of Artificial Intelligence Research**, v. 39, 2014. Disponível em: <https://doi.org/10.1613/jair.2994>. Citado na página 20.

DEEP LEARNING AI. **Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization**. 2016. Disponível em: <https://www.coursera.org/learn/deep-neural-network/>. Acesso em: 06 set 2020. Citado na página 35.

DELMERICO, J.; MUEGGLER, E.; NITSH, J.; SCARAMUZZA, D. Active Autonomous Aerial Exploration for Ground Robot Path Planning. **IEEE Robotics and Automation Letters**, v. 2, n. 2, p. 664–671, 2017. Disponível em: <https://doi.org/10.1109/LRA.2017.2651163>. Citado 4 vezes nas páginas 14, 15, 23 e 43.

DENKER, J.; GARDNER, W. R.; GRAF, H.; HENDERSON, D.; HOWARD, R.; HUBBARD, W.; JACKEL, L.; BAIRD, H.; GUYON, I. Neural Network Recognizer for Hand-Written Zip Code Digits: **Conference on Neural Information Processing Systems (NeurIPS)**. [s.n.], 1988. p. 323–331. Disponível em: <http://papers.nips.cc/paper/107-neural-network-recognizer-for-hand-written-zip-code-digits.pdf>. Citado na página 24.

DHILLON, A.; VERMA, G. Convolutional neural network: a review of models, methodologies and applications to object detection. **Progress in Artificial Intelligence**, v. 9, p. 85–112, 2019. Disponível em: <https://doi.org/10.1007/s13748-019-00203-0>. Citado 3 vezes nas páginas 24, 28 e 31.

DIERCKX, P. **Curve and Surface Fitting with Splines** . Clarendon Press, 1995. (Monographs on numerical analysis). ISBN 9780198534402. Disponível em: <https://books.google.com.br/books?id=-RIQ3SR0sZMC>. Citado na página 49.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische Mathematik**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 1, n. 1, p. 269–271, dez. 1959. Disponível em: <http://dx.doi.org/10.1007/BF01386390>. Citado 2 vezes nas páginas 20 e 48.

DOMINGOS, P. A few useful things to know about machine learning. **Communications of the ACM**, v. 55, n. 10, p. 78–87, out. 2012. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/2347736.2347755>. Citado na página 42.

DOUGLAS, D. H.; PEUCKER, T. K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. **Cartographica: The International Journal for Geographic Information and Geovisualization**, v. 10, n. 2, p. 112–122, 1973. Disponível em: <https://doi.org/10.3138/FM57-6770-U75U-7727>. Citado na página 49.

DROUILLY, R.; RIVES, P.; MORISSET, B. Hybrid metric-topological-semantic mapping in dynamic environments: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [s.n.], 2015. p. 5109–5114. Disponível em: <https://core.ac.uk/display/48162068>. Citado na página 18.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. **Journal of Machine Learning Research**, v. 12, n. 61, p. 2121–2159, 2011. Disponível em: <http://jmlr.org/papers/v12/duchi11a.html>. Citado na página 35.

FEDORENKO, R.; GABDULLIN, A.; FEDORENKO, A. Global UGV path planning on point cloud maps created by UAV: **IEEE International Conference on Intelligent Transportation Engineering (ICITE)**. IEEE, 2018. Disponível em: <https://doi.org/10.1109/icite.2018.8492584>. Citado na página 14.

FERGUSON, D.; STENTZ, A. Field D*: An Interpolation-Based Path Planner and Replanner: THRUN, S.; BROOKS, R.; DURRANT-WHYTE, H. (Ed.). **Robotics Research**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 239–253. ISBN 978-3-540-48113-3. Citado na página 20.

FLORIANI, L. D.; MAGILLO, P. Digital elevation models: _____. **Encyclopedia of Database Systems**. New York, NY: Springer New York, 2018. p. 1078–1083. ISBN 978-1-4614-8265-9. Citado 2 vezes nas páginas 23 e 53.

FLORINSKY, I. V. Chapter 3 - Digital Elevation Models: FLORINSKY, I. V. (Ed.). **Digital Terrain Analysis in Soil Science and Geology (Second Edition)**. Second edition. Academic Press, 2016. p. 77 – 108. ISBN 978-0-12-804632-6. Disponível em: <https://doi.org/10.1016/B978-0-12-804632-6.00003-1>. Citado na página 23.

FOROUTAN, M. **Traversability analysis in unstructured forested terrains for off-road autonomy using LIDAR data** . Tese (Doutorado) — Mississippi State University, 2020. Disponível em: <https://ir.library.msstate.edu/handle/11668/18478>. Citado na página 23.

FRAICHARD, T.; SCHEUER, A. From reeds and shepp's to continuous-curvature paths. **IEEE Transactions on Robotics**, v. 20, n. 6, p. 1025–1035, 2004. Disponível em: <https://doi.org/10.1109/TRO.2004.833789>. Citado na página 21.

FREECODECAMP. **An intuitive guide to Convolutional Neural Networks** . 2018. Largura: 678 pixels. Altura: 462 pixels. 149.5 kB. Formato: PNG. Disponível em: <https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/>. Acesso em: 30 ago 2020. Citado na página 25.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. **Biological Cybernetics**, p. 193–202, 04 1980. Disponível em: <https://doi.org/10.1007/BF00344251>. Citado na página 24.

GENÇAY, R.; QI, M. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. **IEEE Transactions on Neural Networks**, v. 12, p. 726 – 734, 08 2001. Citado na página 36.

GERAERTS, R.; OVERMARS, M. H. A comparative study of probabilistic roadmap planners: _____. **Algorithmic Foundations of Robotics V**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 43–57. ISBN 978-3-540-45058-0. Disponível em: <https://doi.org/10.1007/978-3-540-45058-0_4>. Citado na página 21.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. **Proceedings of the 14th International Conference on Artificial Intelligence and Statisitics (AISTATS)**, v. 15, p. 315–323, 01 2011. Citado na página 29.

GONZÁLEZ, D.; PÉREZ, J.; MILANÉS, V.; NASHASHIBI, F. A review of motion planning techniques for automated vehicles. **IEEE Transactions on Intelligent Transportation Systems**, v. 17, n. 4, p. 1135–1145, 2016. Disponível em: <https://ieeexplore.ieee.org/document/7339478>. Citado 2 vezes nas páginas 20 e 21.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning** . [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 34.

GUASTELLA., D. C.; CANTELLI., L.; MELITA., C. D.; MUSCATO., G. A global path planning strategy for a ugv from aerial elevation maps for disaster response: INSTICC. **Proceedings of the 9th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART**. SciTePress, 2017. p. 335–342. ISBN 978-989-758-219-6. Disponível em: <https://doi.org/10.5220/0006298303350342>. Citado na página 23.

GUL, F.; RAHIMAN, W.; ALHADY, S. S. N. A comprehensive study for robot navigation techniques. **Cogent Engineering**, Cogent OA, v. 6, n. 1, p. 1632046, 2019. Disponível em: <https://www.tandfonline.com/doi/abs/10.1080/23311916.2019.1632046>. Citado na página 17.

GUO, Y.; LIU, Y.; GEORGIOU, T.; LEW, M. S. A review of semantic segmentation using deep neural networks. **International Journal of Multimedia Information Retrieval**, v. 7, n. 2, p. 87–93, Jun 2018. ISSN 2192-662X. Disponível em: <https://doi.org/10.1007/s13735-017-0141-z>. Citado na página 32.

GUO, Y.; SONG, A.; BAO, J.; ZHANG, H. Optimal path planning in field based on traversability prediction for mobile robot: **International Conference on Electric Information and Control Engineering**. [s.n.], 2011. p. 563–566. Disponível em: <https://ieeexplore.ieee.org/document/5777948>. Citado 9 vezes nas páginas 14, 22, 23, 45, 52, 53, 58, 59 e 61.

GUO, Y.; SONG, A.; CAO, Y.; TANG, H. Research on navigation for search and rescue robot based on traversability: XIONG, C.; HUANG, Y.; XIONG, Y.; LIU, H. (Ed.). **Intelligent Robotics and Applications**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 853–862. ISBN 978-3-540-88513-9. Citado na página 14.

HAMANDI, M.; ASMAR, D.; SHAMMAS, E. Ground segmentation and free space estimation in off-road terrain. **Pattern Recognition Letters**, v. 108, p. 1 – 7, 2018. ISSN 0167-8655. Citado 2 vezes nas páginas 15 e 43.

HAN, L.; YASHIRO, H.; NEJAD, H. T. N.; DO, Q. H.; MITA, S. Bézier curve based path planning for autonomous vehicle in urban environment: **IEEE Intelligent Vehicles Symposium**. [s.n.], 2010. p. 1036–1042. Disponível em: <https://ieeexplore.ieee.org/document/5548085>. Citado na página 21.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. **IEEE Transactions on Systems Science and Cybernetics**, v. 4, n. 2, p. 100–107, July 1968. ISSN 0536-1567. Disponível em: <https://ieeexplore.ieee.org/document/4082128>. Citado 2 vezes nas páginas 20 e 48.

HINTON, G. E.; SRIVASTAVA, N.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Improving neural networks by preventing co-adaptation of feature detectors. **arXiv**, abs/1207.0580, 2012. Disponível em: <http://arxiv.org/abs/1207.0580>. Citado na página 30.

HORST, J.; BARBERA, A. Trajectory generation for an on-road autonomous vehicle: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Unmanned Systems Technology VIII**. SPIE, 2006. v. 6230, p. 866 – 877. Disponível em: <https://doi.org/10.1117/12.663643>. Citado na página 21.

HOWARD, A.; SERAJI, H.; WERGER, B. A terrain-based path planning method for mobile robots. **Seventh International Conference on Automation Technology, NASA Jet Propulsion Laboratory**, NASA Jet Propulsion Laboratory, 2003. Citado na página 14.

HUANG, S.; DISSANAYAKE, G. Robot Localization: An Introduction: _____. **Wiley Encyclopedia of Electrical and Electronics Engineering**. [s.n.], 2016. p. 1–10. ISBN 9780471346081. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W8318>. Citado na página 19.

HUDJAKOV, R.; TAMRE, M. Ortophoto analysis for UGV long-range autonomous navigation. **Estonian Journal of Engineering**, v. 17, p. 17–27, 01 2011. Disponível em: <https://doi.org/10.3176/eng.2011.1.03>. Citado na página 52.

HUDJAKOV, R.; TAMRE, M. Orthophoto Classification for UGV Path Planning using Heterogeneous Computing. **International Journal of Advanced Robotic Systems**, SAGE Publications, v. 10, n. 6, p. 268, jan 2013. Citado 8 vezes nas páginas 14, 15, 23, 43, 52, 58, 59 e 61.

IAGNEMMA, K.; SHIBLY, H.; DUBOWSKY, S. On-line terrain parameter estimation for planetary rovers: **Proceedings 2002 IEEE International Conference on Robotics and Automation**. [s.n.], 2002. v. 3, p. 3142–3147 vol.3. Disponível em: <https://ieeexplore.ieee.org/document/1013710>. Citado na página 22.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift: **Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37**. [S.l.]: JMLR.org, 2015. (ICML'15), p. 448–456. Citado 2 vezes nas páginas 29 e 30.

IWASHITA, Y.; NAKASHIMA, K.; STOICA, A.; KURAZUME, R. TU-Net and TDeepLab: Deep Learning-Based Terrain Classification Robust to Illumination Changes, Combining Visible and Thermal Imagery: **IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)**. [S.l.: s.n.], 2019. p. 280–285. Citado 2 vezes nas páginas 15 e 43.

JADON, S. A survey of loss functions for semantic segmentation. **arXiv**, 2020. Disponível em: <https://arxiv.org/abs/2006.14822>. Citado na página 33.

JOUPPI, N. P.; YOUNG, C.; PATIL, N.; PATTERSON, D.; AGRAWAL, G.; BAJWA, R.; BATES, S.; BHATIA, S.; BODEN, N.; BORCHERS, A.; BOYLE, R.; CANTIN, P. luc; CHAO, C.; CLARK, C.; CORIELL, J.; DALEY, M.; DAU, M.; DEAN, J.; GELB, B.; GHAEMMAGHAMI, T. V.; GOTTIPATI, R.; GULLAND, W.; HAGMANN, R.; HO, C. R.; HOGBERG, D.; HU, J.; HUNDT, R.; HURT, D.; IBARZ, J.; JAFFEY, A.; JAWORSKI, A.; KAPLAN, A.; KHAITAN, H.; KOCH, A.; KUMAR, N.; LACY, S.; LAUDON, J.; LAW, J.; LE, D.; LEARY, C.; LIU, Z.; LUCKE, K.; LUNDIN, A.; MACKEAN, G.; MAGGIORE, A.; MAHONY, M.; MILLER, K.; NAGARAJAN, R.; NARAYANASWAMI, R.; NI, R.; NIX, K.; NORRIE, T.; OMERNICK, M.; PENUKONDA, N.; PHELPS, A.; ROSS, J.; ROSS, M.; SALEK, A.; SAMADIANI, E.; SEVERN, C.; SIZIKOV, G.; SNELHAM, M.; SOUTER, J.; STEINBERG, D.; SWING, A.; TAN, M.; THORSON, G.; TIAN, B.; TOMA, H.; TUTTLE, E.; VASUDEVAN, V.; WALTER, R.; WANG, W.; WILCOX, E.; YOON, D. H. In-datacenter performance analysis of a tensor processing unit. **arXiv**, 2017. Citado na página 24.

KAVRAKI, L. E.; SVESTKA, P.; LATOMBE, J.; OVERMARS, M. H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. **IEEE Transactions on Robotics and Automation**, v. 12, n. 4, p. 566–580, 1996. Disponível em: <https://ieeexplore.ieee.org/document/508439>. Citado na página 20.

KILBURN, R. N.; STAHLSCHMIDT, P. Description of two new species of drillia from the indo-pacific (gastropoda: Conoidea: Drilliidae). Archiv für Molluskenkunde: International Journal of Malacology, v. 141, n. 1, 2012. Citado na página 26.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization: **Proceedings of the 3rd International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2015. Citado 2 vezes nas páginas 35 e 45.

KIRAN, P. S. R.; KUMAR, A.; MOHAN, R. Aerial-ground robotic system for terrain estimation and navigation: **Fifth Indian Control Conference (ICC)**. IEEE, 2019. Disponível em: <https://doi.org/10.1109/indiancc.2019.8715614>. Citado na página 14.

KOHLER, J.; DANESHMAND, H.; LUCCHI, A.; HOFMANN, T.; ZHOU, M.; NEYMEYR, K. Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization: . PMLR, 2019. (**Proceedings of Machine Learning Research**, v. 89), p. 806–815. Disponível em: <http://proceedings.mlr.press/v89/kohler19a.html>. Citado na página 30.

KUHNER, D.; FIEDERER, L.; ALDINGER, J.; BURGET, F.; VöLKER, M.; SCHIRRMEISTER, R.; DO, C.; BOEDECKER, J.; NEBEL, B.; BALL, T.; BURGARD, W. A service assistant combining autonomous robotics, flexible goal formulation, and deep-learning-based brain–computer interfacing. **Robotics and Autonomous Systems**, v. 116, p. 98 – 113, 2019. ISSN 0921-8890. Disponível em: <https://doi.org/10.1016/j.robot.2019.02.015>. Citado na página 20.

LADICKY, L. **Global Structured Models Towards Scene Understanding** . Tese (Doutorado) — Oxford Brookes University, 2011. Citado na página 53.

LÄNGKVIST, M.; KISELEV, A.; ALIREZAIE, M.; LOUTFI, A. Classification and segmentation of satellite orthoimagery using convolutional neural networks. **Remote Sensing**, v. 8, p. 329, 04 2016. Disponível em: <https://doi.org/10.3390/rs8040329>. Citado na página 15.

LAVALLE, S. M.; KUFFNER, J. J. J. Randomized kinodynamic planning. **The International Journal of Robotics Research**, v. 20, n. 5, p. 378–400, 2001. Disponível em: <https://doi.org/10.1177/02783640122067453>. Citado na página 20.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation Applied to Handwritten Zip Code Recognition. **Neural Computation**, v. 1, n. 4, p. 541–551, 1989. Disponível em: <https://doi.org/10.1162/neco.1989.1.4.541>. Citado na página 24.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998. Disponível em: <https://doi.org/10.1109/5.726791>. Citado 2 vezes nas páginas 24 e 34.

LEONARD, J. J.; DURRANT-WHYTE, H. F. Mobile robot localization by tracking geometric beacons. **IEEE Transactions on Robotics and Automation**, v. 7, n. 3, p. 376–382, 1991. Disponível em: <https://ieeexplore.ieee.org/document/88147>. Citado na página 17.

LI, F.-F.; JOHNSON, J.; YEUNG, S. **Lecture 5: Convolutional Neural Networks** . 2019. Disponível em: <http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture05.pdf>. Acesso em: 30 ago 2020. Citado 2 vezes nas páginas 27 e 28.

LIM, J.; FRAHM, J.-M.; POLLEFEYS, M. Online environment mapping using metric-topological maps. **The International Journal of Robotics Research**, v. 31, n. 12, p. 1394–1408, 2012. Disponível em: <https://doi.org/10.1177/0278364912461455>. Citado na página 18.

LINHUI, L.; MENGMENG, W.; XINLI, D.; JING, L.; YUNPENG, Z. Convolutional neural network applied to traversability analysis of vehicles. **Advances in Mechanical Engineering**, v. 5, p. 542832, 2013. Disponível em: <https://doi.org/10.1155/2013/542832>. Citado 2 vezes nas páginas 14 e 15.

LÜDDECKE, T.; KULVICIUS, T.; WÖRGÖTTER, F. Context-based affordance segmentation from 2D images for robot actions. **Robotics and Autonomous Systems**, v. 119, p. 92 – 107, 2019. ISSN 0921-8890. Citado na página 15.

MAGGIORI, E.; TARABALKA, Y.; CHARPIAT, G.; ALLIEZ, P. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark: **IEEE International Geoscience and Remote Sensing Symposium (IGARSS)**. [S.l.: s.n.], 2017. p. 3226–3229. ISSN 2153-7003. Citado 11 vezes nas páginas 38, 39, 40, 41, 49, 55, 61, 76, 77, 78 e 79.

MARÍN, P.; HUSSEIN, A.; GOMEZ, D. M.; ESCALERA, A. de la. Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles. **Journal of Advanced Transportation**, v. 2018, p. 1–10, 02 2018. Disponível em: <https://doi.org/10.1155/2018/6392697>. Citado na página 20.

MARTINEZ-SOLTERO, G.; ALANIS, A. Y.; ARANA-DANIEL, N.; LOPEZ-FRANCO, C. Semantic segmentation for aerial mapping. **Mathematics**, Multidisciplinary Digital Publishing Institute, v. 8, n. 9, p. 1456, 2020. Citado 2 vezes nas páginas 15 e 43.

MISSING LINK AI. **7 Types of Neural Network Activation Functions: How to Choose?** . 2019. Disponível em: <https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/>. Acesso em: 31 ago 2020. Citado na página 29.

NAGI, J.; DUCATELLE, F.; CARO, G. A. D.; CIREŞAN, D.; MEIER, U.; GIUSTI, A.; NAGI, F.; SCHMIDHUBER, J.; Gambardella, L. M. Max-pooling convolutional neural networks for vision-based hand gesture recognition: **IEEE International Conference on Signal and Image Processing Applications (ICSIPA)**. [s.n.], 2011. p. 342–347. Disponível em: <https://doi.org/10.1109/ICSIPA.2011.6144164>. Citado na página 28.

NEGENBORN, R. **Robot Localization and Kalman Filters: On Finding Your Position in a Noisy World** . Dissertação (Mestrado) — Institute of Information and Computing Sciences, Utrecht Universiy, Utrecht, Setembro 2003. Disponível em: <https://www.researchgate.net/publication/291456241_Map_Representation_for_Robots>. Citado na página 17.

NEON. **Airborne data sample** . 2013. National Ecological Observatory Network. Disponível em: <http://www.neonscience.org/data-resources/get-data/airborne-data>. Acesso em: 19 dec 2017. Citado 5 vezes nas páginas 38, 40, 41, 55 e 79.

ONO, M.; HEVERLY, M.; ROTHROCK, B.; ALMEIDA, E.; CALEF, F.; SOLIMAN, T.; WILLIAMS, N.; GENGL, H.; ISHIMATSU, T.; NICHOLAS, A.; STILLEY, E.; OTSU, K.; LANGE, R.; MILKOVICH, S. M. Mars 2020 site-specific mission performance analysis: Part 2. surface traversability: _____. **2018 AIAA SPACE and Astronautics Forum and Exposition**. [S.l.: s.n.], 2018. Citado 2 vezes nas páginas 15 e 43.

PAOLA, D. D.; MILELLA, A.; CICIRELLI, G.; DISTANTE, A. An autonomous mobile robotic system for surveillance of indoor environments. **International Journal of Advanced Robotic Systems**, v. 7, n. 1, p. 8, 2010. Disponível em: <https://doi.org/10.5772/7254>. Citado na página 20.

PAPADAKIS, P. Terrain traversability analysis methods for unmanned ground vehicles: A survey. **Engineering Applications of Artificial Intelligence**, Elsevier BV, v. 26, n. 4, p. 1373–1385, apr 2013. Disponível em: <https://doi.org/10.1016/j.engappai.2013.01.006>. Citado 4 vezes nas páginas 14, 21, 22 e 23.

PETERSON, J.; CHAUDHRY, H.; ABDELATTY, K.; BIRD, J.; KOCHERSBERGER, K. Online aerial terrain mapping for ground robot navigation. **Sensors**, MDPI AG, v. 18, n. 2, p. 630, feb 2018. Disponível em: <https://doi.org/10.3390/s18020630>. Citado 8 vezes nas páginas 14, 15, 23, 52, 53, 58, 59 e 61.

PIAZZI, A.; BIANCO, C. G. L.; BERTOZZI, M.; FASCIOLI, A.; BROGGI, A. Quintic g/sup 2/-splines for the iterative steering of vision-based autonomous vehicles. **IEEE Transactions on Intelligent Transportation Systems**, v. 3, n. 1, p. 27–36, 2002. Disponível em: <https://ieeexplore.ieee.org/document/994793>. Citado na página 21.

PRECHELT, L. **Early Stopping – But When? Neural Networks: Tricks of the Trade: Second Edition**: _____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 53–67. ISBN 978-3-642-35289-8. Citado na página 45.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. Searching for activation functions. **arXiv**, 2017. Disponível em: <http://arxiv.org/abs/1710.05941>. Citado na página 29.

RAMER, U. An iterative procedure for the polygonal approximation of plane curves. **Computer Graphics and Image Processing**, v. 1, n. 3, p. 244 – 256, 1972. ISSN 0146-664X. Disponível em: <https://doi.org/10.1016/S0146-664X(72)80017-0>. Citado na página 49.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, Oct 1986. ISSN 1476-4687. Disponível em: <https://doi.org/10.1038/323533a0>. Citado 2 vezes nas páginas 24 e 34.

SAEEDI, S.; TRENTINI, M.; SETO, M.; LI, H. Multiple-Robot Simultaneous Localization and Mapping: A Review. **Journal of Field Robotics**, v. 33, n. 1, p. 3–46, 2016. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21620>. Citado na página 19.

SALGADO, R.; PRIETO, A.; BELLAS, F.; CALVO-VARELA, L.; DURO, R. Motivational engine with autonomous sub-goal identification for the multilevel darwinist brain. **Biologically Inspired Cognitive Architectures**, v. 17, p. 1 – 11, 2016. ISSN 2212-683X. Disponível em: <https://doi.org/10.1016/j.bica.2016.07.003>. Citado na página 20.

SANTURKAR, S.; TSIPRAS, D.; ILYAS, A.; MADRY, A. How does batch normalization help optimization?: BENGIO, S.; WALLACH, H.; LAROCHELLE, H.; GRAUMAN, K.; CESA-BIANCHI, N.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 31**. Curran Associates, Inc., 2018. p. 2483–2493. Disponível em: <http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf>. Citado na página 30.

SCHILLING, F.; CHEN, X.; FOLKESSON, J.; JENSFELT, P. Geometric and visual terrain classification for autonomous mobile navigation: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2017. p. 2678–2684. Citado 2 vezes nas páginas 15 e 43.

SERAJI, H. Traversability index: a new concept for planetary rovers. **Proceedings 1999 IEEE International Conference on Robotics and Automation**, v. 3, p. 2006–2013 vol.3, 1999. ISSN 1050-4729. Citado na página 14.

SHELHAMER, E.; LONG, J.; DARRELL, T. Fully convolutional networks for semantic segmentation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 39, n. 4, p. 640–651, 2017. Citado na página 32.

SHNEIER, M.; CHANG, T.; HONG, T.; SHACKLEFORD, W.; BOSTELMAN, R.; ALBUS, J. S. Learning traversability models for autonomous mobile vehicles. **Autonomous Robots**, v. 24, n. 1, p. 69–86, Jan 2008. ISSN 1573-7527. Disponível em: <https://doi.org/10.1007/s10514-007-9063-6>. Citado na página 14.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. **Journal of Big Data**, v. 6, n. 1, p. 60, Jul 2019. ISSN 2196-1115. Disponível em: <https://doi.org/10.1186/s40537-019-0197-0>. Citado na página 42.

SHU, T.; GHARAATY, S.; XIE, W.; JOUBAIR, A.; BONEV, I. A. Dynamic path tracking of industrial robots with high accuracy using photogrammetry sensor. **IEEE/ASME Transactions on Mechatronics**, v. 23, n. 3, p. 1159–1170, 2018. Disponível em: <https://doi.org/10.1109/TMECH.2018.2821600>. Citado na página 23.

SICILIANO, B.; KHATIB, O. **Springer Handbook of Robotics** . Berlin, Heidelberg: Springer-Verlag, 2008. ISBN 978-3-319-32552-1. Disponível em: <https://www.springer.com/gp/book/9783540303015>. Citado 2 vezes nas páginas 19 e 22.

SOUISSI, O.; BENATITALLAH, R.; DUVIVIER, D.; ARTIBA, A.; BELANGER, N.; FEYZEAU, P. Path planning: A 2013 survey: **Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)**. [s.n.], 2013. p. 1–8. Disponível em: <https://ieeexplore.ieee.org/abstract/document/6761521>. Citado na página 20.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>. Citado na página 30.

STENTZ, A. Optimal and efficient path planning for partially-known environments: **Proceedings of the 1994 IEEE International Conference on Robotics and Automation**. [s.n.], 1994. p. 3310–3317 vol.4. Disponível em: <https://doi.org/10.1109/ROBOT.1994.351061>. Citado na página 20.

SULTANA, F.; SUFIAN, A.; DUTTA, P. A Review of Object Detection Models Based on Convolutional Neural Network: _____. **Intelligent Computing: Image Processing Based Applications**. Singapore: Springer Singapore, 2020. p. 1–16. ISBN 978-981-15-4288-6. Disponível em: <https://doi.org/10.1007/978-981-15-4288-6_1>. Citado na página 32.

TAYLOR, J. Proprioception: SQUIRE, L. R. (Ed.). Encyclopedia of Neuroscience. Oxford: Academic Press, 2009. p. 1143–1149. ISBN 978-0-08-045046-9. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780080450469019070>. Citado na página 22.

THRUN, S. Learning metric-topological maps for indoor mobile robot navigation. **Artificial Intelligence**, v. 99, n. 1, p. 21 – 71, 1998. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0004370297000787>. Citado 2 vezes nas páginas 18 e 25.

TOWARDS DATA SCIENCE. **Coding Deep Learning for Beginners — Linear Regression (Part 3): Training with Gradient Descent** . 2018. Disponível em: <https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-gradient-descent-fcd5e0fc077d>. Acesso em: 06 set 2020. Citado na página 34.

VANDAPEL, N.; DONAMUKKALA, R. R.; HEBERT, M. Unmanned ground vehicle navigation using aerial ladar data. **The International Journal of Robotics Research**, Sage Publications, Pittsburgh, PA, v. 25, n. 1, p. 31–51, January 2006. Citado na página 14.

VÉRAS, L. G. D. O.; MEDEIROS, F. L. L.; GUIMARÁES, L. N. F. Systematic literature review of sampling process in rapidly-exploring random trees. **IEEE Access**, v. 7, p. 50933–50953, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2908100>. Citado na página 21.

WAIBEL, A.; HANAZAWA, T.; HINTON, G.; SHIKANO, K.; LANG, K. J. Phoneme recognition using time-delay neural networks. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, v. 37, n. 3, p. 328–339, 1989. Disponível em: <https://doi.org/10.1109/29.21701>. Citado na página 24.

WALTER, M.; HEMACHANDRA, S.; HOMBERG, B.; TELLEX, S.; TELLER, S. Learning semantic maps from natural language descriptions: **Robotics: Science and Systems IX**. Robotics: Science and Systems Foundation, 2013. Disponível em: <https://doi.org/10.15607%2Frss.2013.ix.004>. Citado 3 vezes nas páginas 18, 19 e 20.

WANG, Q.; MA, Y.; ZHAO, K.; TIAN, Y. A comprehensive survey of loss functions in machine learning. **Annals of Data Science**, Apr 2020. ISSN 2198-5812. Disponível em: <https://doi.org/10.1007/s40745-020-00253-5>. Citado na página 33.

WASSERMAN, P. D.; SCHWARTZ, T. Neural networks. ii. what are they and why is everybody so interested in them now? **IEEE Expert**, v. 3, n. 1, p. 10–15, 1988. Disponível em: <https://doi.org/10.1109/64.2091>. Citado na página 24.

WULFMEIER, M.; RAO, D.; WANG, D. Z.; ONDRUSKA, P.; POSNER, I. Large-scale cost function learning for path planning using deep inverse reinforcement learning. **The International Journal of Robotics Research**, v. 36, n. 10, p. 1073–1087, 2017. Citado 2 vezes nas páginas 15 e 43.

YANG, J.-M.; TSENG, C.-M.; TSENG, P. Path planning on satellite images for unmanned surface vehicles. **International Journal of Naval Architecture and Ocean Engineering**, Elsevier BV, v. 7, n. 1, p. 87–99, jan 2015. Disponível em: <https://doi.org/10.1515/ijnaoe-2015-0007>. Citado 2 vezes nas páginas 14 e 23.

YANG, K.; BERGASA, L. M.; ROMERA, E.; WANG, K. Robustifying semantic cognition of traversability across wearable RGB-depth cameras. **Applied Optics**, OSA, v. 58, n. 12, p. 3141–3155, Apr 2019. Disponível em: <https://doi.org/10.1364/AO.58.003141>. Citado 2 vezes nas páginas 15 e 43.

YANG, L.; QI, J.; SONG, D.; XIAO, J.; HAN, J.; XIA, Y. Survey of robot 3d path planning algorithms. **Journal of Control Science and Engineering**, Hindawi Limited, London, GBR, v. 2016, p. 5, mar. 2016. ISSN 1687-5249. Disponível em: <https://doi.org/10.1155/2016/7426913>. Citado na página 20.

YE, C.; WEBB, P. A sub goal seeking approach for reactive navigation in complex unknown environments. **Robotics and Autonomous Systems**, v. 57, n. 9, p. 877 – 888, 2009. ISSN 0921-8890. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0921889009000839>. Citado na página 19.

YI, C.; JEONG, S.; CHO, J. Map representation for robots. **Smart Computing Review**, v. 2, p. 18–27, 02 2012. Disponível em: <https://www.researchgate.net/publication/291456241_Map_Representation_for_Robots>. Citado na página 17.

ZEILER, M. D. Adadelta: An adaptive learning rate method. **arXiv**, 2012. Disponível em: <https://arxiv.org/abs/1212.5701>. Citado na página 35.

ZHOU, D.-X. Universality of deep convolutional neural networks. **Applied and Computational Harmonic Analysis**, v. 48, n. 2, p. 787 – 794, 2020. ISSN 1063-5203. Disponível em: <https://doi.org/10.1016/j.acha.2019.06.004>. Citado na página 28.

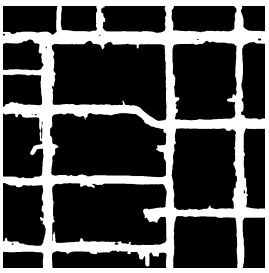
ZHOU, R.; GAO, H.; FENG, W.; DENG, Z.; LI, N. Mapping for planetary rovers from terramechanics perspective: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [s.n.], 2019. p. 1869–1874. Disponível em: <https://doi.org/10.1109/IROS40897.2019.8967984>. Citado 2 vezes nas páginas 15 e 43.

APÊNDICE A – DATASET

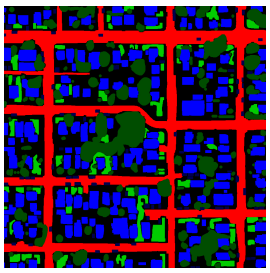
Amostra 1

Imagem aérea 1

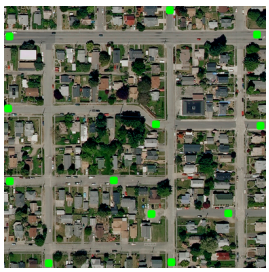
Localização	Condado de Kitsap, WA, EUA
Área	$300 \times 300\ m^2$
Dimensões da imagem	$1000 \times 1000\ px$
Resolução espacial	$0,3 \times 0,3\ m^2/px$
Fonte	Recorte de Maggiori et al. (2017)



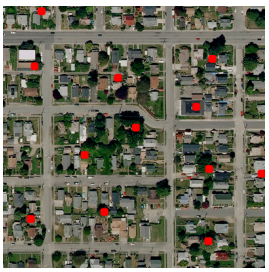
Atravessabilidades



Classes



Pontos alcançáveis



Pontos inalcançáveis

Classes:

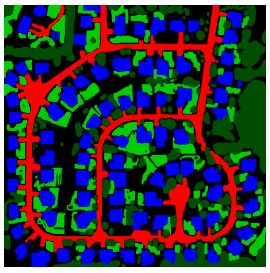
■ Desconhecido ■ Estrada ■ Construções ■ Gramado ■ Vegetação ■ Veículo

Amostra 2

Imagem aérea 2

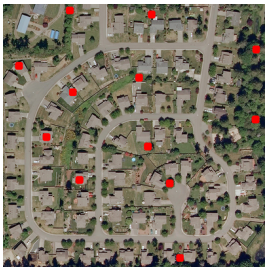
Localização	Condado de Kitsap, WA, EUA
Área	$300 \times 300\ m^2$
Dimensões da imagem	$1000 \times 1000\ px$
Resolução espacial	$0,3 \times 0,3\ m^2/px$
Fonte	Recorte de Maggiori et al. (2017)

Atravessabilidades



Classes

Pontos alcançáveis



Pontos inalcançáveis

Classes:

■ Desconhecido ■ Estrada ■ Construções ■ Gramado ■ Vegetação ■ Veículo

Amostra 3

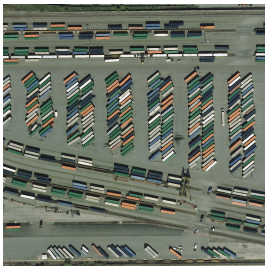
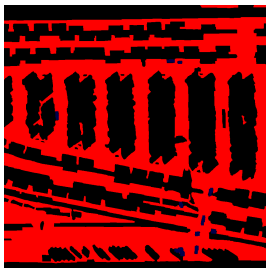


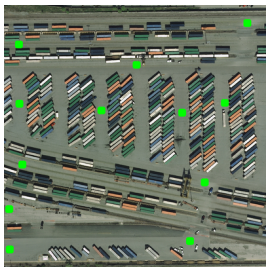
Imagem aérea 3

Localização	Chicago, IL, EUA
Área	$300 \times 300\ m^2$
Dimensões da imagem	$1000 \times 1000\ px$
Resolução espacial	$0,3 \times 0,3\ m^2/px$
Fonte	Recorte de Maggiori et al. (2017)

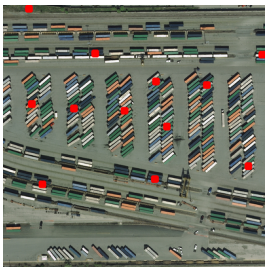
Atravessabilidades



Classes



Pontos alcançáveis



Pontos inalcançáveis

Classes:

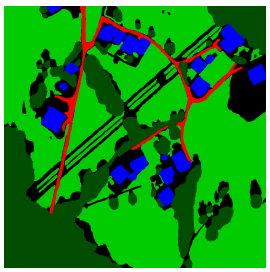
■ Desconhecido ■ Estrada ■ Veículo

Amostra 4

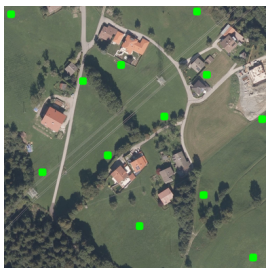
Imagem aérea 4

Localização	Innsbruck, Áustria
Área	$300 \times 300\ m^2$
Dimensões da imagem	$1000 \times 1000\ px$
Resolução espacial	$0,3 \times 0,3\ m^2/px$
Fonte	Recorte de Maggiori et al. (2017)

Atravessabilidades



Classes



Pontos alcançáveis

Pontos inalcançáveis

Classes:

■ Desconhecido ■ Estrada ■ Construções ■ Gramado ■ Vegetação

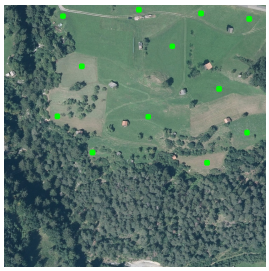
Amostra 5

Imagem aérea 5

Localização	Tyrol Oeste, Áustria
Área	$300 \times 300\ m^2$
Dimensões da imagem	$1000 \times 1000\ px$
Resolução espacial	$0,3 \times 0,3\ m^2/px$
Fonte	Recorte de Maggiori et al. (2017)

Atravessabilidades

Classes



Pontos alcançáveis

Pontos inalcançáveis

Classes:

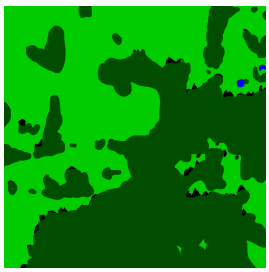
■ Desconhecido ■ Estrada ■ Construções ■ Gramado ■ Vegetação

Amostra 6

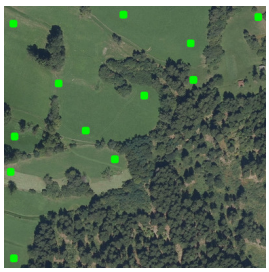
Imagem aérea 6

Localização	Tyrol Leste, Áustria
Área	$300 \times 300\ m^2$
Dimensões da imagem	$1000 \times 1000\ px$
Resolução espacial	$0,3 \times 0,3\ m^2/px$
Fonte	Recorte de Maggiori et al. (2017)

Atravessabilidades



Classes



Pontos alcançáveis

Pontos inalcançáveis

Classes:

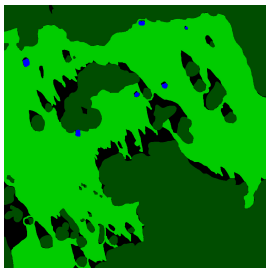
■ Desconhecido ■ Construções ■ Gramado ■ Vegetação

Amostra 7

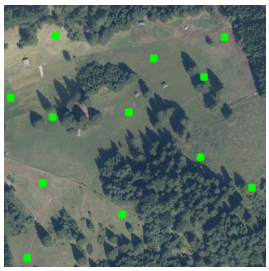
Imagem aérea 7

Localização	Tyrol Oeste, Áustria
Área	$300 \times 300\ m^2$
Dimensões da imagem	$1000 \times 1000\ px$
Resolução espacial	$0,3 \times 0,3\ m^2/px$
Fonte	Recorte de Maggiori et al. (2017)

Atravessabilidades



Classes



Pontos alcançáveis

Pontos inalcançáveis

Classes:

■ Desconhecido ■ Construções ■ Gramado ■ Vegetação

Amostra 8

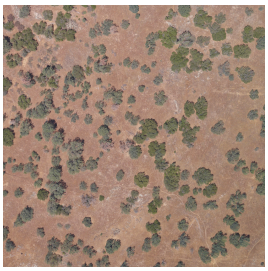
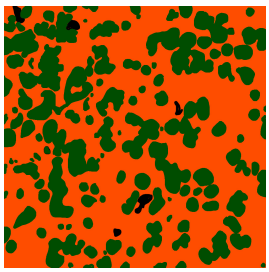


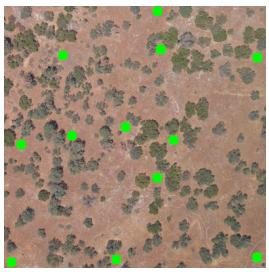
Imagem aérea 8

Localização	Coarsegold, CA, EUA
Área	$250 \times 250\ m^2$
Dimensões da imagem	$1000 \times 1000\ px$
Resolução espacial	$0,25 \times 0,25\ m^2/px$
Fonte	NEON (2013)

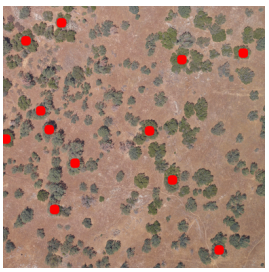
Atravessabilidades



Classes



Pontos alcançáveis



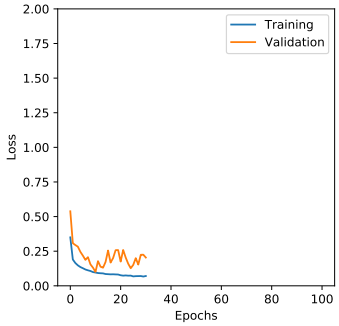
Pontos inalcançáveis

Classes:

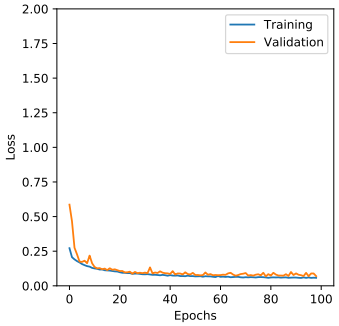
■ Desconhecido ■ Vegetação ■ Terreno arenoso

APÊNDICE B – PERDAS DE TREINAMENTO E VALIDAÇÃO

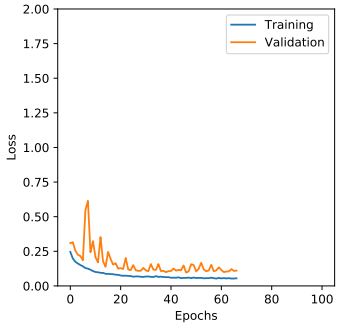
TFCN-RGB (*dataset* completo)



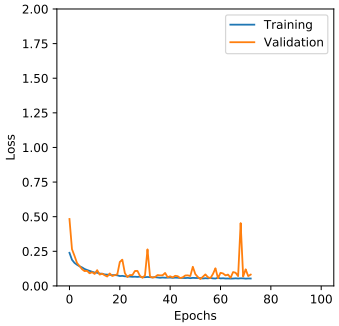
(a) Etapa 1



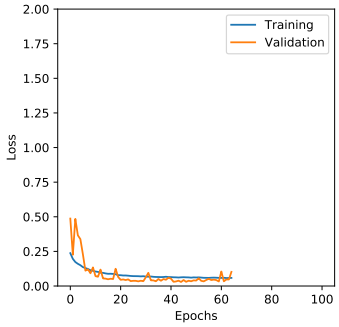
(b) Etapa 2



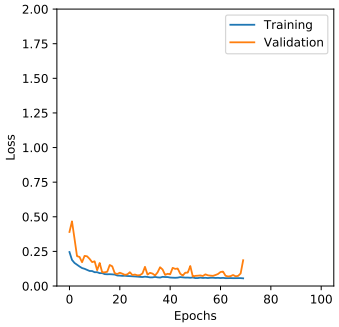
(c) Etapa 3



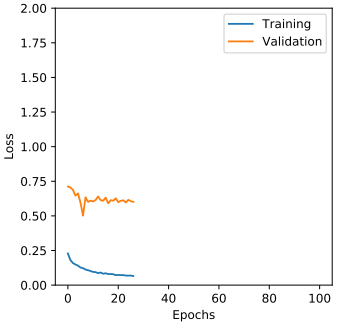
(d) Etapa 4



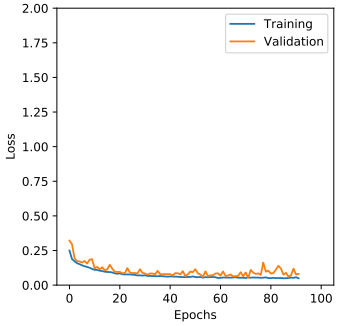
(e) Etapa 5



(f) Etapa 6

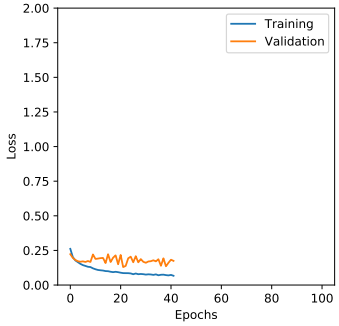


(g) Etapa 7

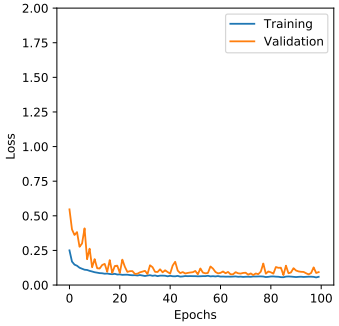


(h) Etapa 8

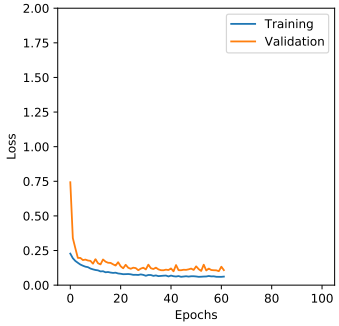
TFCN-G (*dataset* completo)



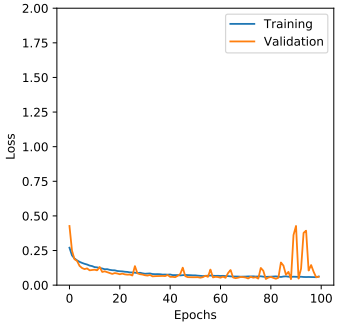
(a) Etapa 1



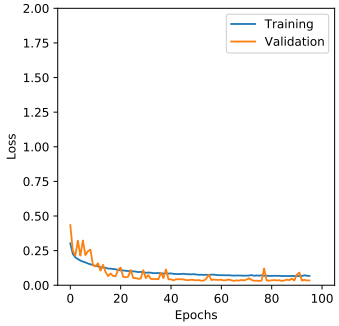
(b) Etapa 2



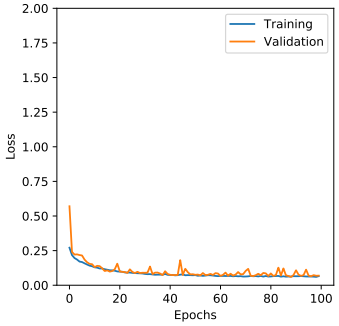
(c) Etapa 3



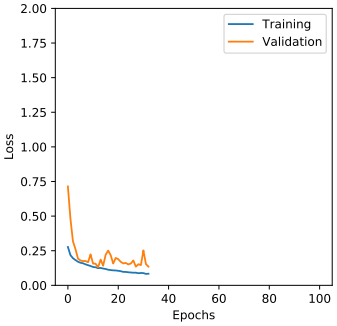
(d) Etapa 4



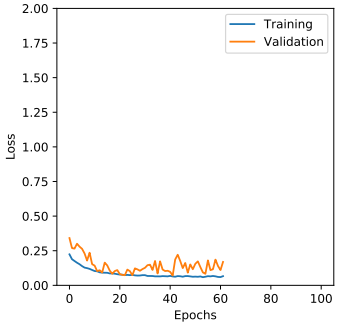
(e) Etapa 5



(f) Etapa 6

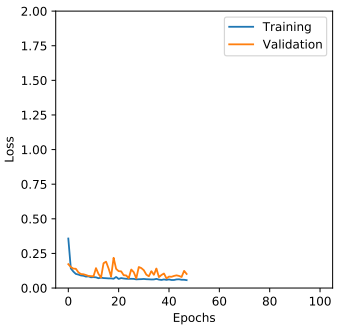


(g) Etapa 7

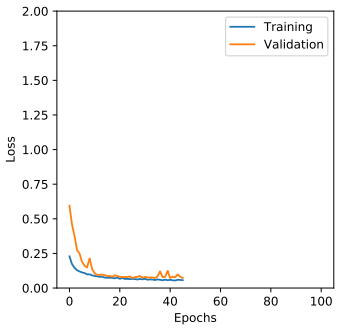


(h) Etapa 8

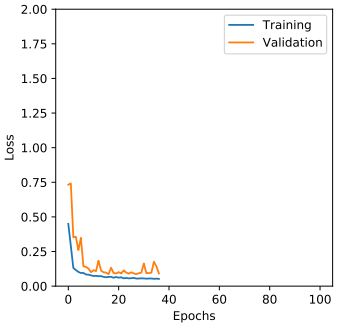
TFCN-RGB (*outliers* removidos)



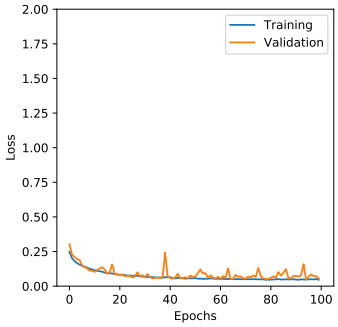
(a) Etapa 1



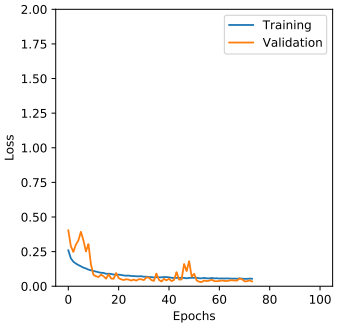
(b) Etapa 2



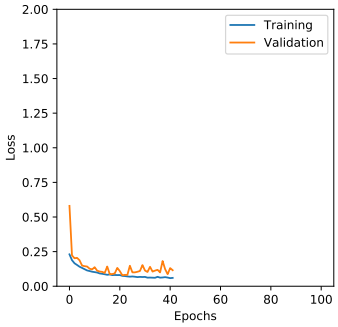
(c) Etapa 3



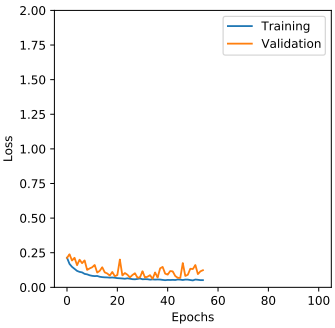
(d) Etapa 4



(e) Etapa 5

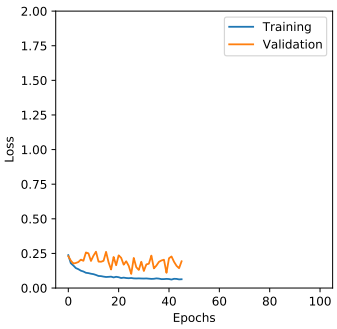


(f) Etapa 6

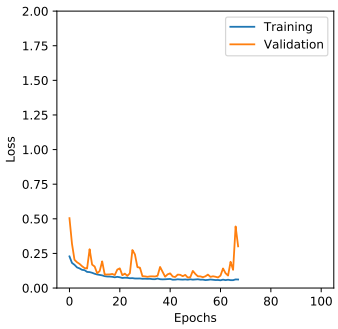


(g) Etapa 7

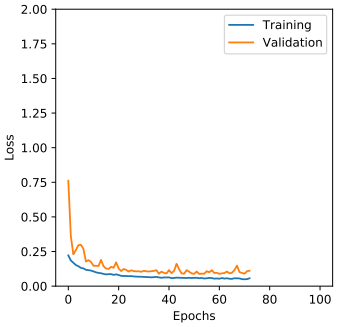
TFCN-G (*outliers* removidos)



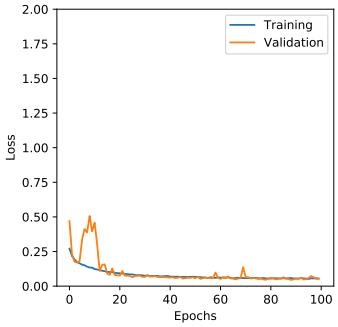
(a) Etapa 1



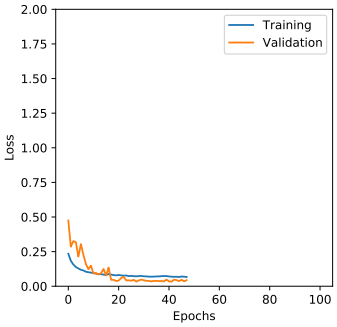
(b) Etapa 2



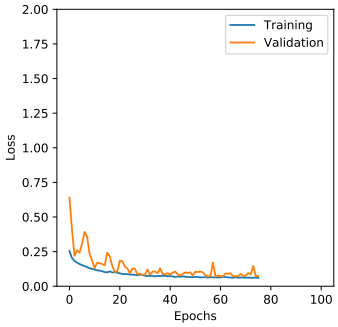
(c) Etapa 3



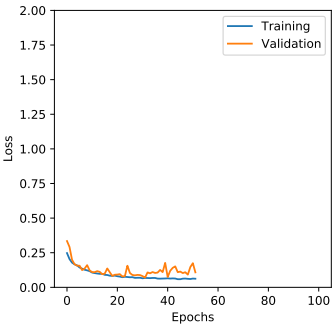
(d) Etapa 4



(e) Etapa 5



(f) Etapa 6



(g) Etapa 7