UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Claiton Neisse

REDES NEURAIS PROFUNDAS NA COMPUTAÇÃO DE HEURÍSTICAS PARA ALGORITMOS DE BUSCA DE CAMINHOS EM MAPAS VIRTUAIS CONTENDO ELEVAÇÃO E INCLINAÇÃO

Santa Maria, RS
2021

Claiton Neisse

REDES NEURAIS PROFUNDAS NA COMPUTAÇÃO DE HEURÍSTICAS PARA ALGORITMOS DE BUSCA DE CAMINHOS EM MAPAS VIRTUAIS CONTENDO ELEVAÇÃO E INCLINAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

ORIENTADOR: Prof. Luís Alvaro de Lima Silva

475
Santa Maria, RS
2021

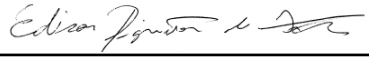
Claiton Neisse

REDES NEURAIS PROFUNDAS NA COMPUTAÇÃO DE HEURÍSTICAS PARA ALGORITMOS DE BUSCA DE CAMINHOS EM MAPAS VIRTUAIS CONTENDO ELEVAÇÃO E INCLINAÇÃO

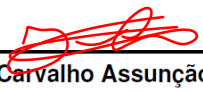
Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

Aprovado em 9 de fevereiro de 2021:

Luís Alvaro de Lima Silva, Dr. (UFSM)
(Presidente/Orientador)



Edison Pignaton de Freitas, Dr. (UFRGS)



Joaquim Vinicius Carvalho Assunção, Dr. (UFSM)

Santa Maria, RS
2021

RESUMO

REDES NEURAIS PROFUNDAS NA COMPUTAÇÃO DE HEURÍSTICAS PARA ALGORITMOS DE BUSCA DE CAMINHOS EM MAPAS VIRTUAIS CONTENDO ELEVAÇÃO E INCLINAÇÃO

AUTOR: Claiton Neisse
ORIENTADOR: Luís Alvaro de Lima Silva

Redes neurais profundas e algoritmos de busca de caminhos (*pathfinding*) têm sido investigados na área de Inteligência Artificial (IA). Apesar disso, essas áreas de pesquisa ainda requerem uma maior integração, principalmente visando a proposta de algoritmos de *pathfinding* que exploram informações de relevo nas computações de rotas com menores distâncias e custos topográficos. Este trabalho emprega técnicas de redes neurais profundas na construção de funções heurísticas utilizadas na otimização de algoritmos de *pathfinding* que exploram elevação e inclinação representadas em mapas virtuais de grandes dimensões. Experimentos compararam tais heurísticas com a heurística tradicional na execução do algoritmo de *pathfinding* A^*. Resultados mostraram que o emprego de redes neurais profundas pode reduzir o custo computacional do algoritmo de *pathfinding* em mapas virtuais contendo informações topográficas.

Palavras-chave: Busca de caminhos. Redes neurais profundas. Inteligência artificial.

ABSTRACT

DEEP NEURAL NETWORKS IN COMPUTING HEURISTICS FOR PATHFINDING ALGORITHMS IN VIRTUAL MAPS CONTAINING ELEVATION AND SLOPE

AUTHOR: Claiton Neisse
ADVISOR: Luís Alvaro de Lima Silva

Deep neural networks and pathfinding algorithms have been investigated in the area of Artificial Intelligence (AI). Despite this, these areas of research still require greater integration, mainly with a view to proposing pathfinding algorithms that explore relief information in route computations with lower distances and topographic costs. This work uses techniques of deep neural networks in the construction of heuristic functions used in the optimization of pathfinding algorithms that explore elevation and inclination represented in large virtual maps. Experiments compared such heuristics with traditional heuristics in the execution of the A^* pathfinding algorithm. Results showed that the use of deep neural networks can reduce the computational cost of the pathfinding algorithm in virtual maps containing topographic information.

Keywords: Pathfinding. Deep neural networks. Artificial Intelligence.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Redes neurais profundas (Deep Neural Networks – DNN) (GOODFELLOW et al., 2016) têm sido exploradas com sucesso em diversas áreas de aplicação. DNN são algoritmos de aprendizagem de máquina formados por diversas camadas de unidades de processamento elementares, interagindo entre si, resultando num modelo matemático que possibilita a aprendizagem ou o reconhecimento de padrões em conjuntos de dados. O processo de aprendizagem consiste, essencialmente, na busca das características das unidades elementares, de maneira que a DNN tenha boa capacidade de generalização.

Em contraste, algoritmos de *pathfinding* (ALGFOOR; SUNAR; KOLIVAND, 2015) procuram o caminho mais curto ou de menor custo entre dois nodos, uma origem e um destino, em uma estrutura de representação do terreno em uso. Alguns algoritmos utilizam heurísticas na composição de suas funções de custo. Nesse caso, uma heurística é uma função que estima o custo entre um nodo qualquer do mapa e um destino, sendo zero quando aplicada ao destino. Essa estimativa utiliza o conhecimento sobre o problema abordado para auxiliar o algoritmo na escolha do próximo nodo a ser analisado, indicando o quanto o nodo é promissor em relação a tarefa de chegar ao destino. Entre outros objetivos, o emprego dessa técnica procura otimizar o processo de busca reduzindo a quantidade de nodos avaliados da estrutura de representação.

Embora técnicas de DNN e *pathfinding* da Inteligência Artificial (IA) tenham sido exploradas com sucesso na resolução de inúmeros problemas de aplicação, elas ainda são pouco exploradas em conjunto, havendo a necessidade de investigar a construção de heurísticas baseadas em DNN para otimizar a busca de caminhos em terrenos virtuais. Assim como explorado neste trabalho, existe a necessidade de desenvolvimento de algoritmos de *pathfinding* que consideram informações topográficas nas suas computações de caminhos de menor custo, onde tal custo pode ser determinado de diferentes formas na computação de caminhos em mapas de grandes dimensões em diferentes problemas de aplicação.

Trabalhos descritos na literatura têm explorado informações topográficas de mapas virtuais na computação de algoritmos de *pathfinding*. (CHEN; SHI; LIU, 2009) utilizam uma estrutura irregular para representar a superfície de um terreno, onde cada polígono possui um vetor normal ao plano que representa o terreno. A busca de caminhos considera o ângulo entre os vetores normais em polígonos adjacentes. (GANGANATH; CHENG; TSE, 2015) transformam um modelo digital de elevação em um grafo ponderado, onde cada vértice representa um ponto do terreno. A função heurística utilizada resulta em caminhos com um balanceamento entre o comprimento e o perfil de inclinação do caminho, buscando caminhos com menor custo energético para deslocamento de robôs. No projeto de pesquisa onde este Trabalho de Conclusão de Curso (TCC) está inserido, (CHAGAS, 2019) propõe um algoritmo de *pathfinding* que utiliza inclinações do terreno em sua função

de custo buscando caminhos suavizados em mapas virtuais com informações topográficas. Em particular, este TCC expande a pesquisa apresentada por (CHAGAS, 2019) em diferentes frentes.

Outros trabalhos descritos na literatura têm investigado o uso de DNN no apoio a resolução de diferentes problemas de busca. (AGOSTINELLI et al., 2019) propõem a utilização de técnicas de aprendizagem por reforço profundo em funções heurísticas para solução do cubo de Rubik. (ARIKI; NARIHIRA, 2019) descrevem uma arquitetura convolucional de DNN para analisar um mapa de obstáculos e um destino, produzindo um mapa heurístico, onde cada ponto do mapa contém o custo heurístico até o destino. (JINDAL et al., 2017) propõem uma arquitetura de DNN para predizer o tempo de viagem de táxi entre duas coordenadas geográficas, combinando a predição da distância entre origem e destino e o horário do dia que a viagem deve ser realizada. Um algoritmo semelhante ao A^* é proposto por (LI et al., 2016), utilizando uma DNN como heurística, para predizer a dificuldade de buscar um caminho em um mapa. (TAKAHASHI et al., 2019) exploram a aplicação de arquiteturas para segmentação de objetos e geração de imagens em heurísticas de problemas de *pathfinding*. (WANG et al., 2019) utilizam uma arquitetura recorrente na heurística do algoritmo A^* para gerar recomendações de rota personalizadas. No projeto de pesquisa onde este TCC está inserido, (DOEBBER, 2019) descreve a aplicação de uma arquitetura *feedforward* nas computações de *pathfinding* em mapas de labirintos bidimensionais. Novamente, este TCC expande a pesquisa apresentada por (DOEBBER, 2019).

Em resumo, apesar desses trabalhos empregarem diferentes arquiteturas de DNN, eles não investigam o uso de DNN na otimização da busca de caminhos em terrenos contendo informações topográficas. Para atacar esse problema, esse TCC analisa técnicas de DNN na resolução de problemas de *pathfinding* em mapas virtuais de grandes dimensões, tal como (DOEBBER, 2019), porém na resolução de problemas de *pathfinding* em mapas contendo informações topográficas, como empregado em (CHAGAS, 2019) e utilizando mapas oriundos de modelo digital de elevação, como em (GANGANATH; CHENG; TSE, 2015). Em particular, o trabalho descreve como usar DNN na construção de heurísticas para algoritmos de *pathfinding*. A partir de diferentes experimentos realizados, o desempenho de heurísticas construídas com DNN são comparados e analisados com heurísticas tradicionais na busca de caminhos em diferentes mapas contendo informações topográficas.

O trabalho está estruturado da seguinte forma. O segundo capítulo apresenta uma revisão bibliográfica sobre redes neurais profundas, técnicas de *ensemble* de redes neurais e busca de caminhos. Aborda, ainda, trabalhos relacionados que investigam a utilização de DNN em computações heurísticas e problemas de busca de caminhos que utilizam mapas com informações de altura e inclinação. O terceiro capítulo detalha a metodologia empregada na construção do conjunto de dados utilizado em treinamento de DNN, na

elaboração das arquiteturas de redes neurais profundas e nos experimentos realizados. O quarto capítulo apresenta e discute os resultados encontrados nos experimentos. O quinto e último capítulo apresenta uma visão geral do trabalho realizado, suas contribuições e trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Este capitulo apresenta os conceitos de busca de caminhos, redes neurais profundas, *ensemble* de redes neurais profundas e, ao final, uma revisão de trabalhos relacionados.

2.1 REVISÃO DE CONCEITOS

2.1.1 Busca de caminhos

Problemas de busca de caminhos (ALGFOOR; SUNAR; KOLIVAND, 2015) tratam da minimização do custo de caminhos usados no deslocamento de agentes em um ambiente. Três aspectos são importantes: a estrutura de abstração do ambiente, a definição de custo do deslocamento e o algoritmo de busca.

O custo de deslocamento está relacionado a natureza do problema abordado. Esse custo pode envolver diferentes características: distância do caminho resultante, tempo de deslocamento, energia necessária para o deslocamento, limitações das capacidades físicas de deslocamento do agente, questões logísticas relevantes ao problema, entre outras.

A topologia do ambiente pode ser modelada a partir do emprego de estruturas regulares, irregulares ou hierárquicas de representação. As regulares discretizam o ambiente utilizando polígonos regulares de mesmo tamanho. Essa estrutura pode não representar com precisão o ambiente ou partes dele, exigindo que refinamentos sejam feitos. Mesmo que a necessidade seja pontual, tais refinamentos da estrutura de representação podem ter um impacto no espaço de memória e tempo de execução de algoritmos de busca. Por outro lado, estruturas irregulares dividem o ambiente utilizando polígonos com formas variadas, possibilitando que a variabilidade das características do ambiente seja melhor representada. Técnicas hierárquicas permitem que o ambiente seja continuamente discretizado, possibilitando uma representação com níveis diferentes de granularidade. O uso de tais níveis de representação permite mitigar o uso de memória das técnicas anteriores, como também pode permitir a otimização das computações realizadas por algoritmos de busca.

Um caminho entre nodos de origem e destino é uma sucessão de nodos adjacentes da estrutura de representação. Os algoritmos de busca podem funcionar com ou sem funções heurísticas compondo as computações de custos de deslocamento. Uma heurística estima o custo entre um nodo qualquer do mapa e um destino, sendo zero quando aplicada ao destino. Essa estimativa auxilia o algoritmo na escolha do próximo nodo a ser analisado

durante o processo de busca de um caminho, indicando o nodo mais promissor em relação a tarefa de encontrar um caminho que leve ao destino. De maneira geral, a função custo é expressa pela Equação 2.1, onde g expressa o custo entre a origem (o) e o nodo sendo avaliado (n), h expressa o custo heurístico entre o nodo sendo avaliado (n) e o destino (d) e f indica o custo total do nodo (n).

$$f(n) = g(o, n) + h(n, d) \tag{2.1}$$

O algoritmo de *Dijkstra* (DIJKSTRA et al., 1959) é um exemplo de algoritmo de busca não heurístico, com a função de custo da forma $f(n) = g(o, n)$. Baseado nele, o algoritmo A^* (HART; NILSSON; RAPHAEL, 1968)(Algoritmo 1) utiliza a Equação 2.1 como função de custo. Comumente as distâncias Euclidiana ou de Manhattan são empregadas na formulação da heurística h. Quando o ambiente possui obstruções ao movimento do agente, nodos podem ser expandidos além do necessário durante um processo de busca de caminho. Entre outros motivos, essas heurísticas tradicionais desconsideram tais obstruções, uma vez que seus valores são a menor distância entre dois nodos de uma estrutura de representação do ambiente.

Quando informações de altura ou inclinação de terrenos virtuais são tratadas em algoritmos de busca, assim como explorado neste TCC, essas informações geralmente são representadas como vetores normais ou como valores discretos na estrutura de representação do ambiente. Em muitas aplicações que consideram características topográficas do terreno nas computações de caminhos, as informações de ângulos obtidas a partir de vetores normais, por exemplo, são utilizados na formulação das parcelas da Equação 2.1. Como empregar essas informações topográficas representadas depende das características do agente e da definição de custo do problema abordado, com o cuidado de não formular uma função heurística h que superestime o custo real, implicando que o A^* venha a computar caminhos que não sejam ótimos.

O pseudocódigo do algoritmo A^*, empregado no desenvolvimento deste TCC, é apresentado no Algoritmo 1. Durante seu funcionamento o algoritmo recebe como entrada um grafo que representa o terreno e um par de nodos, o primeiro sendo a origem do caminho e o segundo sendo o destino. Durante a execução do algoritmo, dois conjuntos são utilizados. O conjunto de nodos abertos (A) recebe inicialmente a origem do caminho (linha 5). O conjunto de nodos fechados (F) começa vazio. Então, o elemento com o menor custo, dado pela função 2.1, é removido do conjunto de nodos abertos (linha 7) e adicionado ao conjunto de fechados (linha 8). Se esse elemento removido for o nodo destino, o algoritmo encerra retornando o caminho encontrado (linhas 9 à 11). Senão, seus vizinhos que não estiver no conjunto de nodos fechados são adicionados ao conjunto de nodos abertos com o custo atualizado (linhas 12 à 24). Se o destino não for encontrado, esse processo se repete até o conjunto de nodos abertos esvaziar. Assim, o algoritmo encerra retornando que não foi encontrado um caminho entre a origem e o destino (linha

26).

A cardinalidade do conjunto de nodos fechados, após a execução do algoritmo, é utilizada como métrica na avaliação do comportamento de algoritmos de busca baseados no A^* e que utilizam outras maneiras de estimar a parcela heurística da função de custo 2.1.

```
Entrada: grafo, origem, destino
1  início
2      origem.g = 0;
3      origem.h = valor heurístico entre origem e destino;
4      origem.parent = none;
5      Insere origem em A;
6      enquanto A ≠ ∅ faça
7          corrente = elemento de A com menor custo f;
8          Adiciona corrente à F;
9          se corrente == destino então
10             retorna caminho;
11         fim
12         para cada adjacente de corrente faça
13             se adjacente ∉ F então
14                 d = corrente.g + peso da aresta (corrente, adjacente);
15                 se d ≤ adjacente.g ou adjacente ∉ A então
16                     adjacente.g = d;
17                     adjacente.h = valor heurístico entre adjacente e destino;
18                     adjacente.parent = corrente;
19                     se adjacente ∉ A então
20                         Insere adjacente em A;
21                     fim
22                 fim
23             fim
24         fim
25     fim
26     retorna caminho não encontrado;
27 fim
```

Algoritmo 1: Pseudo código do algoritmo A^*

2.1.2 Redes neurais profundas

Redes neurais profundas (Deep Neural Networks - DNN) (GOODFELLOW et al., 2016) são algoritmos de aprendizagem de máquina formados pela concatenação de diversas camadas de unidades de processamento elementares, chamadas neurônios. A topologia básica de um DNN é composta por camadas de entrada, intermediárias e de saída, resultando num modelo matemático que possibilita a aprendizagem ou o reconhecimento de padrões em um conjunto de dados. As camadas intermediárias são chamadas de camadas ocultas. Quando a informação flui da camada de entrada para a de saída, inclusive nas ocultas, a rede é denominada *feedforward*. Quando a resposta dos neurônios de uma camada são entradas para os neurônios da camada seguinte, arquitetura empregada neste TCC, tem-se uma DNN totalmente conectada. A Equação 2.2 ilustra o processamento efetuado individualmente pelos neurônios.

$$\phi(\sum_{i=1}^{n} w_i x_i + b) = z \tag{2.2}$$

Os neurônios computam o produto interno entre um vetor de entrada x e um vetor de pesos w, aplicando uma função de ativação ϕ. Funções de ativação não-lineares tornam a rede capaz de aproximar funções não-lineares (CALIN, 2020). As mais comumente utilizadas, por terem derivadas favoráveis, são a sigmoide, tangente hiperbólica e uma aproximação da sigmoide chamada *Rectified Linear Unit* (ReLU). As Equações dessas funções são, respectivamente, 2.3a, 2.3b, 2.3c.

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \text{(2.3a)} \qquad tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad \text{(2.3b)} \qquad R(x) = max(0,x) \quad \text{(2.3c)}$$

O processo de aprendizagem de uma DNN consiste, essencialmente, na busca do conjunto de vetores de peso dos neurônios que a compõem. Esse processo pode ocorrer de três maneiras: supervisionada, não supervisionada e por reforço.

No processo por reforço um agente é exposto à um ambiente desconhecido e suas ações nesse ambiente são recompensadas ou penalizadas, fazendo com que o agente aprenda como atingir um objetivo.

Quando cada instância de um conjunto de dados não possui a saída esperada, temos um aprendizado do tipo não supervisionado. Esse aprendizado busca encontrar padrões desconhecidos nos dados tomados como entradas da rede neural.

Na abordagem supervisionada, e considerando redes *feedforward*, os dados de treinamento são propagados pela DNN e a saída da DNN é comparada com a saída esperada (a qual está presente nos dados) utilizando uma função de erro. Com isso, os pesos dos neurônios são atualizados buscando a minimização da função de erro. O algoritmo de mini-

mização mais comumente utilizado é o método do gradiente descendente. Neste método, o erro é propagado pela DNN no sentido contrário, da camada de saída para a de entrada, e utilizado para o cálculo do gradiente, informação que então é utilizada para atualizar os pesos dos neurônios. Esse processo é chamado de *backpropagation*.

Após o treinamento, a capacidade de generalização da DNN é medida num conjunto diferente de dados de teste (LECUN; BENGIO; HINTON, 2015).

Um problema comum as funções tangente hiperbólica (Equação 2.3b) e sigmoide (Equação 2.3a) é a saturação, onde valores pequenos ou grandes na entrada se ajustam aos limites das imagens das funções. Com a saturação, camadas internas da DNN não recebem informações úteis de gradiente, tornando difícil saber em qual direção os pesos dos neurônios devem ser alterados para melhorar a função de erro, produzindo um aprendizado ineficaz. A utilização de funções lineares por partes, como a ReLU, melhoraram o desempenho de redes *feedforward* (GOODFELLOW et al., 2016). Sendo essa a função de ativação utilizada neste TCC.

Para o método do gradiente descendente funcionar corretamente, uma inicialização dos pesos dos neurônios é importante. A estratégia a ser adotada está relacionada com a função de ativação. Para as funções ReLU a forma de inicialização dos pesos dos neurônios recomendada em (KUMAR, 2017) é a denominada *He*.

Um parâmetro que influencia a atualização dos pesos dos neurônios é a *taxa de aprendizagem*. Uma taxa pequena torna o treinamento lento, enquanto uma muito alta torna o treinamento mais rápido mas dificulta a convergência do processo. Os pesos podem ser atualizados a cada dado ser propagado ou após um lote. Quando apresentados sequencialmente, a DNN está suscetível a ordem de apresentação dos dados e requer uma quantidade menor de memória quando comparado ao processo em lote, o qual torna o aprendizado menos suscetível a ordem de apresentação dos exemplos e permite uma melhor estimativa do erro.

Os critérios de parada do processo de treinamento envolvem: i) o número de vezes que o conjunto de dados de treinamento são apresentados a DNN (épocas). Uma quantidade grande de vezes pode levar a um sobreajuste (*overfitting*) e um número pequeno a um subajuste (*underfitting*) da DNN em relação aos dados apresentados a ela, diminuindo a capacidade de generalização da DNN treinada; ii) a norma do gradiente abaixo de um valor pré-definido, o que não implica necessariamente em boa generalização; iii) e teste de generalização, onde a cada determinado número de épocas o erro é estimado em um conjunto de validação, não utilizado no treinamento. O treinamento é interrompido definitivamente se o erro começar a crescer, momento em que a DNN começaria a perder a capacidade de generalização, e estaria se sobreajustando ao conjunto de dados.

DNN são geralmente aplicadas para problemas de classificação e regressão. Em problemas de classificação, DNN predizem a classe de uma instância. Em problemas de regressão, DNN aproximam uma função, onde a imagem da função é um número.

Algoritmos de busca de caminhos heurísticos podem se beneficiar da capacidade das DNN de aproximar funções em suas computações heurísticas. Um conjunto de informações sobre o comportamento da heurística é apresentado a uma DNN.

A acurácia de uma DNN aplicada em problemas de regressão pode ser expressa pelo erro percentual médio absoluto (MAPE) (MYTTENAERE et al., 2016) dado pela Equação 2.4. Como função de erro, durante o processo de treinamento, o erro médio absoluto (MAE) dado pela Equação 2.5 pode ser utilizado, onde $y_t - \hat{y}_t$ é erro entre o valor previsto pela DNN e o valor esperado, n é a cardinalidade do conjunto utilizado para avaliar a acurácia da DNN treinada. Quanto menor o valor retornado pela função MAPE, melhor é o ajuste do modelo de DNN treinado. MAE e MAPE são as funções empregadas neste TCC.

$$\mathrm{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \qquad (2.4) \qquad\qquad \mathrm{MAE} = \frac{1}{n} \sum_{t=1}^{n} |y_t - \hat{y}_t| \qquad (2.5)$$

Dentre outras arquiteturas de DNN, combinar um número finito de DNN (*ensemble*) (SAGI; ROKACH, 2018), treinadas para a mesma finalidade, em um único preditor, pode produzir um preditor com melhor generalização. Para melhorar a generalização, individualmente as DNN devem possuir boa generalização, sem cometer os mesmos erros no conjunto de dados de testes. Se os componentes do conjunto tem o mesmo padrão de generalização não há ganho efetivo em combiná-los. O ganho na generalização não é o único motivo para combinar preditores. Os padrões em um conjunto de dados podem ser complicados de tal maneira que apenas um preditor não seja capaz de generalizá-los. Outra razão é o *overfitting*.

Formas de obter preditores com padrão de generalização diferentes, com a intenção de combiná-los, incluem a variação dos valores de inicialização da DNN, a topologia, o algoritmo de aprendizagem ou o conjunto de dados de treinamento. Os métodos que variam o conjunto de dados, abordagem adotada neste TCC, produzem melhores resultados quando comparados aos que variam as condições internas dos componentes do conjunto (SHARKEY, 1996).

As técnicas mais frequentemente utilizados para agrupar DNN são o *bootstrap aggregating* (Bagging) (BREIMAN, 1996) e *Boosting*. Para combinar as predições, as principais abordagens para problemas de regressão são a média simples ou ponderada. Para problemas de classificação, a votação por maioria é utilizada.

No algoritmo *Bagging*, utilizado neste TCC, n conjuntos são gerados a partir do conjunto de dados original por amostragem com reposição. Então, esses conjuntos são utilizados para o treinamento de n redes. O resultados das predições são combinados pela média. Os candidatos para compor são os preditores instáveis, onde variações pequenas no conjunto de dados de treinamento provocam grandes variações no preditor (BREIMAN, 1996). Esse processo pode ser paralelizado, pois as redes são treinadas independente-

mente.

No *Boosting*, as redes são treinadas sequencialmente. A primeira é treinada numa amostra do conjunto original. As posteriores são treinadas amostrando o conjunto original. Porém, os mais propensos a serem escolhidos, que possuem maior probabilidade, serão aqueles que a rede anterior cometeu erros durante seus testes. Isso aumenta a probabilidade de padrões mais complicados no conjunto de dados de treinamento serem apreendidos (DRUCKER, 1997). No entanto, isso pode levar a um *overfitting*. *Bagging* tende a diminuir o *overfitting*. As predições das redes são combinadas utilizando média ponderada, onde as redes com maior acurácia têm mais peso.

2.2 TRABALHOS RELACIONADOS

(QIU et al., 2014) agrupou 20 *deep belief network* (DBN) treinadas com números de épocas diferentes em um mesmo conjunto de dados, para problemas de regressão e séries temporais. As predições individuais foram agregadas utilizando um *Support Vector Regression* (SVR), um algoritmo de aprendizagem de máquina para problemas de regressão. Quatro conjuntos de dados sobre séries temporais e três conjuntos de dados sobre regressão foram utilizados para comparar o *ensemble* com algoritmos de aprendizagem de máquina. O agrupamento de DBN proposto é comparado com a) um *Support Vector Regression* (SVR) com kernel *Radial Basis Function* (RBF), b) uma DBN, c) uma rede *feedforward* compostas por 3 camadas, sendo uma delas oculta, com respectivamente, 48, 96 e 1 neurônios, e d) um conjunto de 20 redes *feedforward*, com as mesmas características do item c, treinadas com número de épocas diferentes variando entre 100 e 2000 e com as predições sendo agregadas utilizando um SVR. Os conjuntos de dados sobre séries temporais são linearmente escalados no intervalo $[0, 1]$.

As análises foram feitas usando como métricas a raiz do erro quadrático médio (RMSE) e o erro absoluto percentual médio (Equação 2.4). Os resultados mostraram que o método proposto foi superior aos demais, que métodos de combinam DNN apresentam melhores predições quando comparados a algoritmos individuais, que uma DBN apresenta melhores predições que uma *feedforward* com uma camada oculta, o que pode estar relacionada a falta de profundidade da *feedforward*. Segundo (REN; ZHANG; SUGANTHAN, 2016), esse foi o primeiro trabalho a utilizar *ensemble* para problemas de regressão.

Similar ao que foi apresentado em (QIU et al., 2014), este TCC emprega um conjunto de DNN combinadas para problemas de regressão. Isso é realizado ao aproximar a função heurística utilizada no algoritmo A^* em problemas de busca de caminhos em terrenos com elevação. O trabalho também é direcionado para redes *feedforward* com mais de uma camada oculta. O conjunto de dados utilizado tem cardinalidade maior do que os descritos em (QIU et al., 2014). A forma de treinamento do *ensemble* utiliza técnicas de

amostragem com reposição, que parece não ser o que foi desenvolvido naquele trabalho. Além disso, o conjunto de dados utilizado é construído a partir de uma modelo digital de elevação. Esse modelo não escala os dados para um intervalo $[0, 1]$ e utiliza apena o erro absoluto percentual médio para avaliar a acurácia dos modelos treinados. A topologia da DNN utilizada também é diferente em relação ao que foi apresentado em (QIU et al., 2014), apresentando mais camadas e com maior quantidade de neurônios.

Outros trabalhos descritos na literatura (GANGANATH; CHENG; TSE, 2015) (CHEN; SHI; LIU, 2009) (PüTZ et al., 2016) (CHAGAS, 2019) têm explorado informações topográficas de mapas virtuais na computação de algoritmos de *pathfinding*.

(GANGANATH; CHENG; TSE, 2015) formulou uma heurística para movimentos com menores custos energéticos para robôs móveis em terrenos com elevação, utilizando um algoritmo Z^* (GANGANATH; CHENG; CHI, 2014) baseado no algoritmo A^*. Um grafo foi utilizado para abstrair um modelo digital de elevação de uma área de 1 km^2 de uma região de *canyons*, onde cada vértice representa um ponto do terreno e possui 8 arestas para seus vizinhos. Os pontos do terreno possuem três coordenadas, duas de posição e uma com a elevação.

O ângulo de inclinação entre dois pontos do terreno foi dado pelo arco tangente da declividade entre eles. Esse ângulo é utilizado na formulação do modelo de custo energético para o deslocamento entre vértices no grafo e na formulação da heurística do algoritmo.

A função de custo utilizada é do tipo da Equação 2.1, onde g é dada pela multiplicação da distância Euclidiana em $\mathbb{R}^3$ com a composição das forças que atuam sobre o robô (gravidade e atrito) que estão relacionadas com o ângulo de inclinação entre os vértices que representam o terreno. O robô possui dois limiares analisados na computação dos seus caminhos, o ângulo máximo que consegue subir e o ângulo máximo que consegue descer sem perder o contato com a superfície. Além desses limiares, o custo é considerado infinito e zero, respectivamente.

A heurística h é semelhante a função g. Porém, não é infinita para ângulos acima do limiar de subida, permitindo o robô subir terrenos com inclinação além da sua capacidade, utilizando caminhos em formato de zigue-zague.

O autor comenta que para problemas de encontrar menores caminhos, sem envolver consumo energético, a distância Euclidiana em $\mathbb{R}^3$ pode ser adotada para a obtenção do custo de deslocamento entre os vértices do grafo que modela as características do terreno. Nenhuma afirmação equivalente é realizada para a função heurística.

Similar a (GANGANATH; CHENG; TSE, 2015), um modelo digital de elevação também é utilizado neste TCC, com aproximadamente a mesma área de cobertura. A distância Euclidiana em $\mathbb{R}^3$ é empregada no algoritmo A^*, na função g e na função h, uma vez que não é considerado o custo energético do deslocamento do agente no mapa virtual. Essa configuração do algoritmo é utilizada como base para a comparação com versões

utilizando DNN para aproximar a função heurística h. Um grafo também é utilizado para abstrair o modelo digital de elevação. No entanto, apenas 4 dos 8 possíveis vizinhos são considerados.

(CHEN; SHI; LIU, 2009) apresentou um algoritmo de busca de caminhos em cenas de jogos 3D. A cena é abstraída utilizando um grafo, onde os vértices representam os polígonos e as arestas representam a adjacência de polígonos utilizados na renderização da cena. As arestas possuem ainda os vetores normais dos polígonos adjacentes e as dimensões máximas do objetos que podem transpor os polígonos adjacentes. A heurística utilizada pelo algoritmo de busca utiliza a distância Euclidiana entre dois nodos da estrutura de representação do terreno. As informações de inclinação do terreno, representadas pelos vetores normais, são utilizadas na composição da função $g(o, n)$ (Equação 2.1), onde uma combinação linear desses vetores representa a dificuldade de deslocamento entre os nodos o e n. Neste TCC, um grafo também é utilizado para abstrair, não uma cena de jogos 3D, mas um modelo digital de elevação, onde os vértices deste grafo armazenam a informação de altura do pixel que ele representa. A dificuldade de deslocamento local entre vértices vizinhos, expressa na função g, não utiliza as características do terreno representadas pelos vetores normais, mas utiliza a diferença de altura. Neste caso, é empregada a declividade entre os vértices vizinhos ao calcular a distância entre eles, a qual é computada pelo uso da distancia Euclidiana no $\mathbb{R}^3$.

(PüTZ et al., 2016) reduz o desafio de computar caminhos para robôs móveis em terrenos irregulares ou acidentados a um problema de busca em grafos ao utilizar uma *Navigation Mesh* que combina uma Half Edge Mesh (WIEMANN; LINGEMANN; HERTZBERG, 2013) com dois grafos, resultando numa estrutura chamada *Graph Half Edge Mesh*. A superfície de navegação é representada por uma *mesh* triangular de pontos 3D. Um grafo é composto pelos vértices e pelos lados dos triângulos. O outro grafo é formado pelos triângulos e pela adjacência de triângulos que compõem a *mesh*. A essa estrutura são associados diferentes camadas de custo, referentes as características da superfície representada, tais como vetores normais aos vértices e aos triângulos, ou uma estimativa de distancia baseada na diferença de altura entre os vértices. As diferentes camadas de custo são agregadas no momento de computar caminhos. Os algoritmos de busca de caminhos utilizados foram A^* e *Dijkstra*

De maneira bem mais simplificada, este TCC abstrai um modelo digital de elevação utilizando um grafo e associando às arestas a distancia entre os vértices adjacentes, para computar caminhos utilizando o algoritmo A^* e comparando sua execução utilizando diferentes funções heurísticas.

(CHAGAS, 2019) descreveu um algoritmo de *pathfinding* que produz caminhos suavizados onde agentes têm restrições de inclinação do terreno que consegue transpor durante suas tarefas de movimentação. Para representar terrenos de grandes dimensões, os mapas são modelados por uma estrutura hierárquica chamada *quadtree*, onde os no-

dos folha possuem vetores normais que representam inclinações do terreno em suas 4 direções. Utilizando as informações capturadas nesses vetores, é elaborado um custo de travessia de fronteira entre nodos adjacentes. O ângulo dos vetores normais é utilizado para tornar a função g do algoritmo proposto, que tem função de custo de acordo com a Equação 2.1, diretamente proporcional ao ângulo. Desta forma, maiores inclinações do terreno representado são mais custosas de transpor pelos agentes. Essa abordagem é semelhante a da função g em (GANGANATH; CHENG; TSE, 2015). Contudo, a abordagem é voltada para o deslocamento de agentes em ambientes de simulação. A função heurística usada pelos algoritmos de *pathfinding* é a distância Euclidiana em $\mathbb{R}^2$.

Em geral, as informações sobre altura e inclinação são armazenadas nos modelos do terreno como valores ou vetores normais, sendo que a partir da informação de altura do terreno em cada nodo é possível obter a informação sobre inclinação entre nodos da estrutura. Esse é o caso de (GANGANATH; CHENG; TSE, 2015), onde a altura é usada para calcular distâncias no $\mathbb{R}^3$ e para obter a inclinação, fundamental para determinar o custo do deslocamento do robô. Essas informações topográficas do terreno acabam sendo convertidas em um valor de custo. Esse custo é utilizado, geralmente, para compor o custo de travessia entre dois nodos da estrutura de representação do terreno virtual.

Este TCC também trata de mapas virtuais de grandes dimensões como em (CHAGAS, 2019), embora não trate da suavização de caminhos. O TCC também adota um *grid* regular de 100 pixels quadrados como estrutura de representação para descrever um modelo digital de elevação, tal como (GANGANATH; CHENG; TSE, 2015). Nessa estrutura, nodos representam cada pixel do modelo de elevação. Informações de inclinação são calculadas da mesma forma que em (CHAGAS, 2019), sem considerar aspectos de eficiência energética. A distância Euclidiana no $\mathbb{R}^3$ é usada para computar g e h durante o processo de busca de caminhos do A^*. Os custos desses caminhos compõem o conjunto de dados utilizado no treinamento de arquiteturas de DNN, utilizadas para aproximar a função heurística usada no processo de busca de caminhos.

Outros trabalhos descritos na literatura (AGOSTINELLI et al., 2019) (ARIKI; NARIHIRA, 2019) (JINDAL et al., 2017) (LI et al., 2016) (TAKAHASHI et al., 2019) (WANG et al., 2019) (DOEBBER, 2019) (SOUZA, 2021) têm investigado o uso de DNN no apoio a resolução de diferentes problemas de busca.

Neste contexto, (AGOSTINELLI et al., 2019) propuseram o *DeepCubeA*, que utiliza técnicas de aprendizagem por reforço profundo em funções heurísticas para solução do cubo de Rubik e para outros quebra-cabeças que possuem grandes espaços de busca e um estado objetivo. Os autores sugerem que o algoritmo pode ser aplicado para outros problemas, além de solucionar quebra-cabeças, que possuem grandes espaços de busca e poucos estados objetivos. O algoritmo proposto utiliza uma DNN para aprender uma função de custo para atingir o objetivo a partir do estado objetivo. Então, essa função aprendida é utilizada como heurística do algoritmo de busca *weighted* A^*. A heurística

resultante não é admissível. No entanto, o algoritmo encontrou o caminho ótimo na maioria dos casos. Este TCC também aproxima uma função heurística utilizando o aprendizado de uma DNN, mas utiliza a abordagem supervisionada e emprega esse aprendizado no algoritmo A^*, em espaços de busca menores.

(ARIKI; NARIHIRA, 2019) aplicaram uma arquitetura convolucional de DNN para aproximar funções heurísticas de planejadores de caminhos. A DNN é treinada em 7 tipos de ambientes diferentes de imagens de mapas 2D, com diferentes tipos de obstáculos, cada um com 800 mapas de treinamento e 100 mapas de teste com dimensões de 201 pixel quadrados. Após o treinamento, é apresentada à DNN uma imagem de um mapa contendo os obstáculos do ambiente e uma posição objetivo. Como resultado, é gerada uma imagem, onde cada pixel representa a distância até a posição objetivo, gerando um mapa heurístico. Esse mapa é consultado durante a busca de caminhos pelo planejador. Os treinamentos da arquitetura, para cada ambiente virtual, levam cerca de 10h para 10 mil épocas. Como resultado, o planejador de caminhos usando um mapa heurístico produzido por uma arquitetura convolucional expandiu uma menor quantidade de nodos durante uma busca quando comparado a outros algoritmos, incluindo o A^* utilizando a distância Euclidiana como função heurística. Similar a (ARIKI; NARIHIRA, 2019), um mapa heurístico também é utilizado neste TCC. Contudo, ele é obtido a partir de uma DNN treinada com um conjunto de caminhos gerados pelo A^* utilizando a distância Euclidiana como função heurística em mapas que representam informações topográficas do terreno.

(JINDAL et al., 2017) propuseram uma arquitetura chamada *Spatio-Temporal Neural Network* (ST-NN) para predizer a distância e o tempo de viagem de táxi entre duas coordenadas geográficas conforme o horário do dia. A arquitetura combina duas DNN diferentes de três camadas. Uma é responsável por predizer a distância entre dois pontos e a outra é responsável por predizer o tempo. Em particular, a DNN responsável pela predição do tempo recebe como entrada as saídas da última camada da DNN responsável por predizer a distância, juntamente com as informações do horário do dia. O conjunto de dados utilizado no processo de treinamento da arquitetura é oriundo de trilhas de GPS embarcados em táxis da cidade de Nova York. Neste TCC, DNN também são aplicadas para problemas de regressão e para predizer distâncias a partir de coordenadas do mapa virtual utilizado. Contudo, o uso das predições é diferente visto que tais predições são aplicadas de duas formas: como substituta ou como um fator multiplicador da parcela heurística do algoritmo A^*.

Um algoritmo semelhante ao A^* foi proposto em (LI et al., 2016). Esse algoritmo utilizou uma DNN como um coeficiente multiplicador da parcela heurística da função de custo do A^* (Equação 2.1) explora na busca de caminhos em mapas representados por *grids* regulares. A DNN apresentada tem um neurônio na camada de saída, característica de DNN aplicadas em problemas de regressão. A DNN é treinada num conjunto de mapas bidimensionais produzidos randomicamente. Após o treinamento, dado um mapa, ela

gera um coeficiente que representa a dificuldade de transitar pelo mapa. As heurísticas consideradas conjuntamente com a DNN são a distância Euclidiana e da de Manhattan. Este TCC apresenta uma abordagem semelhante a (LI et al., 2016). Porém, o valor do fator multiplicador da parcela heurística do algoritmo de busca depende do nodo da estrutura de representação do terreno que está sendo avaliado no momento da execução da busca. Além disso, a parcela heurística é substituída pelo valor predito pela DNN, onde o treinamento da DNN é realizado a partir do custo de caminhos computados entre pares de pontos em um mapa. O algoritmo utilizado também é o A^*. Porém, esse algoritmo avalia apenas a distância Euclidiana como função heurística.

(TAKAHASHI et al., 2019) aborda uma alternativa ao uso de funções heurísticas tradicionais ao propor a utilização da rede neural U-Net (RONNEBERGER; FISCHER; BROX, 2015) em heurísticas de problemas de *pathfinding*. Essa DNN extrai informações de imagens de mapas para o aprendizado de heurísticas utilizadas no planejamento de rotas de robôs móveis, com o objetivo de reduzir o número de nodos expandidos durante o processo de busca. O algoritmo de *Dijkstra* é utilizado para a construção da heurística ideal, utilizada durante o treinamento da DNN. O A^* é um dos algoritmos utilizados para avaliar a heurística apreendida pela DNN. Os resultados mostram que a método proposto reduziu o custo de busca em ambientes discretos e contínuos. Este TCC tem o mesmo objetivo, mas emprega uma arquitetura *feedforward* para aprender uma heurística a partir da heurística ideal, dada pela computação de caminhos em um mapa com informações topográficas. Esses caminhos são computados com o A^* utilizando a distancia Euclidiana como função heurística.

(WANG et al., 2019) apresentaram um modelo chamado *Neuralized A-Star based personalized route Recommendation* (NASR) para recomendação de rotas personalizadas. O modelo é baseado no A^*; no entanto, a abordagem de utilização de DNN é um pouco diferente do que visto até agora. Uma rede neural recorrente é empregada para aproximar a primeira parcela da Equação 2.1.

(DOEBBER, 2019) descreveu a aplicação de uma arquitetura *feedforward* totalmente conectada e inspirada em (JINDAL et al., 2017) nas computações de *pathfinding* em mapas de labirintos bidimensionais. A parcela heurística do A^* (Equação 2.1) é substituída pela DNN. Para isso, um mapa heurístico (ARIKI; NARIHIRA, 2019) é computado e empregado nas computações de custo. Os resultados obtidos para quatro mapas de labirintos de tamanhos diferentes mostraram que o A^* com uma heurística gerada por uma DNN expande uma quantidade menor de nodos durante o processo de busca de caminhos do que o A^* utilizando a distância Euclidiana ou a de Manhattan como heurísticas. As arquiteturas de DNN utilizadas têm uma topologia em formato de losango, onde a quantidade de neurônios cresce a partir da entrada até a camada central e então decresce na mesma proporção. As funções de ativação utilizadas são a ReLU (Equação 2.3c) nas camadas ocultas, e a sigmoide (Equação 2.3a) na camada de saída.

(SOUZA, 2021) utiliza técnicas de *pathfinding* hierárquico na expansão do trabalho de (DOEBBER, 2019) de forma a computar caminhos em mapas de dimensões maiores. Isso é possível por meio do treinamento de DNN em porções menores do mapa as quais são organizadas como clusters de dimensões menores. Para cada cluster, uma DNN é treinada. Então, a busca de caminhos acontece primeiro entre os clusters e, depois, apenas entre os clusters que pertencem ao caminhos encontrado no primeiro passo de busca.

Neste TCC, a arquitetura de DNN utilizada é a *feedforward* assim como explorado em (SOUZA, 2021), embora essas redes tenham sido exploradas no aprendizado de caminhos obtidos em mapas com informações topográficas. A topologia da DNN utilizada é semelhante, diferindo na quantidade de neurônios na camada de entrada, na primeira camada oculta e na função de ativação da camada de saída. A forma de construção do conjunto de dados utilizado é semelhante ao que é descrito em (SOUZA, 2021). Contudo, o algoritmo A^* é usado e uma amostragem estratificada do mapa é usada. Além disso, este TCC avalia a utilização da DNN como um fator multiplicador da função heurística, além de substituí-la. Em contraste, este TCC não explora técnicas hierárquicas na busca de caminhos. Outra diferença é que a pesquisa desenvolvida neste TCC analisa o emprego de *ensemble* de DNN para aproximar uma função a função heurística usada pelo algoritmo de busca de caminhos.

Tabela 2.1 – Comparativo entre trabalhos relacionados

Ambiente	Arquitetura DNN	Algoritmo de *pathfinding*	Referência
mapa com topografia	-	Z^*	(GANGANATH; CHENG; TSE, 2015)
cena de jogo 3D	-	D^*	(CHEN; SHI; LIU, 2009)
cena 3D	-	A^* e *Dijkstra*	(PüTZ et al., 2016)
mapa com topografia	-	HPT^*, $Theta^*$, $HPATheta^*$	(CHAGAS, 2019)
-	*Feedforward*	*weighted* A^*	(AGOSTINELLI et al., 2019)
2D grid	Convolucional	A^*, *Backward Dijkstra*	(ARIKI; NARIHIRA, 2019)
2D	Spatio-Temporal Neural Network	-	(JINDAL et al., 2017)
2D	*Feedforward*	ANN^*	(LI et al., 2016)
2D	Convolucional	A^*	(TAKAHASHI et al., 2019)
2D	Rede Neural Recorrente	NASR	(WANG et al., 2019)
2D grid	*Feedforward*	A^*	(DOEBBER, 2019)
2D grid	*Feedforward*	A^*, *Dijkstra*	(SOUZA, 2021)
mapa com topografia	*Feedforward*	A^*	Este TCC

Fonte: Autor.

Em resumo, apesar desses trabalhos terem empregado diferentes arquiteturas de DNN, nenhum deles investiga a construção de heurísticas utilizando um *ensemble* de DNN. Mais ainda, eles não analisaram como usar tais DNN na otimização da busca de caminhos em terrenos contendo informações topográficas. No entanto, algumas ideias desenvolvidas na pesquisa deste TCC são similares ao que foi apresentado nos trabalhos relacionados, como o mapa heurístico utilizado por (ARIKI; NARIHIRA, 2019) e (DOEBBER, 2019) e a adaptação do fator multiplicador da parcela heurística do A^* apresentado em (LI et al., 2016). A Tabela 2.1 mostra um comparativo entre os trabalhos relacionados e este TCC.

3 METODOLOGIA

Este capitulo apresenta os processos de obtenção dos mapas, conjuntos de dados e arquiteturas de redes neurais profundas utilizados nos experimentos.

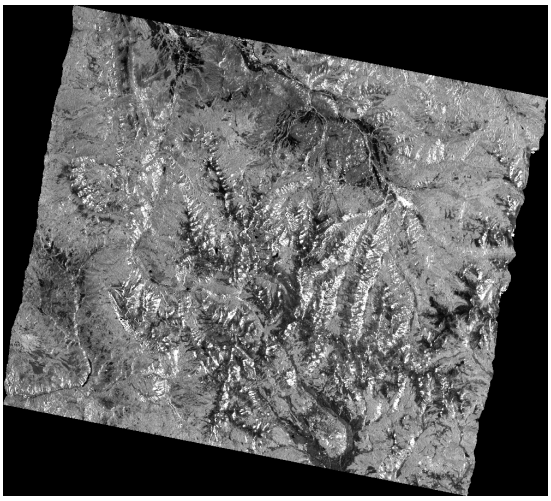
3.1 MAPAS UTILIZADOS

Os mapas contendo informações de relevo utilizados neste TCC foram produzidos a partir de modelo digital de elevação (MDE) utilizado no processo de correção radiométrica (ASF, 2015) de imagens obtidas por sensoriamento remoto com radar de abertura sintética (SAR). Tais imagens são disponibilizadas pelo *Alaska Satellite Facility* (DAAC, 2014). O MDE é uma representação digital da superfície da Terra conforme o modelo matemático (datum) utilizado para projetar a Terra em um plano. O MDE é uma imagem no formato GeoTIFF, onde a cor de cada pixel representa a elevação do ponto na superfície da Terra que ele representa. De acordo com esse modelo, atributos dos terrenos usados, tais como, elevação e inclinação, foram obtidos.

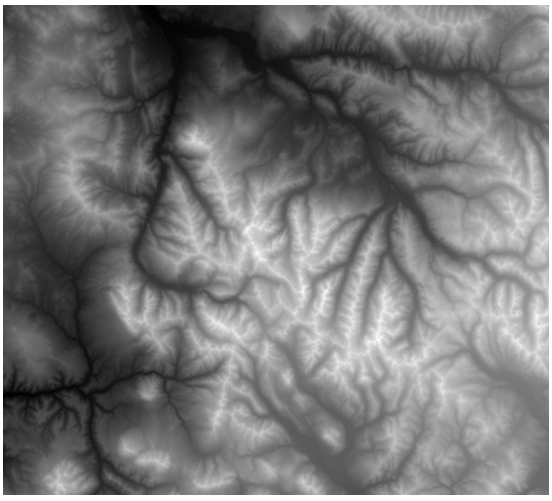
O MDE escolhido foi utilizado para a correção da imagem ALPSRP277832830 (ALPSRP277832830, 2011). A imagem pode ser vista na Figura 3.1a e o MDE correspondente na Figura 3.1b. Ele possui dimensões de 6288x5656 pixels e resolução espacial de 12,5m. No entanto, é importante observar que o MDE é uma reamostragem dos dados do *Shuttle Radar Topography Mission* (SRTM), que possuiu resolução espacial de 30m.

Figura 3.1 – Imagem ALPSRP277832830

(a) Corrigida



(b) MDE utilizado na correção



Fonte: Advanced Land Observing Satellite

Essa imagem foi escolhida por apresentar relevo acidentado, permitindo que se possa bloquear áreas a partir de um limiar de elevação, enquanto outras permanecem livres e apresentado relevo não plano. A forma de bloqueio empregada considera intransitável o pixel que possuir altitude superior a um determinado limiar.

Para computar caminhos utilizando o MDE, ele foi abstraído por um grafo não-dirigido. Cada pixel do MDE é associado a um vértice do grafo e possui um identificador associado a sua posição no MDE. Pixels vizinhos no MDE possuem vértices adjacentes no grafo. O peso atribuído a aresta entre vértices adjacentes é a distância Euclidiana dos respectivos pixels no MDE. Esse cálculo de distância considera a diferença de altitude entre os vértices. Quanto maior essa diferença, maior a distância e, também, a declividade entre os vértices.

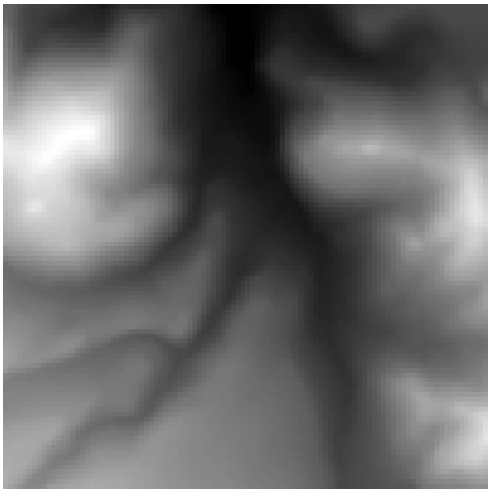
Este TCC investiga computação de caminhos em mapas que apresentam grande espaço de busca. Para isso, o tamanho dos mapas foi definido em 100x100 pixels. A partir dessa definição foram selecionados recortes do MDE, com diferentes percentuais de áreas livres e bloqueadas.

O processo de seleção dos recortes dos mapas seguiu os seguintes passos: (i) recortar o MDE em quadrados de 100x100 pixels, utilizando a ferramenta *gdal_translate* e script escrito em Shell Script; (ii) abstrair o recorte de MDE; (iii) variar o limiar de elevação e verificar os percentuais de área livre e área bloqueada resultantes; e (iv) verificar se o limiar tornou o grafo desconexo.

Além de computar caminhos em mapas com grandes espaços de busca, este TCC avalia o comportamento de DNN em mapas com quantidades diferentes de relevo bloqueado. Para isso, formam selecionados 3 diferentes recortes de 100x100 pixels com, aproximadamente, 30, 50 e 70% de área livre para computação de caminhos. Assim, foram escolhidos os recortes que possuíam algum limiar de elevação que possibilitasse algum desses percentuais de área livre. Além disso, foram selecionados os recortes onde o limiar de boqueio de nodos não tornava o grafo resultante desconexo. Para verificar se o grafo era desconexo foi utilizado uma busca em largura e verificado se a quantidade de vértices visitados era igual a quantidade de vértices no grafo.

Figura 3.2 – Recorte 1

(a) Recorte MDE



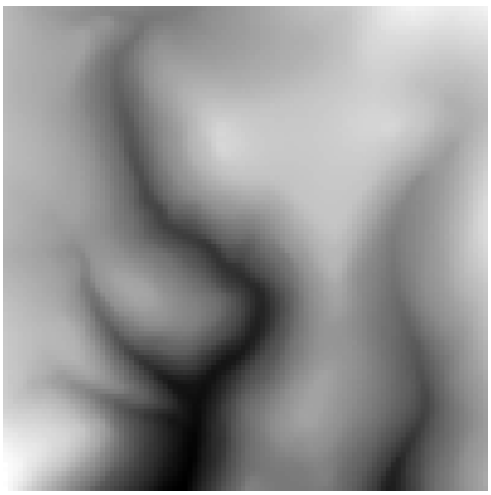
(b) 30% de área livre

Fonte: Advanced Land Observing Satellite

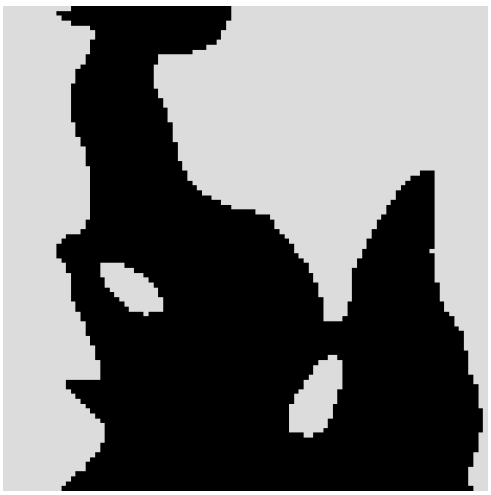
Os recortes selecionados podem ser vistos nas Figuras 3.2a, 3.3a e 3.4a, onde cores mais claras representam maiores altitudes. As áreas livres e bloqueadas para computação de caminhos podem ser vistas nas Figuras 3.2b, 3.3b e 3.4b, onde a cor cinza representa a região bloqueada para o deslocamento de um agente.

Figura 3.3 – Recorte 2

(a) Recorte MDE



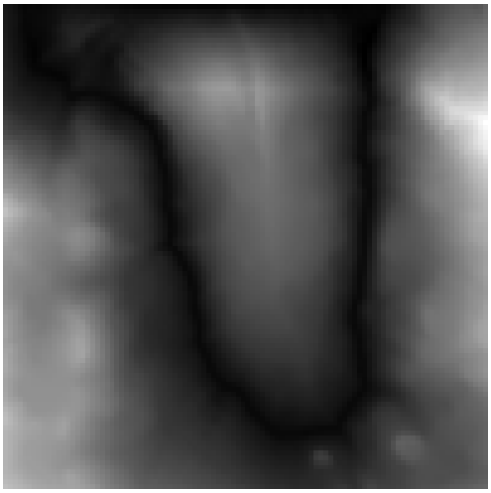
(b) 50% de área livre



Fonte: Advanced Land Observing Satellite

Figura 3.4 – Recorte 3

(a) Recorte MDE



(b) 70% de área livre

Fonte: Advanced Land Observing Satellite

3.2 CONJUNTO DE DADOS

A partir dos recortes selecionados foram construídos três conjuntos de dados, um para cada recorte. Para gerar a base de dados foram selecionados aleatoriamente 50% dos pixels livres para computação de caminhos. Isso foi feito em cada recorte. Esse processo dividiu o recorte em quatro quadrantes e selecionou 25% dos pixels em cada quadrante. Isso buscou reduzir o tamanho do conjunto de dados. O tamanho do conjunto de dados é dado pelo combinação da quantidade de pixels selecionados tomados dois a dois. Tabela 3.2 resume as características dos mapas utilizados nos experimentos e na construção dos dados de treinamento de DNN. O ambiente de execução é apresentado na Tabela 3.1.

Tabela 3.1 – Ambiente de execução dos experimentos

CPU	Intel(R) Core i7-6700k
CPU Núcleos	4 (8)
CPU Frequência	4.00 GHz
GPU	NVIDIA GeForce GTX 1050 Ti
GPU Arquitetura	Pascal
GPU Núcleos	768
GPU Memória	4GB GDDR5
Memória RAM	2x8GB DDR4-2800
Sistema Operacional	Debian stable

Fonte: Autor.

Tabela 3.2 – Dados para treinamento de DNN

	Recorte 1	Recorte 2	Recorte 3
Abstração	grid regular	grid regular	grid regular
Tamanho grid	10^4	10^4	10^4
Pixel bloqueados	6.703 (67%)	5.009 (50%)	2.910 (30%)
Pixel livres	3.297 (33%)	4.991 (50%)	7.090 (70%)
Limiar altitude	2.504 m	2.788 m	3.506 m
Altitude mínima	2.463 m	2.679 m	3.345 m
Altitude máxima	2.601 m	2.861 m	3.614 m
Amostra	1.648 (50%)	2.495 (50%)	3.545 (50%)
Conjunto de dados	$\binom{1.648}{2} = 1.357.128$	$\binom{2.495}{2} = 3.111.265$	$\binom{3.545}{2} = 6.281.740$
Tempo de geração	5,5h	17h	42h
Treino/Validação/Teste	70/15/15	70/15/15	70/15/15

Fonte: Autor.

Com a seleção dos pixels, foram computados caminhos para todas as combinações de pontos tomadas dois a dois (indicando início e fim de caminhos a serem computados). Foi empregando o algoritmo A^* (Algoritmo 1) utilizando a distância Euclidiana no $\mathbb{R}^3$ como função heurística h e também na função de custo g (Equação 2.1), uma vez que o valor de g é igual ao peso da aresta entre vértices adjacentes. Para cada caminho computado, as coordenadas da origem, as coordenadas do destino e o custo do caminho computado entre a origem e o destino foram armazenados em um arquivo de texto. A composição de uma linha no arquivo texto é a seguinte:

<coordenadas origem>,<coordenadas destino>,<custo do caminho entre origem e destino>.

3.3 ARQUITETURA E TREINAMENTO DE DNN

Com os conjuntos de dados formados, passou-se para o processo de escolha da arquitetura de DNN empregada nos experimentos. Todas as implementações utilizaram a versão 3.73 da linguagem Python e a versão 2.4.0 das bibliotecas TensorFlow e Keras. Como modelos iniciais de DNN, os modelos 1, 3 e 5 testados em (DOEBBER, 2019) foram utilizados. Tais modelos possuem quatro entradas, uma saída que utiliza como função de ativação a função sigmóide (Equação 2.3a) e um número variado de camadas ocultas com quantidade de neurônios simétrica em relação a camada oculta central. A quantidade de neurônios das camadas nos modelos de DNN empregados por (DOEBBER, 2019) e utilizado neste TCC foram:

Modelo 1: [4,20,100,20,1]

Modelo 3: [4,20,100,200,300,200,100,20,1]

Modelo 5: [4,20,100,200,300,400,500,400,300,200,100,20,1]

Antes do treinamento desses modelos de DNN, a quantidade de neurônios nas camadas de entrada foi alterada para 6, uma vez que os pontos de origem e destino nos conjuntos de dados possuem três coordenadas. Além disso, a função de ativação da camada de saída foi alterada para uma função linear, uma vez que os conjuntos de dados construídos e utilizados nos experimentos não foram normalizados.

Para a realização de testes com esses modelos, os conjuntos de dados foram separados em três conjuntos, selecionados utilizando sementes aleatórias: treinamento, validação e teste. O conjunto de treinamento possuindo 70% do conjunto de dados e os conjuntos de validação e teste possuindo 15% cada um.

Este TCC utiliza o método do gradiente descendente para minimizar a função de erro, utilizada no treinamento de DNN. O erro pode ser estimado a cada instância do conjunto de dados ser propagada pela DNN, após todas as instâncias terem sido propagadas pela DNN ou após um lote de instâncias terem sido propagadas pela DNN. Estimar o erro frequentemente, com tamanhos de lote pequenos, pode evitar mínimos locais durante o processo de minimização da função de erro, mas tende a tornar o processo de treinamento mais lento. Estimar o erro com menos frequência, com tamanhos de lotes maiores, torna a estimativa de erro mais estável, o processo de treinamento mais rápido, porém exige mais recursos computacionais e torna o processo mais suscetível a mínimos locais. Este TCC utiliza tamanhos de lotes maiores que um e menores que a cardinalidade do conjunto de treinamento da DNN, buscando um equilíbrio entre o tempo de treinamento e recurso computacionais utilizados. Geralmente os tamanhos dos lotes empregados são potências de 2, quando unidades de processamento gráfico são utilizadas durante o treinamento de DNN. Os tamanhos dos lotes testados foram: 512, 1024, 2048, 4096 e 8192.

Como critérios de parada de treinamento, foram determinados empiricamente, o limite de 2 mil épocas ou 100 épocas sem diminuição do erro no conjunto de validação,

dado pela função MAPE (Equação 2.4). Esses limites foram estabelecidos para limitar o tempo máximo de treinamento, uma vez que em testes preliminares realizados foi possível perceber que cada época durava no máximo 24 segundos no ambiente de execução apresentado na Tabela 3.1. À cada época a configuração da DNN era salva em disco. Parado o treinamento, por alguns dos critérios de paradas estipulados, as configurações com menor valor MAPE (Equação 2.4) para cada modelo, tamanho de lote e recorte tiveram a acurácia medida no conjunto de testes, utilizando a função MAPE (Equação 2.4).

Em todos os casos, o modelo com a quantidade de neurônios por camada de [6,20,100,200,300,400,500,400,300,200,100,20,1], com tamanho de lote maior ou igual a 2.048 obteve as melhores acurácias. Assim esse modelo foi escolhido e usado no desenvolvimento dos experimentos realizados neste trabalho. Uma última avaliação foi feita em relação a quantidade de neurônios na primeira camada oculta. Modelos com uma quantidade superior apresentaram melhores resultados. Então, foi adotada como quantidade de neurônios na primeira camada oculta a ordem de grandeza da camada oculta central. Assim, o modelo de DNN empregado nos experimentos possuiu a quantidade de neurônios por camada de: [6,500,100,200,300,400,500,400,300,200,100,20,1], com função de ativação linear na camada de saída e função de ativação ReLU nas camadas ocultas.

Tabela 3.3 – Resultados treinamento DNN

	Acurácia	Horas de treinamento	Épocas	Tamanho do lote
Recorte 1 DNN	98,8%	1	751	4096
Recorte 1 ENN 1	98,0%	0,70	631	2048
Recorte 1 ENN 2	97,6%	0,37	336	2048
Recorte 1 ENN 3	98,5%	0,84	758	2048
Recorte 1 ENN 4	97,5%	0,40	365	2048
Recorte 1 ENN 5	97,3%	0,61	548	2048
Recorte 2 DNN	97,7%	1,17	703	8192
Recorte 2 ENN 1	97,8%	1,22	734	8192
Recorte 2 ENN 2	97,6%	1	620	8192
Recorte 2 ENN 3	97,7%	1	601	8192
Recorte 2 ENN 4	98,1%	1,21	728	8192
Recorte 2 ENN 5	97,3%	0,7	440	8192
Recorte 3 DNN	99,4%	3,7	556	4096
Recorte 3 ENN 1	98,8%	1,54	231	4096
Recorte 3 ENN 2	98,6%	1	152	4096
Recorte 3 ENN 3	98,2%	1,1	165	4096
Recorte 3 ENN 4	98,6%	0,87	131	4096
Recorte 3 ENN 5	98,7%	1,34	201	4096

Fonte: Autor.

Após a escolha do modelo de DNN, o modelo selecionado foi agrupado utilizando a técnica de *bootstrap aggregating* (Bagging) (BREIMAN, 1996). Cinco modelos iguais foram treinados em 5 conjuntos de dados diferentes obtidos do conjunto de dados construído a partir de um recorte. Primeiro, o conjunto de dados foi dividido em dois conjuntos. Um conjunto de teste com 15% dos dados e um segundo conjunto com 85% dos dados. A partir desse segundo conjunto foram gerados 5 novos conjuntos por amostragem com reposição. Esses conjuntos foram divididos em conjuntos de treinamento e validação, com 82% para o conjunto de treinamento e 18% para o conjunto de validação. Dessa maneira, os conjuntos de treinamento, validação e teste possuíam, respectivamente, 70, 15 e 15% dos dados do conjuntos de dados original. Para medir a acurácia dos 5 modelos combinados, as predições resultantes de cada modelo foram combinadas por média simples.

As características das DNN selecionadas estão na Tabela 3.3. As acurácias obtidas foram superiores à $97,3\%$, e os tempos de treinamento foram inferiores a $1,6$ hora.

Assim, como analisado neste TCC, o aprendizado da DNN pode ser utilizado em substituição da função heurística originalmente empregada ou pode ser usado como um fator que determina a influência da parcela heurística h no custo total do caminho f (Equação 2.1). As DNN selecionadas serão empregadas no algoritmo A^* (Algoritmo 1) como funções heurísticas substituindo a função h ou atuando como um fator multiplicador da distância Euclidiana, da forma mostrada nas Equações 3.1, 3.2, 3.3, 3.4 e 3.5, onde d é a distância Euclidiana entre dois pontos, dnn a predição da distância entre dois pontos dada por uma DNN e enn (Emsemble NN) indica a predição da distância entre dois pontos dada pelo conjunto de 5 DNN combinadas (com arquiteturas de rede idênticas). Nas Equações 3.3 e 3.5 enn é dividido pelo maior custo dos caminhos computados no conjunto de dados para pertencer ao intervalo $[0, 1]$, uma vez que, nas saídas das DNN, não foram utilizadas funções de ativação com imagem sendo o intervalo $[0, 1]$. Sendo utilizada uma função linear. O algoritmo A^* adotado neste TCC considera a topografia do terreno, que foi abstraída pelo grafo, no momento em que se foi definido o peso da aresta entre vértices adjacentes como sendo a distância Euclidiana no $\mathbb{R}^3$, entre eles.

$$h_{r3}(n, m) = d(n, m) \tag{3.1}$$

$$h_{dnn\_sub}(n, m) = dnn(n, m) \tag{3.2}$$

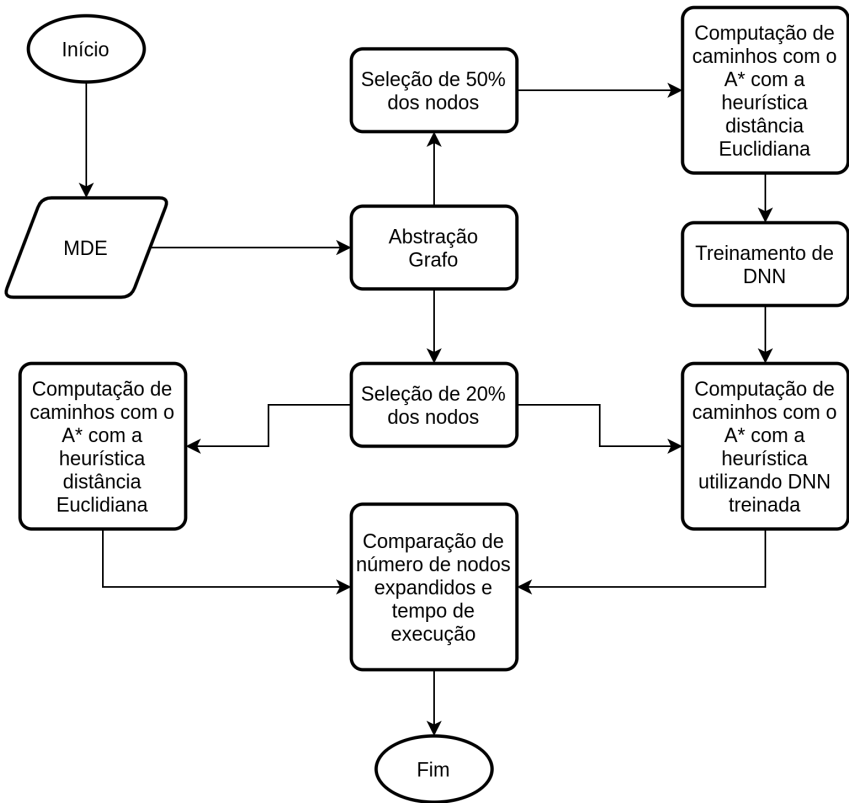
$$h_{dnn\_fator}(n, m) = (1 + dnn(n, d))d(n, d) \tag{3.3}$$

$$h_{enn\_sub}(n, m) = enn(n, m) \tag{3.4}$$

$$h_{enn\_fator}(n, m) = (1 + enn(n, d))d(n, d) \tag{3.5}$$

Antes do processo de busca de caminhos, são calculadas as predições da DNN que está sendo utilizada para todas as combinações, tomadas duas a duas, dos vértices livres no grafo que abstrai o MDE. O tempo total de cálculo das predições é dividido pelo número de predições. Esse valor é considerado o tempo por predição e somado ao tempo de execução do algoritmo A^*. O tempo de treinamento não é considerado na analise dos resultados experimentais.

Figura 3.5 – Fluxograma descrevendo a metodologia explorada na pesquisa desenvolvida neste TCC.



Fonte: Autor.

A Figura 3.5 ilustra a metodologia utilizada neste TCC. Um modelo digital de elevação é abstraído por um grafo. São selecionados 50% dos nodos ou vértices do grafo e computados caminhos utilizando o algoritmo A^* (Algoritmo 1) com a distância Euclidiana no $\mathbb{R}^3$ como função heurística. Os caminhos computados são utilizados no processo de treinamento, validação e teste de acurácia de DNN. Por outro lado, em 20% dos nodos do grafo são computados caminhos com o algoritmo A^* utilizando a distância Euclidiana no $\mathbb{R}^3$ e, também, utilizando o resultado do processo de treinamento de DNN na composição da função heurística. Por fim, para os caminhos computados nos 20% de nodos, são comparados tempo de execução e quantidade de nodos expandidos.

4 EXPERIMENTOS E RESULTADOS

Os experimentos desenvolvidos nesse TCC tem por objetivo avaliar se DNN é capaz de aprender a geometria e as regiões bloqueadas para o deslocamento de um agente, em um terreno contendo informações topográficas. A DNN é empregada como um aproximador da função heurística h (Equação 2.1) do algoritmo A^* (Algoritmo 1) ou atuando como um fator multiplicador da distância Euclidiana. As hipóteses são que essa forma de emprego das DNN podem reduzir o tempo de execução e a quantidade de nodos expandidos durante a computação de caminhos com o algoritmo A^* entre dois pontos do mapa.

Para cada um dos Recortes de mapas utilizados nos testes desenvolvidos neste TCC (Figuras 3.2, 3.3 e 3.4), foram computados caminhos com cinco heurísticas diferentes (Equações 3.1, 3.2, 3.3, 3.4 e 3.5), no mesmo ambiente de execução onde os conjuntos de dados de treinamento foram obtidos (Tabela 3.1). A heurística utilizada como base para as avaliações foi a distância Euclidiana no $\mathbb{R}^3$.

Para analisar o comportamento do Algoritmo 1 em relação as opções de heurísticas, foram computados caminhos em 40% dos 50% de nodos que não foram utilizados na construção do conjunto de dados de cada recorte. Ou seja, foram computados caminhos entre 20% da totalidade de vértices do grafo que abstrai o recorte. A totalidade de caminhos computados é o número de combinações tomadas duas a duas desses 20% de nodos. Esses valores são apresentados na Tabela 4.1. Os tamanhos dos conjuntos de dados são proporcionais à área livre para um agente se movimentar no terreno.

Tabela 4.1 – Dados para os experimentos

	Recorte 1	Recorte 2	Recorte 3
Amostra	661 (20%)	998 (20%)	1418 (20%)
Caminhos computados	$\binom{661}{2} = 218.130$	$\binom{998}{2} = 497.503$	$\binom{1418}{2} = 1.004.653$

Fonte: Autor.

Para realizar as análises comparativas propostas neste TCC, as seguintes métricas foram utilizadas: tempo de execução do algoritmo de *pathfinding*, a quantidade de nodos avaliados durante a busca de caminhos e custo/distância dos caminhos computados.

Para analisar estatisticamente os resultados, foram utilizados modelos de regressão linear generalizada (MCCULLAGH; NELDER, 1989), tal como (DOEBBER, 2019) (CHAGAS, 2019) (SOUZA, 2021). Esses modelos são uma generalização dos modelos de regressão linear que permitem variáveis resposta com distribuição diferente da normal. As variáveis resposta tempo de execução ou quantidade de nodos expandidos são contínuas e tem uma relação assimétrica com a variável explicativa distância de caminho computado, além de possuírem valores positivos. A distribuição *gamma* é comumente empregada nesse caso. Assim, utilizando como função de ligação a função logarítmica, o modelo de

regressão linear generalizada utilizado na análise dos resultados foi:

$$log(\mu) = \beta_0 + \beta_1 X + \beta_2 D_1 + \beta_3 D_2 + \beta_4 D_3 + \beta_5 D_4$$

Onde, μ é a média da distribuição da variável resposta, X é a variável explicativa e D_i são as variáveis *dummy* utilizadas para analisar cinco heurísticas diferentes num mesmo modelo de regressão. Para cada heurística, elas assumem uma configuração demonstrada na Tabela 4.2.

Tabela 4.2 – Configuração das variáveis *dummy*

Variáveis *dummy*	Função heurística
$D_1 = D_2 = D_3 = D_4 = 0$	h_{r3}
$D_2 = D_3 = D_4 = 0$, $D_1 = 1$	h_{dnn_sub}
$D_1 = D_3 = D_4 = 0$, $D_2 = 1$	h_{dnn_fator}
$D_1 = D_2 = D_4 = 0$, $D_3 = 1$	h_{enn_sub}
$D_1 = D_2 = D_3 = 0$, $D_4 = 1$	h_{enn_fator}

Fonte: Autor.

Para desenvolver a análise estatística, foi definido um nível de significância de $\alpha = 0,1\%$. As hipóteses nula $\mathbb{H}_0$ e alternativa $\mathbb{H}_1$ foram definidas como $\beta_i = 0$ ou $\beta_i \neq 0$, para $i = 0, 1, 2$. Se $\alpha >$ valor-p, então $\mathbb{H}_0$ não é rejeitada. $\beta_1 > 0$ significa que a variável resposta é diretamente proporcional a variável explicativa. $\beta_{i+1} > 0$, $\beta_{i+1} = 0$ e $\beta_{i+1} < 0$ para $i = 1, 2, 3, 4$, significam, respectivamente, que a função heurística associada a D_i é mais eficiente, equivalente e menos eficiente que o que a heurística utilizada como base para as comparações.

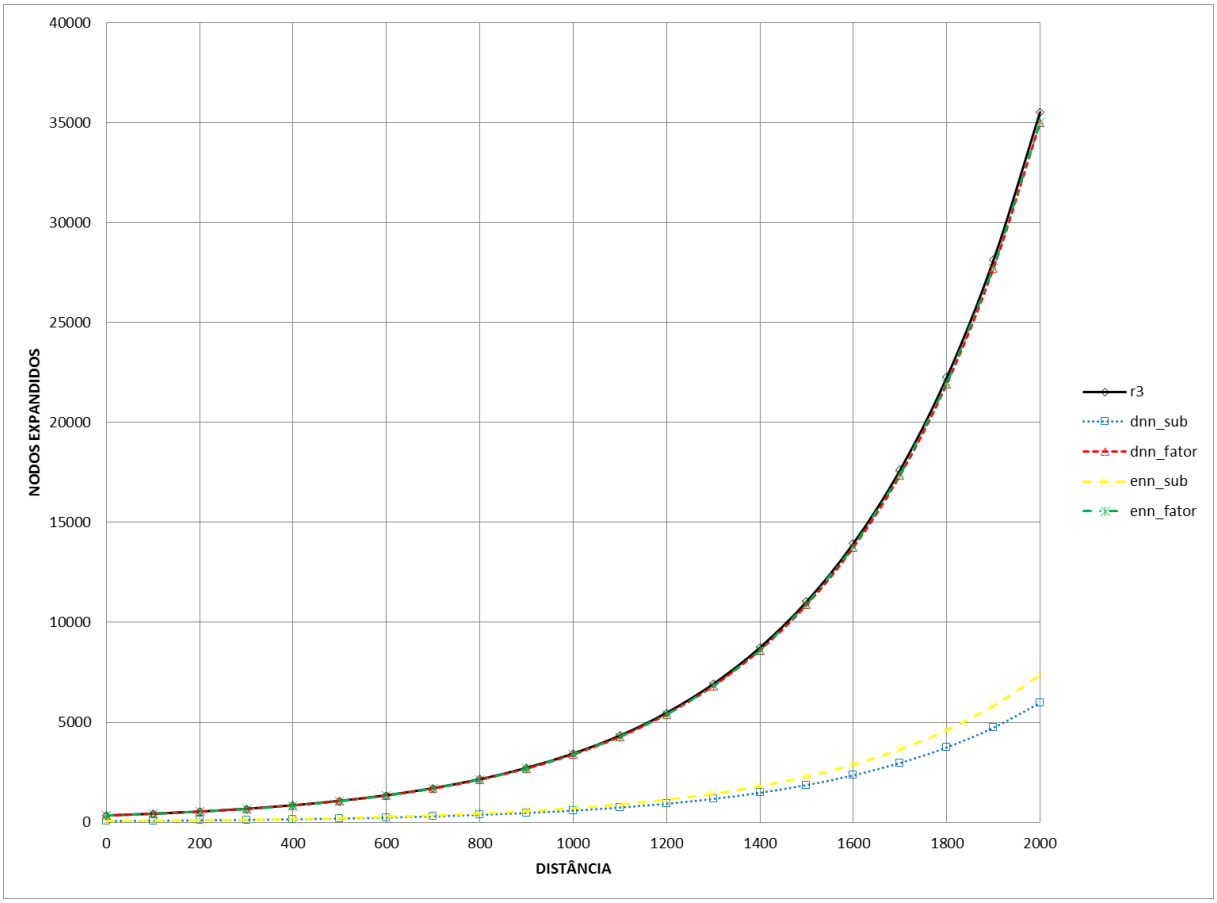
Os resultados obtidos na computação de caminhos para as heurísticas testadas podem ser vistos nas Tabelas 4.4, 4.5, 4.7, 4.8, 4.10 e 4.11. Para todos os casos, valor-p < 0.001, abaixo do nível de significância definido. Assim, os resultados obtidos nos testes desenvolvidos foram estatisticamente significativos.

Tabela 4.3 – Recorte 1: percentual de tempo e expansão de nodos

Heurística	Expansão	Tempo
h_{dnn_sub}	-83,1%	-59,6%
h_{dnn_fator}	-1,45%	+267,7%
h_{enn_sub}	-79,3%	-59,2%
h_{enn_fator}	-1,47%	+244,6%

Fonte: Autor.

Figura 4.1 – Recorte 1: nodos expandidos x distância



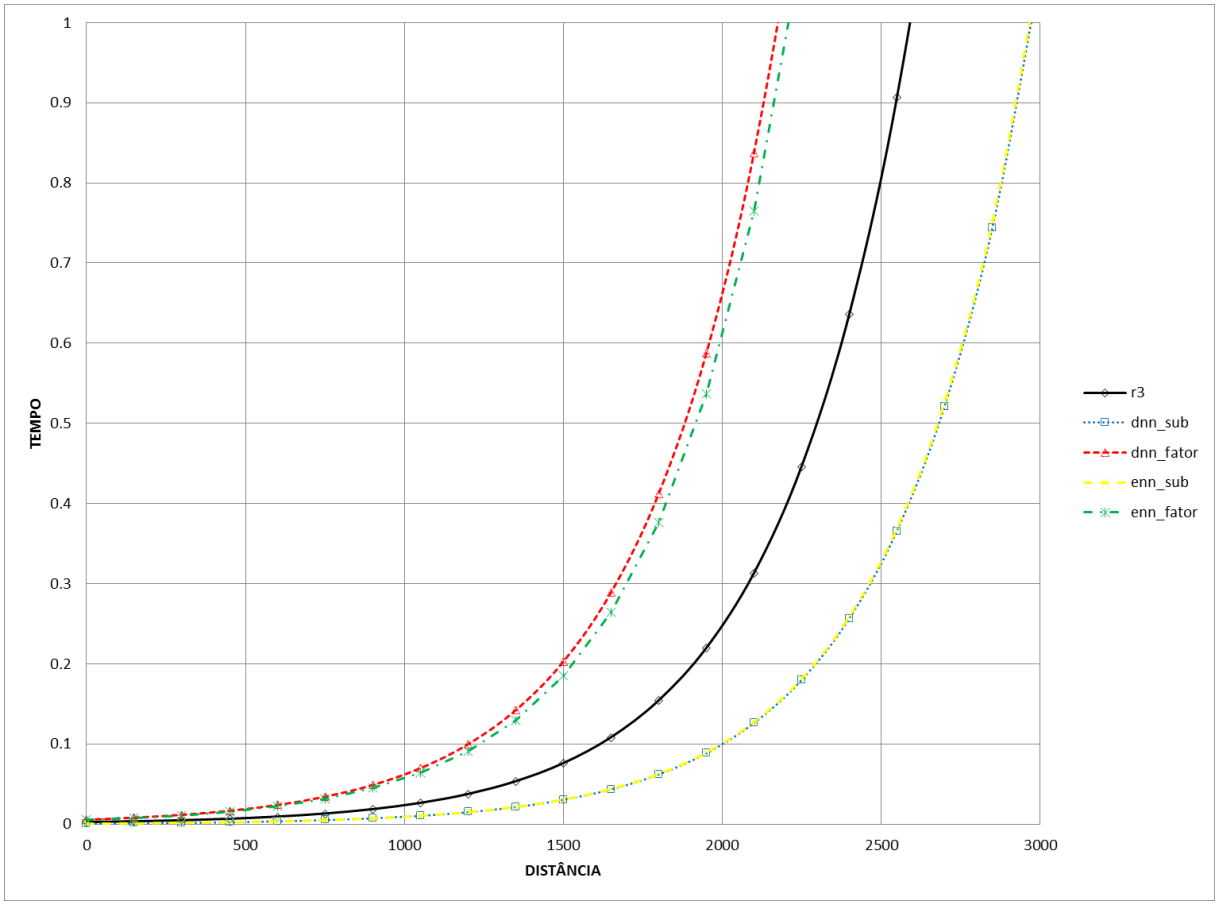
Fonte: Autor.

Tabela 4.4 – Recorte 1: resultados estatísticos, nodos visitados x distância

	Estimativa	Desvio padrão	Valor t	$P(>\|t\|)$
Intercepto	5.802e+00	1.840e-03	3154.226	< 2e-16
h_{r3}	2.338e-03	1.557e-06	1501.397	< 2e-16
h_{dnn_sub}	-1.780e+00	2.003e-03	-888.846	< 2e-16
h_{dnn_fator}	-1.470e-02	1.990e-03	-7.385	1.53e-13
h_{enn_sub}	-1.575e+00	2.000e-03	-787.493	< 2e-16
h_{enn_fator}	-1.484e-02	1.990e-03	-7.457	8.88e-14

Fonte: Autor.

Figura 4.2 – Recorte 1: tempo de execução x distância



Fonte: Autor.

Tabela 4.5 – Recorte 1: resultados estatísticos, tempo de execução x distância

	Estimativa	Desvio padrão	Valor t	$P(>\|t\|)$
Intercepto	-6.129e+00	1.783e-03	-3436.6	<2e-16
h_{r3}	2.365e-03	1.527e-06	1548.5	<2e-16
h_{dnn_sub}	-9.075e-01	1.953e-03	-464.7	<2e-16
h_{dnn_fator}	9.848e-01	1.934e-03	509.1	<2e-16
h_{enn_sub}	-8.979e-01	1.951e-03	-460.3	<2e-16
h_{enn_fator}	8.945e-01	1.934e-03	462.4	<2e-16

Fonte: Autor.

O Recorte 1 (Figura 3.2) apresenta 30% de área livre para a circulação de um agente. Os valores negativos para as estimativas dos parâmetros do modelo de regressão, quando considerada como variável explicativa a quantidade de nodos expandidos (Tabela 4.4), mostram que computar caminhos utilizando a distância Euclidiana expande maior quantidade de nodos em relação as demais heurísticas testadas. Isso indica que a partir dos dados de caminhos pré-computados no Recorte 1 e usados no treinamento, a

DNN aprendeu a geometria e as regiões bloqueadas deste recorte, sendo capaz de generalizar esse aprendizado. No entanto, a ordem de grandeza das estimativas são diferentes. Por exemplo, a estimativa de h_{dnn_sub} é -1.78 enquanto a de h_{dnn_fator} é -0.0147 . Assim, o algoritmo A^* utilizando h_{dnn_sub} como função heurística expande a menor quantidade de nodos em relação ao A^* com h_{r3}: $-83,1\%$. Os valores para as demais heurísticas são apresentados na Tabela 4.3. Observando esses resultados experimentais, os métodos que substituem a heurística pelo valor da predição da DNN se saem melhor quando comparados aos métodos que utilizam a predição da DNN como componente de um fator multiplicador da distância Euclidiana. A Figura 4.1 exibe as curvas para cada função heurística.

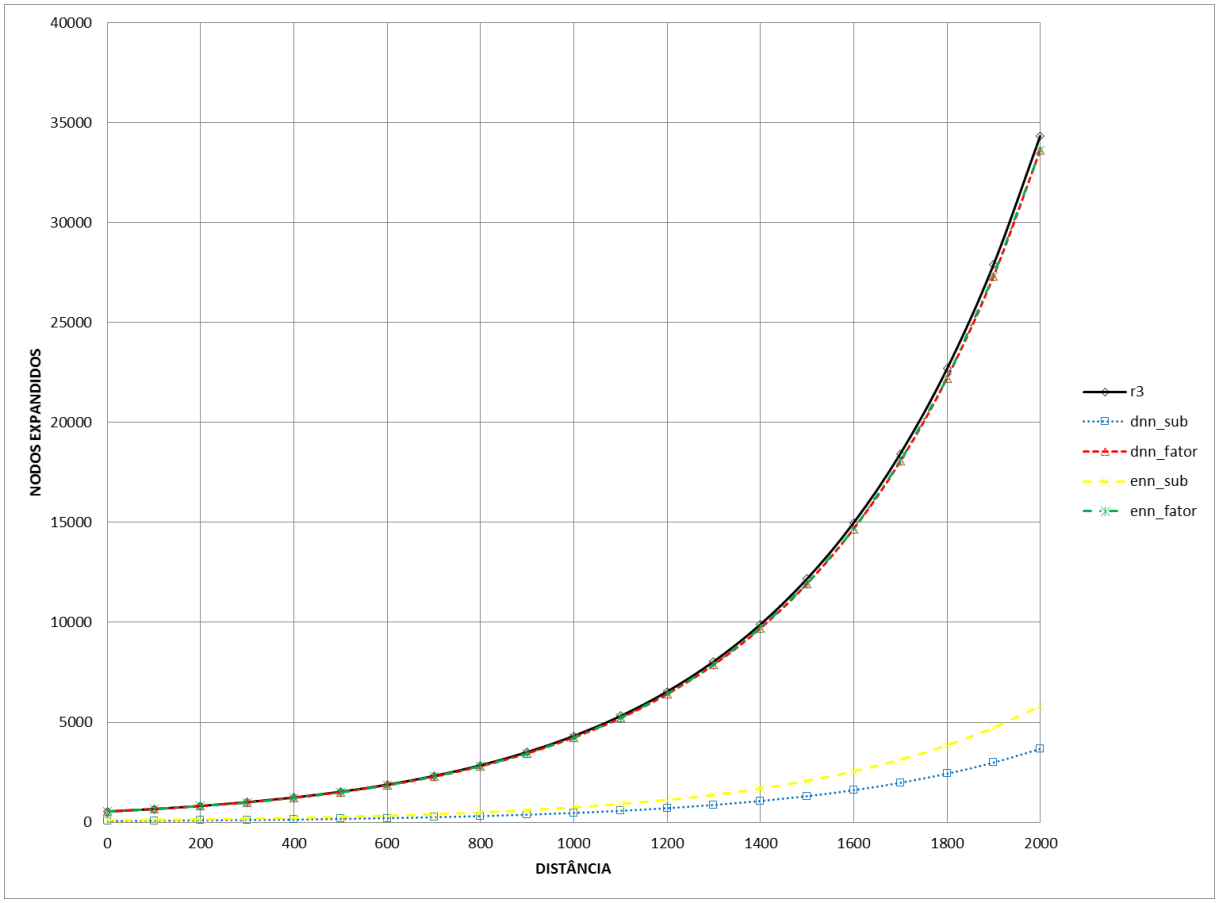
Quando a variável explicativa considerada é o tempo de execução, as estimativas do modelo de regressão para o Recorte 1 (Figura 3.2) mostram que o algoritmo A^* utilizando h_{dnn_fator} ou h_{enn_fator} computa caminhos onde o tempo de busca é mais de duas vezes maior do que utilizando h_{r3} como função heurística. Possivelmente, isso se deve ao fato de serem introduzidos o tempo de predição da DNN (muito maior que uma simples computação de distância entre dois pontos) e a quantidade de nodos expandidos serem próximas do A^* com a heurística h_{r3}. As estimativas mostram ainda que utilizando h_{dnn_sub} ou h_{enn_sub}, o A^* computa caminhos em menor tempo de busca. Tabela 4.3 exibe os percentuais de execução para cada heurística. É importante observar que a diferença entre h_{dnn_sub} e h_{enn_sub} está na primeira casa decimal, apesar de h_{enn_sub} possuir cinco DNN combinadas, o que eleva o tempo de predição quando comparado ao tempo de predição de uma DNN. Além disso, h_{enn_sub} expande mais nodos durante o processo de busca de caminhos. Figura 4.2 exibe as curvas para cada função heurística.

Tabela 4.6 – Recorte 2: percentual de tempo e expansão de nodos

Heurística	Expansão	Tempo
h_{dnn_sub}	-89,3%	-75,7%
h_{dnn_fator}	-2,11%	+272,37%
h_{enn_sub}	-80%	-67,4%
h_{enn_fator}	-2,11%	+265,4%

Fonte: Autor.

Figura 4.3 – Recorte 2: nodos expandidos x distância



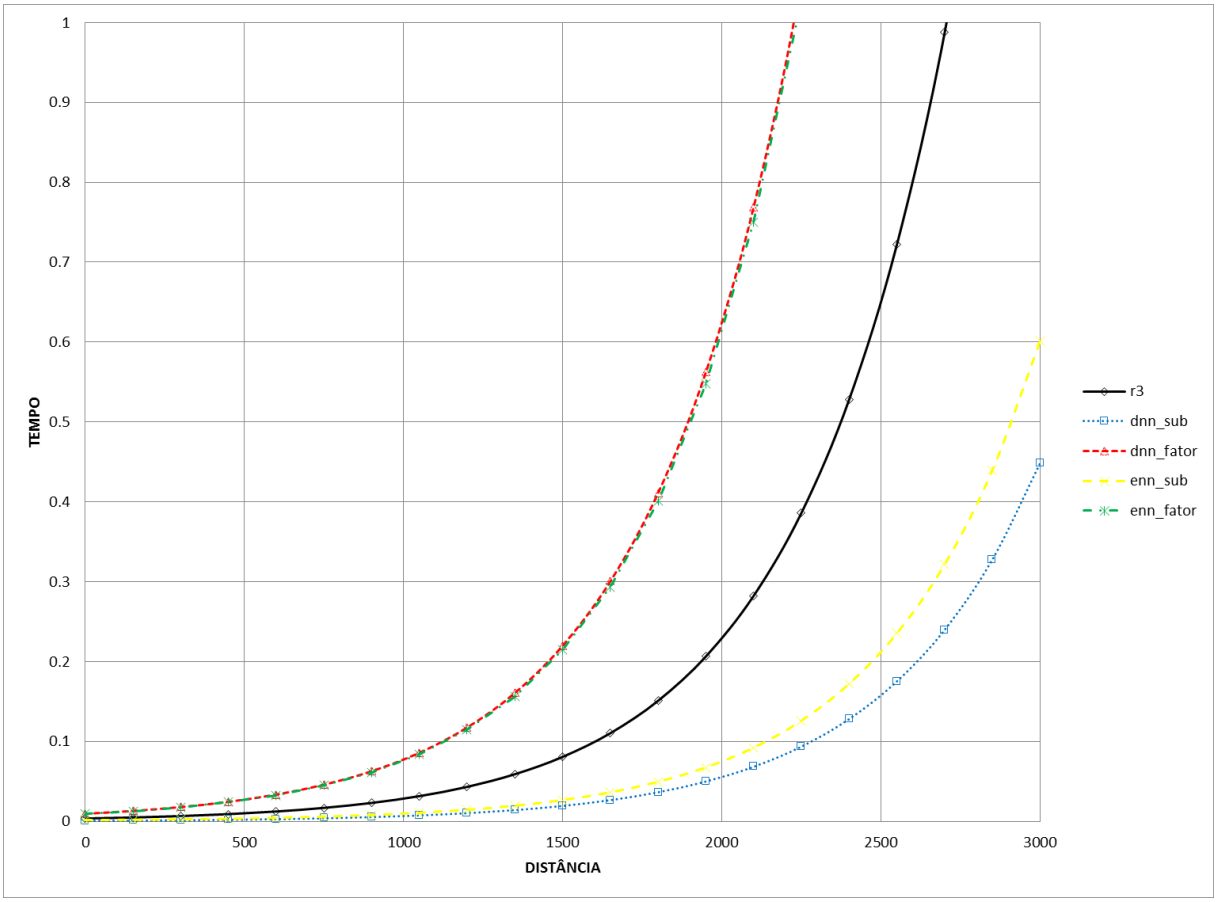
Fonte: Autor.

Tabela 4.7 – Recorte 2: resultados estatísticos, nodos visitados x distância

	Estimativa	Desvio padrão	Valor t	$P(>\|t\|)$
Intercepto	6.300e+00	1.175e-03	5362.66	<2e-16
h_{r3}	2.072e-03	8.949e-07	2315.03	<2e-16
h_{dnn_sub}	-2.235e+00	1.302e-03	-1715.78	<2e-16
h_{dnn_fator}	-2.132e-02	1.299e-03	-16.41	<2e-16
h_{enn_sub}	-1.776e+00	1.304e-03	-1361.76	<2e-16
h_{enn_fator}	-2.130e-02	1.299e-03	-16.40	<2e-16

Fonte: Autor.

Figura 4.4 – Recorte 2: tempo de execução x distância



Fonte: Autor.

Tabela 4.8 – Recorte 2: resultados estatísticos, tempo de execução x distância

	Estimativa	Desvio padrão	Valor t	$P(>\|t\|)$
Intercepto	-5.647e+00	1.132e-03	-4988.1	<2e-16
h_{r3}	2.087e-03	8.740e-07	2387.6	<2e-16
h_{dnn_sub}	-1.416e+00	1.257e-03	-1126.6	<2e-16
h_{dnn_fator}	1.002e+00	1.252e-03	800.5	<2e-16
h_{enn_sub}	-1.122e+00	1.259e-03	-891.4	<2e-16
h_{enn_fator}	9.761e-01	1.252e-03	779.7	<2e-16

Fonte: Autor.

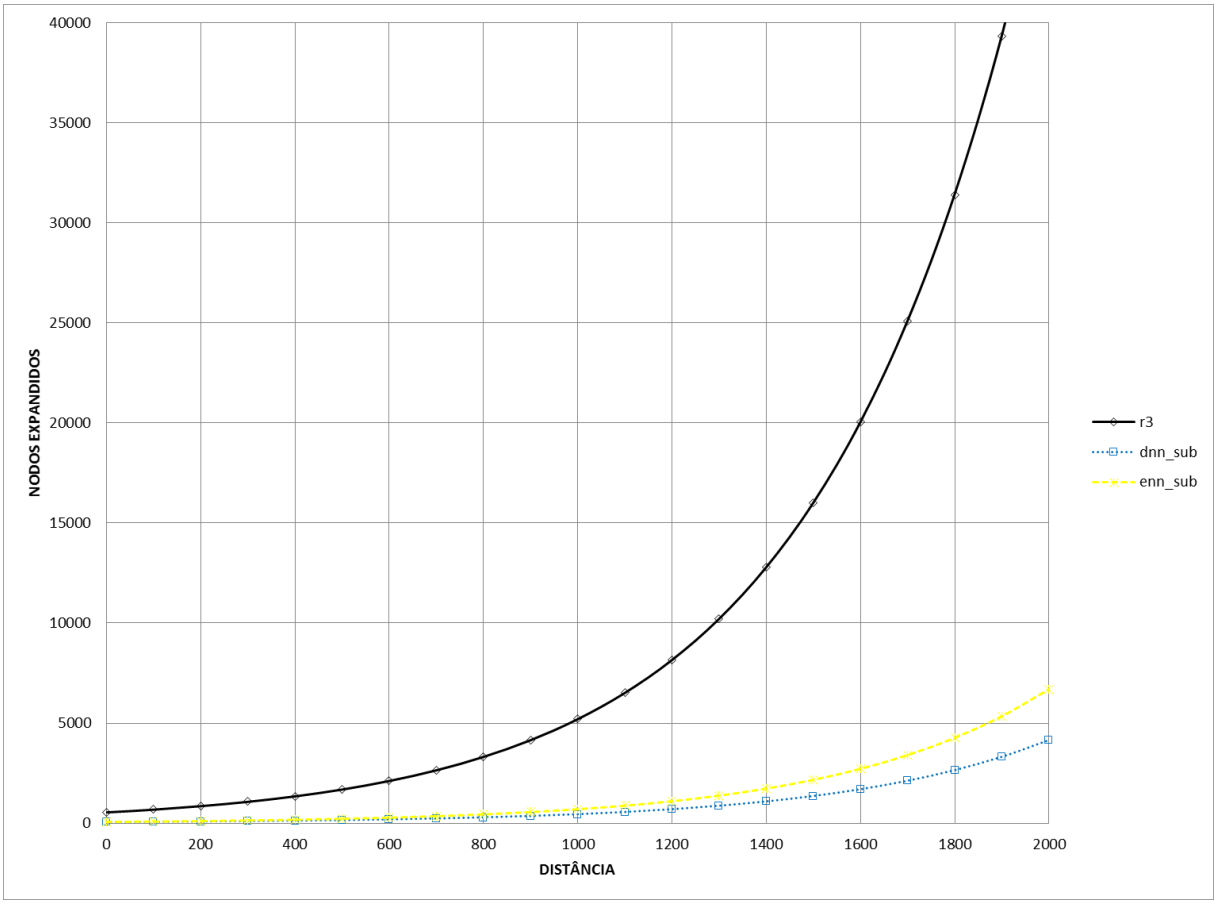
Em contraste com o Recorte 1, o Recorte 2 (Figura 3.3) possui uma área livre para navegação de um agente na ordem de 50% deste mapa. Os resultados para as estimativas dos parâmetros do modelo de regressão, disponíveis nas Tabelas 4.4 e 4.5, mostram que o aumento da área livre não alterou o comportamento entre as heurísticas testadas. As heurísticas h_{dnn_sub} e h_{enn_sub} melhoraram seu desempenho em relação ao Recorte 1, expandindo menos nodos e em um tempo de busca menor. A Tabela 4.6 resume os percentuais. As Figuras 4.3 e 4.4 apresentam os gráficos representando os resultados obtidos neste conjunto de testes.

Tabela 4.9 – Recorte 3: percentual de tempo e expansão de nodos

Heurística	Expansão	Tempo
h_{dnn_sub}	-91,6%	-80,3%
h_{dnn_fator}	-	-
h_{enn_sub}	-86,4%	-69,6%
h_{enn_fator}	-	-

Fonte: Autor.

Figura 4.5 – Recorte 3: nodos expandidos x distância



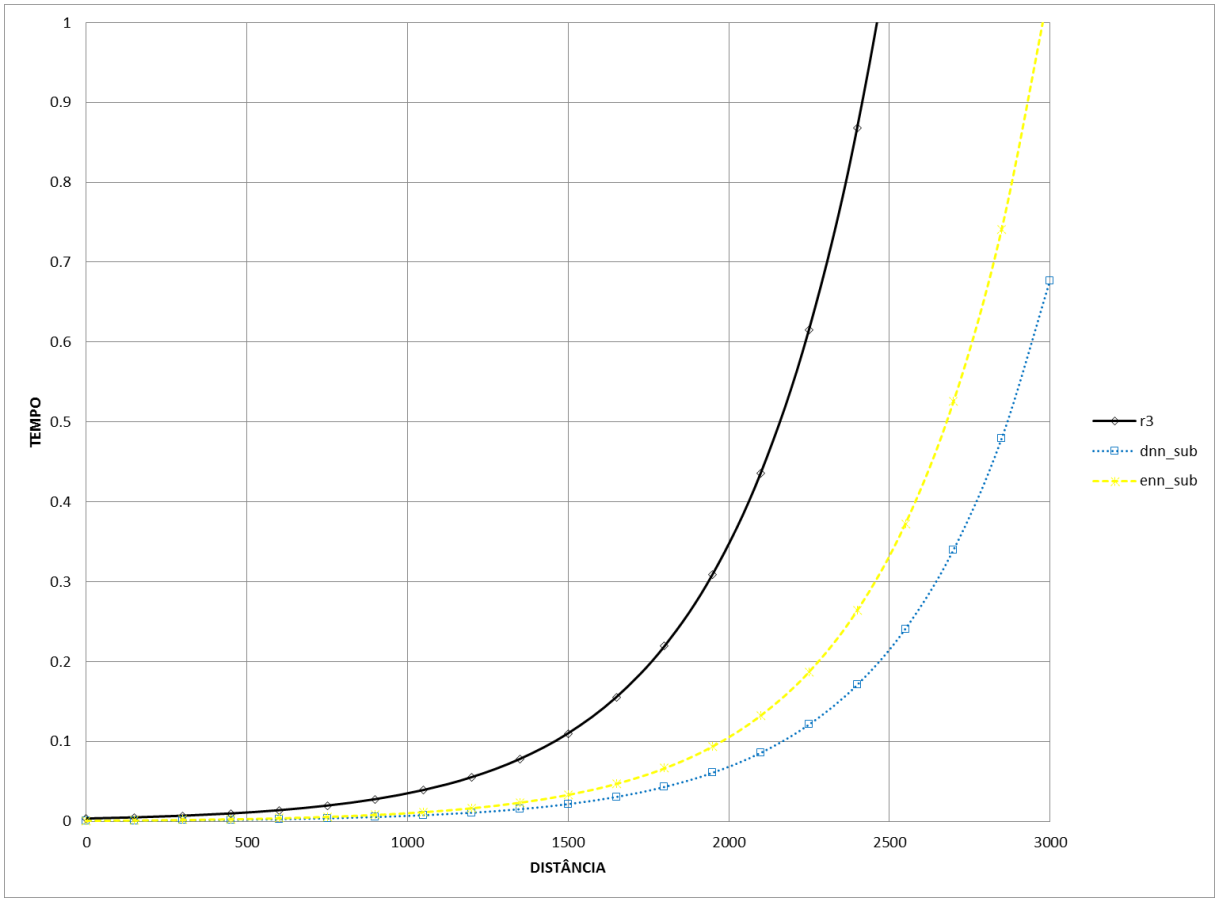
Fonte: Autor.

Tabela 4.10 – Recorte 3: resultados estatísticos, nodos visitados x distância

	Estimativa	Desvio padrão	Valor t	$P(>\|t\|)$
Intercepto	6.310e+00	1.003e-03	6288	<2e-16
h_{r3}	2.247e-03	8.112e-07	2770	<2e-16
h_{dnn_sub}	-2.475e+00	9.253e-04	-2675	<2e-16
h_{enn_sub}	-1.998e+00	9.260e-04	-2157	<2e-16

Fonte: Autor.

Figura 4.6 – Recorte 3: tempo de execução x distância



Fonte: Autor.

Tabela 4.11 – Recorte 3: resultados estatísticos, tempo de execução x distância

	Estimativa	Desvio padrão	Valor t	$P(>\|t\|)$
Intercepto	-5.647e+00	9.713e-04	-5814	<2e-16
h_{r3}	2.294e-03	8.003e-07	2867	<2e-16
h_{dnn_sub}	-1.626e+00	8.954e-04	-1817	<2e-16
h_{enn_sub}	-1.190e+00	8.978e-04	-1325	<2e-16

Fonte: Autor.

O Recorte 3 (Figura 3.4), por sua vez, apresenta uma área livre para deslocamentos da ordem de 70% da área total deste mapa. A quantidade de caminhos computados nesse caso foi de $1.004.653$. O tempo de execução do algoritmo A^* utilizando como função heurística h_{r3} foi de cerca de 8 horas. Com isso, os testes realizados com as heurísticas h_{dnn_fator} e h_{enn_fator} foram interrompidos quando o tempo de execução atingiu 150% do tempo de computação com a heurística h_{r3}. Os tempos de execução podem ser vistos na Tabela 4.12. Para essas duas heurísticas, foi assumido que teriam o mesmo comportamento percebido nos Recortes 1 e 2. Ou seja, que o tempo de execução da busca de

caminhos seria muito superior ao tempo de execução com a heurística h_{r3} e que a quantidade de nodos expandidos seria semelhante à quantidade de nodos expandidos obtida com a utilização da heurística h_{r3}. Com isso, as Tabelas 4.10 e 4.11 e as Figuras 4.5 e 4.11 apresentam resultados apenas para as heurísticas h_{dnn_sub} e h_{enn_sub}. Em resumo, os resultados experimentais para o Recorte 3 apresentam o mesmo comportamento obtidos nos Recortes 1 e 2. Isso sugere que o aumento da área livre para circulação de um agente no mapa não interfere no aprendizado dos modelos de DNN testados na construção de heurísticas. No entanto, mais testes poderiam ser conduzidos para analisar o impacto no treinamento das DNN pelo uso de mapas com diferentes distribuições de obstáculos nas várias áreas dos mapas.

Tabela 4.12 – Tempo de execução experimentos

	Recorte 1	Recorte 2	Recorte 3
h_{r3}	3090s	9988s	30274s
h_{dnn_sub}	1749s	3124s	7598s
h_{enn_sub}	1728s	4321s	12313s
h_{dnn_fator}	8341s	27548s	interrompido
h_{enn_fator}	7617s	26833s	interrompido

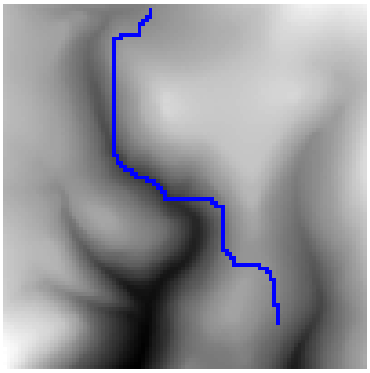
Fonte: Autor.

Como uma maneira de avaliar o quanto os caminhos produzidos pelo $A^\star$ utilizando as heurísticas h_{dnn_sub}, h_{enn_sub}, h_{dnn_fator} e h_{enn_fator} possuem custo diferente dos caminhos computados pelo $A^\star$ com a heurística h_{r3}, foi empregada a função erro percentual médio absoluto (MAPE) (Função 2.4). Os custos dos caminhos produzidos por uma das heurísticas são subtraídos dos custos produzidos pela h_{r3}, somados em módulo e divididos pelo total de caminhos. Os resultados (Tabela 4.13) mostraram que ao substituir a heurística h_{r3} pela predição de uma DNN, o algoritmo $A^\star$ computa caminhos com uma diferença, em média, de até 0,47%. Por outro lado, apesar do tempo de execução, empregar DNN como um fator produz caminhos com o mesmo custo dos caminhos produzidos pelo $A^\star$ com a heurística h_{r3}.

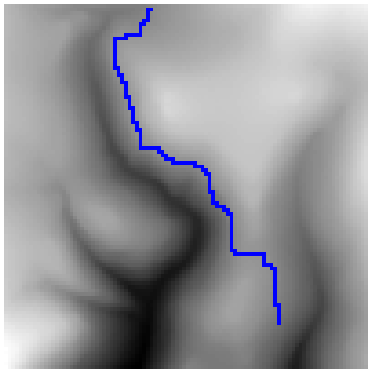
As Figuras 4.7a, 4.7b, 4.7c, 4.7d, 4.7e exibem exemplos de caminhos computados com o algoritmo $A^\star$ no Recorte 2 (Figura 3.3), entre uma origem e um destino utilizando diferentes funções heurísticas. Os caminhos mostrados nas Figuras 4.7a, 4.7c e 4.7e são os mesmos.

Figura 4.7 – Exemplos de caminhos computados com diferentes funções heurísticas

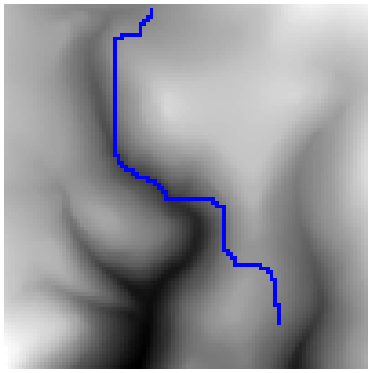
(a) Heurística h_{r3}. Custo $1788m$



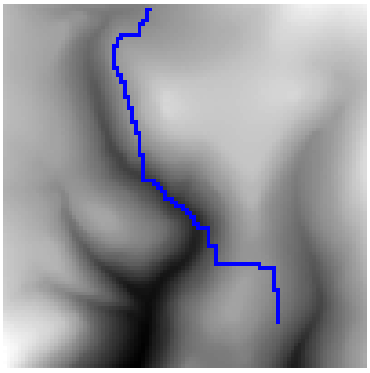
(b) Heurística h_{dnn_sub}. Custo $1799m$



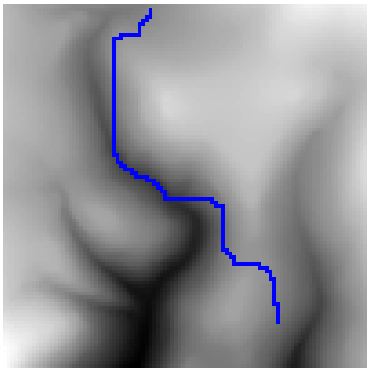
(c) Heurística h_{dnn_fator}. Custo $1788m$



(d) Heurística h_{enn_sub}. Custo $1807m$



(e) Heurística h_{enn_fator}. Custo $1788m$



Fonte: Autor

Tabela 4.13 – Diferenças entre caminhos computados

	Recorte 1	Recorte 2	Recorte 3
h_{dnn_sub}	0,24%	0,44%	0,47%
h_{enn_sub}	0,22%	0,4%	0,36%
h_{dnn_fator}	0%	0%	interrompido
h_{enn_fator}	0%	0%	interrompido

Fonte: Autor.

De maneira geral, os resultados de testes apontam que usar a predição de DNN como um fator, da maneira que foi analisada neste TCC, pode não ser uma boa estratégia para a otimização de algoritmos de busca de caminhos que consideram informações topográficas. Porém, a computação de caminhos de mesmo custo (isto é, com menores custos em termos de distância e inclinações do terreno consideradas ao longo do caminho retornado) justifica investigar as formas de tratar a predição da DNN como um fator de h_{r3}. Mostram ainda que, apesar do tempo de treinamento e do tempo de predição, utilizar i)

uma DNN apenas ou ii) mais de uma DNN produz resultados relativamente próximos entre eles. Por fim, indicam que DNN são capazes de generalizar o conhecimento topográfico aprendido a partir de uma amostra de pontos e respectivos custos de caminhos mínimos computados em tempo de pré-processamento dos mapas, produzindo caminhos próximos dos caminhos obtidos pelo algoritmo $A^\star$ com a função heurística h_{r3}.

O aumento do conjunto de pixels livres para deslocamento de um agente ampliam o tempo de treinamento das DNN de forma não linear. No entanto, esse aumento melhora o desempenho das DNN, de maneira que o algoritmo $A^\star$, utilizando a predição da DNN como valor heurístico, expande menos nodos e tem um tempo de execução menor quando comparado ao algoritmo base. Esse resultados são semelhantes aos obtidos por (DOEBBER, 2019). No entanto, em todos os mapas testados neste TCC houve redução no tempo de execução e quantidade de nodos expandidos quando a DNN foi aplicada em substituição da parcela heurística do algoritmo $A^\star$. Em (DOEBBER, 2019), alguns dos casos testados não obtiveram redução de tempo de execução ou quantidade de nodos expandidos quando a DNN substituiu a parcela heurística. Outra importante diferença em relação aos resultados de (DOEBBER, 2019) é que neste TCC os caminhos computados com o $A^\star$ utilizando a predição da DNN como valor heurístico tem custo diferente do caminho computado pelo algoritmo tomado como base. Em (DOEBBER, 2019) os caminhos computados possuem o mesmo custo do algoritmo tomado como base.

Assim, as hipóteses se verificam para DNN empregadas como substitutas da função heurística h (Equação 2.1) do algoritmo A^* (Algoritmo 1), reduzindo o tempo de execução e a quantidade de nodos expandidos durante a computação de caminhos com o algoritmo A^* entre dois pontos do mapa contendo informações topográficas.

5 CONCLUSÃO

Embora técnicas de DNN e *pathfinding* da Inteligência Artificial (IA) tenham sido exploradas com sucesso na resolução de inúmeros problemas de aplicação, elas ainda são pouco exploradas em conjunto com diferentes tipos de DNN recentemente propostas na literatura. Neste contexto, existe a necessidade de investigar a construção de heurísticas baseadas em DNN que permitam otimizar a busca de caminhos em terrenos virtuais. Em particular, este TCC explora o desenvolvimento de algoritmos de *pathfinding* que consideram informações topográficas nas suas computações de caminhos, as quais podem ser desenvolvidas em mapas de grandes dimensões em diferentes problemas de aplicação.

A contribuição do trabalho é aplicação de DNN como função heurística do algoritmo A^* (Algoritmo 1) na busca de caminhos em terrenos contendo informações topográficas. Em termos de tempo de execução e número de nodos expandidos e para os terrenos testados, a substituição do heurística tradicional (distância Euclidiana no $\mathbb{R}^3$) pela predição da DNN reduziu o custo computacional do algoritmo. A DNN foi treinada em um conjunto de dados formado por caminhos computados em 50% da área livre dos terrenos para circulação de um agente, e utilizando o algoritmo A^* com a heurística tradicional. Os terrenos empregados são oriundos de modelo digital de elevação e foram abstraídos por grafo orientado. A metodologia empregada não é generalista, para cada terreno uma DNN deve ser treinada.

Trabalhos futuros podem investigar outras maneiras de utilizar predições de DNN treinadas em combinação com funções heurísticas mais tradicionais; buscar arquiteturas de DNN mais simplificadas, com menores quantidades de neurônios, mas que ainda consigam aprender as características topográficas dos caminhos computados nos terrenos usados; usar formas diferentes de representar terrenos, além do uso de *grids* regulares, permitindo representar terrenos de grandes dimensões em maiores níveis de detalhe e realizar computações otimizadas de caminhos nestas estruturas; avaliar tipos de DNN diferentes, além das redes *feedforward* empregadas neste TCC; empregar e comparar resultados obtidos a partir do emprego de outras abordagens de aprendizado de máquina; investigar o aprendizado profundo em mapas maiores com abordagens de *pathfinding* hierárquicas; investigar o aprendizado a partir de dados sobre caminhos suavizados; avaliar a escalabilidade do emprego de DNN para mapas maiores; utilizar DNN combinadas em mapas de dimensões maiores; avaliar outros algoritmos de *pathfinding* e as possíveis combinações e otimizações destes com DNN.

REFERÊNCIAS BIBLIOGRÁFICAS

AGOSTINELLI, F.; MCALEER, S.; SHMAKOV, A.; BALDI, P. Solving the rubik's cube with deep reinforcement learning and search. **Nature Machine Intelligence**, v. 1, n. 8, p. 356–363, ago. 2019. ISSN 2522-5839. Disponível em: <https://doi.org/10.1038/s42256-019-0070-z>.

ALGFOOR, Z. A.; SUNAR, M. S.; KOLIVAND, H. A comprehensive study on pathfinding techniques for robotics and video games. **International Journal of Computer Games Technology**, Hindawi Publishing Corporation, v. 2015, p. 736138, abr. 2015. ISSN 1687-7047. Disponível em: <https://doi.org/10.1155/2015/736138>.

ALPSRP277832830. 2011. Disponível em: <https://search.asf.alaska.edu/#/?zoom=7.9746126485718065&center=-106.554889,38.882058&polygon=POLYGON((-108.5899%2037.7438,-105.4815%2037.7438,-105.4815%2040.0195,-108.5899%2040.0195,-108.5899%2037.7438))&dataset=ALOS&resultsLoaded=true&granule=ALPSRP277832830-KMZ>. Acesso em: 20 jan. 2021.

ARIKI, Y.; NARIHIRA, T. Fully convolutional search heuristic learning for rapid path planners. **CoRR**, abs/1908.03343, 2019. Disponível em: <http://arxiv.org/abs/1908.03343>.

ASF. **ASF Radiometrically Terrain Corrected ALOS PALSAR products**. 2015. Disponível em: <https://media.asf.alaska.edu/uploads/RTC/rtc_product_guide_v1.2.pdf>. Acesso em: 20 jan. 2021.

BREIMAN, L. Bagging predictors. **Machine learning**, Springer, v. 24, n. 2, p. 123–140, 1996.

CALIN, O. **Deep Learning Architectures**. [S.l.]: Springer, 2020. ISBN 9783030367206.

CHAGAS, C. **Algoritmos de busca de caminhos voltados para informações de altura e inclinação representadas em mapas de navegação**. 2019. 57 f. Monografia (Trabalho de Conclusão de Curso) — Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Maria, Santa Maria, 2019.

CHEN, S.; SHI, G.; LIU, Y. Fast path searching in real time 3d game. In: **2009 WRI Global Congress on Intelligent Systems**. [S.l.: s.n.], 2009. v. 3, p. 189–194. ISSN 2155-6091.

DAAC, A. **PALSAR_Radiometric_Terrain_Corrected_high_res**. [S.l.]: NASA Alaska Satellite Facility DAAC, 2014.

DIJKSTRA, E. W. et al. A note on two problems in connexion with graphs. **Numerische mathematik**, v. 1, n. 1, p. 269–271, 1959.

DOEBBER, D. M. **Uso de redes neurais profundas para o aprendizado de funções heurísticas para algoritmos de busca de caminhos**. 2019. 48 f. Monografia (Trabalho de Conclusão de Curso) — Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Maria, Santa Maria, 2019.

DRUCKER, H. Improving regressors using boosting techniques. In: **ICML**. [S.l.: s.n.], 1997. v. 97, p. 107–115.

GANGANATH, N.; CHENG, C.; TSE, C. K. A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains. **IEEE Transactions on Industrial Informatics**, v. 11, n. 3, p. 601–611, 2015. ISSN 1941-0050.

GANGANATH, N.; CHENG, C.-T.; CHI, K. T. Finding energy-efficient paths on uneven terrains. In: IEEE. **2014 10th France-Japan/8th Europe-Asia Congress on Mecatronics (MECATRONICS2014-Tokyo)**. [S.l.], 2014. p. 383–388.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A.; BENGIO, Y. **Deep learning**. [S.l.]: MIT press Cambridge, 2016. v. 1.

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**, IEEE, v. 4, n. 2, p. 100–107, 1968.

JINDAL, I.; TONY; QIN; CHEN, X.; NOKLEBY, M.; YE, J. A unified neural network approach for estimating travel time and distance for a taxi trip. **arXiv preprint arXiv:1710.04350**, 2017.

KUMAR, S. K. On weight initialization in deep neural networks. **CoRR**, abs/1704.08863, 2017. Disponível em: <http://arxiv.org/abs/1704.08863>.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, maio 2015. ISSN 1476-4687. Disponível em: <https://doi.org/10.1038/nature14539>.

LI, G.; WANG, G.; WANG, Q.; FEI, F.; Lü, S.; GUO, D. Ann: A heuristic search algorithm based on artificial neural networks. In: **Proceedings of the 2016 International Conference on Intelligent Information Processing**. New York, NY, USA: Association for Computing Machinery, 2016. (ICIIP '16). ISBN 9781450347990. Disponível em: <https://doi.org/10.1145/3028842.3028893>.

MCCULLAGH, P.; NELDER, J. A. **Generalized Linear Models**. [S.l.]: CRC Press, 1989. v. 37.

MYTTENAERE, A. D.; GOLDEN, B.; GRAND, B. L.; ROSSI, F. Mean absolute percentage error for regression models. **Neurocomputing**, Elsevier, v. 192, p. 38–48, 2016.

PüTZ, S.; WIEMANN, T.; SPRICKERHOF, J.; HERTZBERG, J. 3d navigation mesh generation for path planning in uneven terrain. **9th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2016**, v. 49, n. 15, p. 212–217, jan. 2016. ISSN 2405-8963. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2405896316310102>.

QIU, X.; ZHANG, L.; REN, Y.; SUGANTHAN, P. N.; AMARATUNGA, G. Ensemble deep learning for regression and time series forecasting. In: **2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)**. [S.l.: s.n.], 2014. p. 1–6.

REN, Y.; ZHANG, L.; SUGANTHAN, P. N. Ensemble classification and regression-recent developments, applications and future directions. **IEEE Computational intelligence magazine**, IEEE, v. 11, n. 1, p. 41–53, 2016.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. **International Conference on Medical image computing and computer-assisted intervention**. [S.l.], 2015. p. 234–241.

SAGI, O.; ROKACH, L. Ensemble learning: A survey. **WIREs Data Mining Knowl Discov**, John Wiley & Sons, Ltd, v. 8, n. 4, p. e1249, jul. 2018. ISSN 1942-4787. Disponível em: <https://doi.org/10.1002/widm.1249>.

SHARKEY, A. J. C. On combining artificial neural nets. **Connection Science**, Taylor & Francis, v. 8, n. 3-4, p. 299–314, 1996. Disponível em: <https://doi.org/10.1080/095400996116785>.

SOUZA, L. da Rocha e. **Busca hierárquica de caminhos com redes neurais profundas**. 2021. Monografia (Trabalho de Conclusão de Curso) — Curso de Graduação em Ciência da Computação, Universidade Federal de Santa Maria, Santa Maria, 2021.

TAKAHASHI, T.; SUN, H.; TIAN, D.; WANG, Y. Learning heuristic functions for mobile robot path planning using deep neural networks. **ICAPS**, v. 29, n. 1, p. 764–772, jul. 2019. Disponível em: <https://ojs.aaai.org/index.php/ICAPS/article/view/3545>.

WANG, J.; WU, N.; ZHAO, W. X.; PENG, F.; LIN, X. Empowering a* search algorithms with neural networks for personalized route recommendation. In: **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. New York, NY, USA: Association for Computing Machinery, 2019. (KDD '19), p. 539–547. ISBN 9781450362016. Disponível em: <https://doi.org/10.1145/3292500.3330824>.

WIEMANN, T.; LINGEMANN, K.; HERTZBERG, J. Automatic map creation for environment modelling in robotic simulators. In: **ECMS**. [S.l.: s.n.], 2013. p. 712–718.