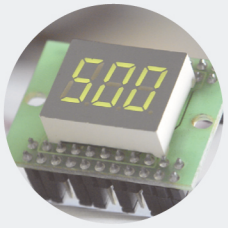
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Modelação e previsão de trajetórias com recurso a inteligência artificial

DIOGO SAMUEL TEIXEIRA DA SILVA
novembro de 2022

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Modelação e previsão de trajetórias com recurso a inteligência artificial

Diogo Samuel Teixeira da Silva

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Novembro, 2022

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Automação e Sistemas.

Candidato: Diogo Samuel Teixeira da Silva, Nº 1200475,
1200475@isep.ipp.pt

Orientação Científica: Ramiro Sousa Barbosa, rsb@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Novembro, 2022

Agradecimentos

Desde sempre fui apaixonado pela área da tecnologia e foi isso que me fez embarcar nesta jornada que não começou agora, mas sim há 5 anos atrás.

Esta jornada não tem sido fácil, tem tido altos e baixos, como tudo na vida, mas graças a várias pessoas que me acompanharam ao longo desde caminho, consegui terminar este capítulo da minha vida, com esperança de que o próximo seja tão bom ou melhor que este. Assim, começo por agradecer ao meu orientador de projeto que sempre se disponibilizou a ajudar-me em todos os momentos, facilitando, assim, o meu trabalho.

Agradeço principalmente à minha família, que me deu todo o seu apoio nesta que foi uma jornada longa e difícil em alguns momentos, ajudando-me a nunca desistir deste meu sonho de criança.

Por último e não menos importante agradeço à Daniela por toda a paciência, amor e carinho que me dedicou nesta caminhada.

A todos os envolvidos neste projeto, o meu muito obrigado por todo o apoio que me deram!

Termino desta forma, os meus agradecimentos, com uma frase de Fernando Pessoa: *"Tenho em mim todos os sonhos do mundo"*.

Resumo

A aplicação de Inteligência Artificial (IA) no setor automóvel tem vindo a ser uma mais valia principalmente quando se fala em segurança rodoviária.

A ligação entre o Homem e Máquina através da IA está vindo a ser mais fortalecida desde os anos 50 do século XX e a partir do século XXI, tem-se visto um crescimento tecnológico exponencial, estando disponíveis atualmente uma infinidade de aplicações em *Machine Learning* (ML) e *Deep Learning* (DL) nos vários setores existentes, desde o setor industrial ao setor da saúde.

No sub-setor que é a condução autónoma, a aplicação de algoritmos é feita principalmente para a segurança rodoviária, isto é, mapeamento do ambiente, deteção de objetos, pessoas, entre outros, leitura de sinalizações e previsão de trajetórias. Sendo também utilizada na categoria de conetividade como iteração com o condutor na leitura de gestos ou conetividade entre veículos, conhecida como *Vehicle-to-everything* (V2X). Já com uma certa maturidade deste conceito no setor automóvel, são várias as marcas e grupos que pretendem aplicá-lo em tarefas mais complexas, levando um dos propósitos da IA, a condução autónoma, para um próximo patamar.

Esta dissertação pretende assim dar entrada à condução autónoma com o processamento de diferentes dados veiculares e implementação de técnicas de DL para a previsão de trajetórias em MATLAB. Com este objetivo, foi primeiramente feita uma revisão bibliográfica sobre o tema, depois uma análise dos dados *Next Generation Simulation* (NGSIM) fornecidos pelo Departamento de Transporte dos Estados Unidos e desenvolvimento dos algoritmos e, por fim, realização de testes e comparação do desempenho dos vários algoritmos utilizados.

Desta forma, e sendo a previsão de trajetórias, em MATLAB, um tópico ainda com pouca aplicação de inteligência artificial, comparado com a classificação de imagens, pretende-se fornecer uma base de projeto para a entrada nesta área da condução autónoma.

Palavras-Chave: Previsão de Trajetórias, NGSIM, Inteligência Artificial, *Machine Learning*, *Deep Learning*, MATLAB.

Abstract

The application of artificial intelligence in the automotive sector has been an asset, especially when it comes to road safety.

The link between Man and Machine through artificial intelligence has been strengthened since the 50s of the 20th century and from the 21th century, technological advances have been seen with a exponentially growth, with currently a countless applications in ML and DL in the differents existing sectors, from the industrial sector to the health sector.

In the autonomous driving sub-sector, the application of algorithms is mainly done for road safety, that is, mapping the environment, detecting objects, people, among others, reading signs and predicting trajectories, being also used in category of connectivity such as iteration with the driver in reading gestures or connectivity between vehicles, known as V2X. With a certain maturity of this concept in the automotive sector, there are several brands and groups that intend to apply it in more complex tasks, taking one of the purposes of IA, autonomous driving, to the next level.

This dissertation therefore intends to introduce autonomous driving with the processing of different vehicular data and implementation of DL techniques for the prediction of trajectories in MATLAB. With this goal, a bibliographic review on the subject was first carried out, then an analysis of the NGSIM data provided by the US Department of Transportation and development of the algorithms and, finally, tests and comparison of the performance of the differents algorithms used.

In this way, and since the trajectory prediction, in MATLAB, is still a topic with low number of applications of artificial intelligence, compared to image classification, it is intended to provide a project basis for entry into this area of autonomous driving.

Keywords: Trajectory Prediction, NGSIM, Artificial Intelligence, Machine Learning, Deep Learning, MATLAB.

Índice

Lista de Figuras **vii**

Lista de Tabelas **ix**

Lista de Acrónimos **xi**

1 Introdução **1**
1.1 Contextualização . . . 1
1.2 Motivação . . . 2
1.3 Identificação do Problema . . . 2
1.4 Objetivos . . . 2
1.5 Contribuições . . . 3
1.6 Calendarização . . . 3
1.7 Estrutura . . . 4

2 Estado da arte **5**
2.1 Condução autónoma e tecnologias . . . 5
2.1.1 Previsão de cenários de tráfego . . . 7
2.1.2 Projetos associados . . . 8
2.2 Inteligência artificial . . . 9
2.2.1 *Machine Learning* . . . 10
2.2.2 *Deep Learning* . . . 13
2.2.3 Métodos de previsão de trajetórias . . . 16
2.3 *Software* MATLAB e Simulink . . . 22
2.3.1 Introdução ao MATLAB . . . 22
2.3.2 *Toolboxes* . . . 22

3 Dados e metodologia **25**
3.1 Estudo da base de dados . . . 25
3.2 Tratamento e análise de dados . . . 27
3.2.1 Conversão de unidades de medida . . . 27
3.2.2 Simulação das trajetórias . . . 28
3.2.3 Análise dos parâmetros . . . 30
3.2.4 Parâmetro distância lateral . . . 36

3.3 Pré-processamento e escolha de parâmetros 38
3.4 Arquitetura dos Algoritmos 40
3.4.1 Configurações e opções de treino 40
3.4.2 Camadas 41
3.4.3 LSTM-1 42
3.4.4 LSTM-2 42
3.4.5 LSTM-Drop-1 42
3.4.6 LSTM-Drop-2 43
3.4.7 Modelos Bidirecionais 43
3.4.8 Regression 44

4 Testes e resultados **47**
4.1 Cálculo dos erros 47
4.2 Demonstração e discussão de resultados 50

5 Conclusão e trabalhos futuros **57**

Referências **59**

Anexo A Código Projeto em MATLAB **63**

Lista de Figuras

2.1 Arquitetura do sistema de um veículo autónomo [4] 7
2.2 Segmentação geral de reconhecimento de objetos [11] 9
2.3 Grupos de aprendizagem em *Machine Learning* (ML) 11
2.4 Comparação entre classificação e regressão [24] 11
2.5 Exemplo de *clustering* [26] 13
2.6 Arquitetura de uma rede neuronal [27] 14
2.7 Arquitetura de uma rede neuronal convolucional [29] 14
2.8 Arquitetura de uma rede neuronal recorrente [30] 15
2.9 Estrutura básica do perceprão [31] 15
2.10 Sistema de coordenação e previsão de trajetórias [32] 18
2.11 Típico modelo de retropropagação [33] 19
2.12 Modelo de rede neuronal de retropropagação utilizado [33] 20
2.13 Arquitetura de uma *Long Short-Term Memory* (LSTM) [36] 20
2.14 Modelo proposto usando LSTM [38] 21
2.15 Ambiente de trabalho MATLAB [39] 22

3.1 troço da auto-estrada Interstate 80 [41] 26
3.2 Apresentação dos dados na tabela em MATLAB 28
3.3 *Frame* 557 obtido com a compilação do código de visualização 29
3.4 Velocidade média em cada faixa da I-80 31
3.5 Aceleração média em cada faixa da I-80 32
3.6 Distribuição de velocidade em cada faixa 33
3.7 Distribuição de aceleração em cada faixa 34
3.8 Distribuição do espaçamento entre veículos em cada faixa 35
3.9 Distribuição do tempo entre veículos em cada faixa 36
3.10 Distribuição da distância lateral quando há mudança faixa 37
3.11 Identificação do veículo escolhido no *frame* 2519 38
3.12 Gráfico da localização do veículo no troço 39
3.13 Gráfico da velocidade do veículo no troço 39
3.14 Opções de treino em MATLAB 41
3.15 Arquitetura LSTM-1 42
3.16 Arquitetura LSTM-2 42
3.17 Arquitetura LSTM com *dropout* 43

3.18 Arquitetura LSTM-Drop-2 43
3.19 Arquitetura Bi-LSTM-1 44
3.20 Arquitetura Bi-LSTM-2 44
3.21 Arquitetura Bi-LSTM-Drop-1 44
3.22 Arquitetura Bi-LSTM-Drop-2 44
3.23 Arquitetura Regression 45

4.1 Modelo LSTM-1 sem conhecimento (a) e (b) e com conhecimento (c) e (d) 50
4.2 Modelo LSTM-2 sem conhecimento (a) e (b) e com conhecimento (c) e (d) 51
4.3 Modelo LSTM-Drop-1 sem conhecimento (a) e (b) e com conhecimento (c) e (d) 52
4.4 Modelo LSTM-Drop-2 sem conhecimento (a) e (b) e com conhecimento (c) e (d) 53
4.5 Modelo Bi-LSTM-1 sem conhecimento (a) e (b) e com conhecimento (c) e (d) 53
4.6 Modelo Bi-LSTM-2 sem conhecimento (a) e (b) e com conhecimento (c) e (d) 54
4.7 Modelo Bi-LSTM-Drop-1 sem conhecimento (a) e (b) e com conhecimento (c) e (d) 54
4.8 Modelo Bi-LSTM-Drop-2 sem conhecimento (a) e (b) e com conhecimento (c) e (d) 55
4.9 Modelo Regression sem conhecimento (a) e (b) e com conhecimento (c) e (d) 55

Lista de Tabelas

1.1 Calendarização dos processos da dissertação 3

3.1 Descrição dos parâmetros dos dados *Next Generation Simulation* (NGSIM) 27

4.1 Resultados do cálculo do erro *Root Mean Squared Error* (RMSE), no parâmetro localização 48

4.2 Resultados do cálculo do erro, RMSE, no parâmetro velocidade . . . 49

Lista de Acrónimos

ADAM	*Adaptive Moment Estimation*
ADAS	*Advanced driver-assistance systems*
Bi-LSTM	*Bidirectional Long Short-Term Memory*
C-BCGAN	*Coordination-Bayesian Conditional. Generative Adversarial Network*
CNN	*Convolutional Neural Networks*
DL	*Deep Learning*
IA	Inteligência Artificial
LiDar	*Light Detection And Ranging*
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
NGSIM	*Next Generation Simulation*
ONNX	*Open Neural Network Exchange*
RMSE	*Root Mean Squared Error*
RNN	*Recurrent Neural Networks*
SAE	Society of Automotive Engineers
V2X	*Vehicle-to-everything*
VAE	*Variational AutoEncoder*
VRNN	*Variational Recurrent Neural Networks*

Capítulo 1

Introdução

Neste capítulo será realizada a introdução ao tema de dissertação. Será feita a apresentação da contextualização do tema, dos objetivos a cumprir, da calendarização e da estrutura do documento.

1.1 Contextualização

A segurança na condução tem sido, desde os finais do século XX, um dos focos principais quando é desenvolvido qualquer tipo de veículo. Embora as regras e sinais de estrada sejam fundamentais, são várias outras condições que aumentam a segurança rodoviária, como estruturas de veículos reforçadas, a criação de testes de segurança obrigatórios e utilização de sensores e assistências na condução. Esta última preocupação tem sido das mais desenvolvidas, uma vez que existe maior margem para evolução, tanto na parte dos sensores, como dos algoritmos.

Com o crescimento exponencial das várias tecnologias, são já inúmeros os veículos que estão equipados com sistemas de condução autónoma que são compostos por complexos sistemas em *hardware* e *software*. Consoante o nível de autonomização, podem ser encontradas diferentes aplicações a nível de *software* mas, nos níveis mais avançados, são principalmente realizadas operações com base em Inteligência Artificial (IA).

Os algoritmos de IA são aplicados em muitos setores e dependendo da complexidade e objetivo do projeto, são vários os modelos, técnicas e redes disponibilizadas, mais precisamente entre algoritmos em ML e *Deep Learning* (DL), campo da IA e

ML, respetivamente. Estes algoritmos funcionam à base de aprendizagem e dentro do campo de ML, são encontradas diferentes tipos de aprendizagens como supervisionada, não supervisionada e aprendizagem por reforço. A essência destes tipos de aprendizagem é a mesma, ou seja, os sistemas aprendem padrões a partir da análise de dados e são capazes de tomar decisões com ou sem intervenção humana.

As técnicas a aplicar para a previsão de valores futuros de trajetórias podem variar entre as referidas anteriormente, uma vez que se trata de simulação e não existem danos reais. Porém, e como se pretende que a previsão seja acertada e rápida de se obter, as melhores técnicas são de aprendizagem supervisionada e não supervisionada. Com esta determinação e fornecimento de parâmetros capazes de influenciar a resposta final, é possível o desenvolvimento de um método eficiente e com uma resposta realista.

1.2 Motivação

O desenvolvimento deste trabalho surge a partir do interesse pela área da inteligência artificial e pela área automóvel. Ambas as áreas têm-se vindo a complementar cada vez mais uma à outra através da assistência à condução, conexão entre veículos, entre outros e a margem para a inovação, a certeza destes setores no futuro e os princípios de funcionamento tornam estas áreas bastante interessantes e propensas ao acolhimento de alunos que queiram ingressar no mundo automóvel e no mundo da inteligência artificial.

1.3 Identificação do Problema

Considerando a complexidade que é a condução autónoma e o sub-campo da previsão de trajetórias, um tema ainda pouco aplicado em MATLAB, a pergunta fundamental que nos surge e que pretendemos responder é a seguinte:

É possível desenvolver e validar métodos precisos de previsão de trajetórias compostos por arquiteturas de DL em MATLAB?

1.4 Objetivos

Esta dissertação teve como principal objetivo desenvolver e testar diferentes metodologias de ML e DL para a previsão de valores futuros de localizações veiculares.

Para concretizar o objetivo principal foram estipuladas diversas tarefas, nomeadamente:

- Recolher e analisar o tipo de informação fornecida pelo Departamento de Transporte dos Estados Unidos;

- Identificar e comparar os melhores parâmetros para a previsão de valores futuros de localização;
- Desenvolver e comparar diferentes tipos de arquiteturas neste tipo de área;
- Validar os diferentes métodos desenvolvidos e avaliar o desempenho de cada um.

1.5 Contribuições

Esta dissertação engloba um conjunto de contribuições, das quais:

- Estudo de estado de arte do setor da condução autónoma e suas tecnologias, mais precisamente do reconhecimento de imagens e classificação de objetos e previsão de trajetórias, da área da IA, com foco no campo de DL e redes neuronais, e MATLAB;
- Análise de bases de dados disponibilizadas pelo departamento de transporte dos Estados Unidos da América, suas composição e arquiteturas de DL;
- Análise comparativa de diferentes modelos de DL e resultados na previsão de trajetórias;
- Disponibilização de um código em MATLAB para experimentação.

1.6 Calendarização

Esta dissertação foi iniciada em Março de 2022 e teve uma duração de 7 meses, terminando em Outubro de 2022. Durante este período, foram efetuados vários processos para a realização desta dissertação, que podem ser observados na Tabela 1.1.

Tabela 1.1: Calendarização dos processos da dissertação

	2022							
Atividades	**Mar**	**Abr**	**Maio**	**Jun**	**Jul**	**Ago**	**Set**	**Out**
Pesquisa bibliográfica								
Estudo dos dados								
Análise dos parâmetros								
Pré-processamento dos dados								
Desenvolvimento dos algoritmos								
Treino e teste de algoritmos								
Discussão dos resultados								
Relatório Dissertação								

1.7 Estrutura

A dissertação encontra-se dividida em cinco capítulos, dos quais:

- Capítulo 1, em que é feita uma contextualização do trabalho, apresentados sequencialmente os objetivos pretendidos, a calendarização desta e a estrutura do relatório;
- Capítulo 2, composto pelo estado de arte, onde está localizada grande parte da informação obtida para o desenvolvimento deste projeto. Neste capítulo é feita uma apresentação da área de trabalho e da inteligência artificial, com uma explicação das técnicas, tipos de aprendizagem, entre outros, e termina com a apresentação do programa utilizado;
- Capítulo 3, onde é feito um estudo da base de dados, apresentada a distribuição dos vários parâmetros, demonstradas as configurações e os algoritmos desenvolvidos;
- Capítulo 4, que se foca principalmente no cálculo de erros e definição dos melhores algoritmos, e amostragem dos resultados finais;
- Capítulo 5, e último, onde são feitas as conclusões obtidas com este projeto e trabalhos futuros pretendidos.

Capítulo 2

Estado da arte

Neste capítulo será feita uma apresentação da condução autónoma e tecnologias aplicadas, uma introdução à inteligência artifical, explicação de categorias, como ML e DL e aprofundamento em métodos de previsão de trajetórias. O capítulo é finalizado com a apresentação do *software* MATLAB.

2.1 Condução autónoma e tecnologias

A autonomização é um assunto que tem vindo a despertar cada vez mais interesse no mundo automóvel. A procura pela segurança, fiabilidade e conforto leva a que praticamente todos os fabricantes e grupos neste ramo invistam milhares de milhões anualmente no que toca ao desenvolvimento e melhoria de sistemas de apoio à condução [1]. Até agora, os sistemas de apoio à condução têm sido de tal forma desenvolvidos ao ponto de se tornarem sistemas autónomos e fazerem do veículo uma máquina capaz de levar os ocupantes de A a B sem qualquer intervenção do condutor. Consoante o nível de condução, o veículo é composto por diferentes versões no que toca a sensores, atuadores e algoritmos e, por isso, a Society of Automotive Engineers (SAE) classificou a autonomização em 6 níveis [2], nomeadamente:

- **Nível 0**: Inexistência de qualquer auxílio de sistemas tecnológicos, ou seja, o condutor executa todas as tarefas de condução;

- **Nível 1**: O controlo é totalmente humano, existindo unicamente e quando necessário uma assistência na travagem ou aceleração. Um exemplo é monitorização da velocidade com *cruise control*;
- **Nível 2**: A existência de assistências, como o sistema *Advanced driver-assistance systems* (ADAS), permite uma condução autónoma, porém, o condutor é obrigado a detetar a presença de obstáculos e a assumir o controlo da viatura quando necessário;
- **Nível 3**: Em algumas situações, o condutor pode desviar a atenção da estrada, sendo o veículo capaz de acelerar, travar, detetar outros veículos, objetos e sinais rodoviários e alterar de faixa, se necessário. Este nível já é visível em alguns fabricantes, porém, é restrito em alguns países. Uma das suas maiores utilizações é em auto-estrada;
- **Nível 4**: O veículo e seus sistemas são capazes de receber sinais em seu redor e controlar totalmente as suas ações, porém, por questões de segurança e dependendo das circunstâncias, o condutor precisa de intervir;
- **Nível 5**: Último nível, onde a máquina controla totalmente as decisões a tomar a toda a hora. Neste caso, o arranque, manobras, escolha do trajeto é efetuado pelo sistema e, caso existam ocupantes, os mesmos não necessitam de intervir em qualquer momento, sendo o sistema e veículo capazes de atuar em qualquer situação.

Até ao momento, já existem fabricantes com veículos equipados com o nível 4 de autonomização, porém, devido às legislações, pouca maturidade no mercado e incorreto funcionamento em algumas situações, o nível 4 de condução autónoma não é permitido na totalidade. Quanto ao nível 5, ainda não existe qualquer veículo em produção equipado com esta tecnologia.

Referente à arquitetura de um veículo com condução autónoma, esta é normalmente composta por diversos sensores, processadores, atuadores, componentes de *software* e comunicação [3], como se pode visualizar na Figura 2.1.

Todos estes componentes, com a presença da inteligência artificial permitem ao veículo tornar-se numa máquina inteligente que consegue responder à complexidade da condução.

Tal como um ser humano, o sistema necessita de visualizar o ambiente que o rodeia e, com a presença de sensores, câmaras, radares e sistemas *Light Detection And Ranging* (LiDar), o veículo recolhe imagens e dados para depois serem modelados. Estes dados são modelados por métodos com algoritmos que irão identificar e classificar os objetos próximos do veículo, como pedestres, carros, ciclistas, entre outros, [5][6]. Outro instrumento importante é a comunicação *Vehicle-to-everything* (V2X). Esta comunicação veicular permite alertar o sistema sobre perigos iminentes, como

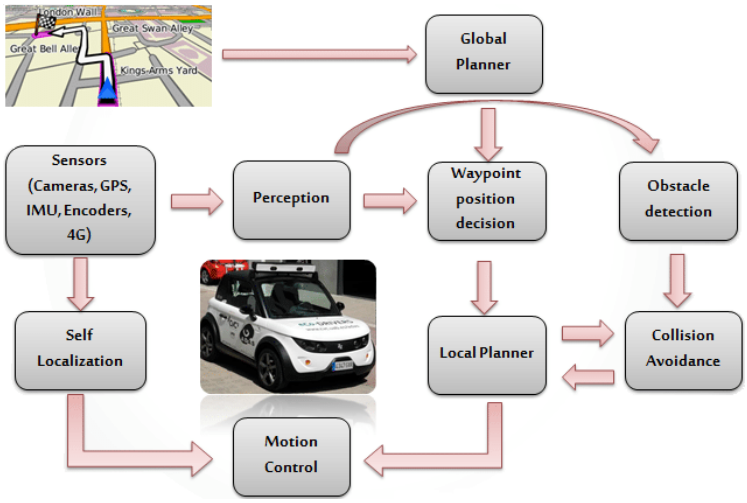


Figura 2.1: Arquitetura do sistema de um veículo autónomo [4]

paragens de emergência efetuadas por outros automóveis ou potenciais colisões em interseções, mas necessita de sensores de localização para funcionar corretamente [7].

O sistema, com conhecimento da sua localização, da localização de outros e do comportamento dos mesmos, consegue fazer um planeamento das melhores trajetórias e das velocidades ideais para a ocasião. Por fim, após o veículo ter uma visão do ambiente em que se encontra, saber a sua localização, ter um destino e processado a melhor rota possível, é necessário haver um sistema de controlo e sistemas de atuadores para a direção, aceleração e travagem. Desta forma, é criado um sistema de condução autónoma com foco na segurança e conforto dos ocupantes.

2.1.1 Previsão de cenários de tráfego

O desenvolvimento de sistemas de ajuda à condução vieram reduzir significativamente a percentagem de acidentes rodoviários, segundo [8]. Os condutores estão constantemente em contacto uns com os outros e são influenciados pelos mesmos. Devido a esta relação, os condutores com mais experiência tentam analisar o ambiente em que se situam e prever assim futuros movimentos ou ocasiões de outros condutores para a sua segurança. Este conjunto de acontecimentos também acontece na condução autónoma.

A previsão de cenários de tráfego permite ao sistema optar pelas melhores trajetórias ou velocidades consoante o ambiente em que se situam (ou possam situar), para que os ocupantes estejam o mais seguros possíveis [9]. Na previsão de cenários de tráfego existe o planeamento de rotas e o planeamento de trajetórias a curto prazo. O planeamento de rotas utiliza informações do mapa para definir o melhor trajeto a ser efetuado com base no tipo de trânsito que se possa vir a encontrar, tipos de vias a circular, economia do automóvel e tempo de viagem. Quanto ao

planeamento de trajetórias a curto prazo, o sistema é responsável pela determinação de um caminho a seguir na estrada em que se encontra. Neste tipo de planeamento, que é o tema a ser estudado nesta dissertação, o veículo necessita de avaliar as características em seu redor e compreender quais os movimentos futuros que outros condutores possam tomar de forma rápida e fiável, para que o veículo efetue uma trajetória de movimento o mais rápido possível de modo que o risco seja minimizado [10].

A complexidade das tarefas no planeamento de rotas e a sua importância para a segurança rodoviária, obriga a que sejam aplicados métodos de análise e de previsão precisos e eficientes. Normalmente, os métodos implementados são à base de algoritmos de inteligência artificial e a escolha dos mesmos deve-se à sua capacidade de análise a vários conjuntos de dados, *datasets*, como a velocidade dos veículos, em que vias se situam, qual o tipo de veículo, as suas localizações, entre outras características, e à sua fiabilidade no que toca à determinação de possíveis alterações de trajetórias, consoante os dados fornecidos pelos *datasets*.

2.1.2 Projetos associados

Os ambientes que um típico condutor pode encontrar durante uma viagem são imensos e é necessário ter em conta todos os detalhes possíveis, como velocidade do automóvel, proximidade de outros veículos, trajetórias, estado do pavimento, possíveis ocorrências de perigo, deteção de sinalização, entre outros. Assim, são vários os pontos que os engenheiros dedicados à condução autónoma podem ter como desenvolvimento de um projeto.

Dentro dos projetos possíveis de se implementar nesta área, um dos mais estudados é o reconhecimento e classificação de objetos.

Reconhecimento de imagens e classificação de objetos

O reconhecimento de imagens e classificação de objetos é de elevada importância quando se fala de condução autónoma e visão computacional. É com esta tecnologia que o sistema obtém e processa dados do ambiente onde se encontra permitindo, posteriormente, ao algoritmo implementado determinar trajetórias, situações de perigo, entre outros.

Para que seja possível haver um reconhecimento de imagem correto, segundo [11], e exemplificado na Figura 2.2 é necessário implementar um conjunto de tarefas: verificação de imagem, deteção de objetos, classificação da imagem, conhecimento do cenário e reconhecimento de objetos específicos.

Neste tipo de projetos é fortemente utilizada um tipo de arquitetura dentro de DL, chamada de redes neuronais convolucionais, *Convolutional Neural Networks* (CNN), que será posteriormente explicada. Resumidamente, desde as primeiras

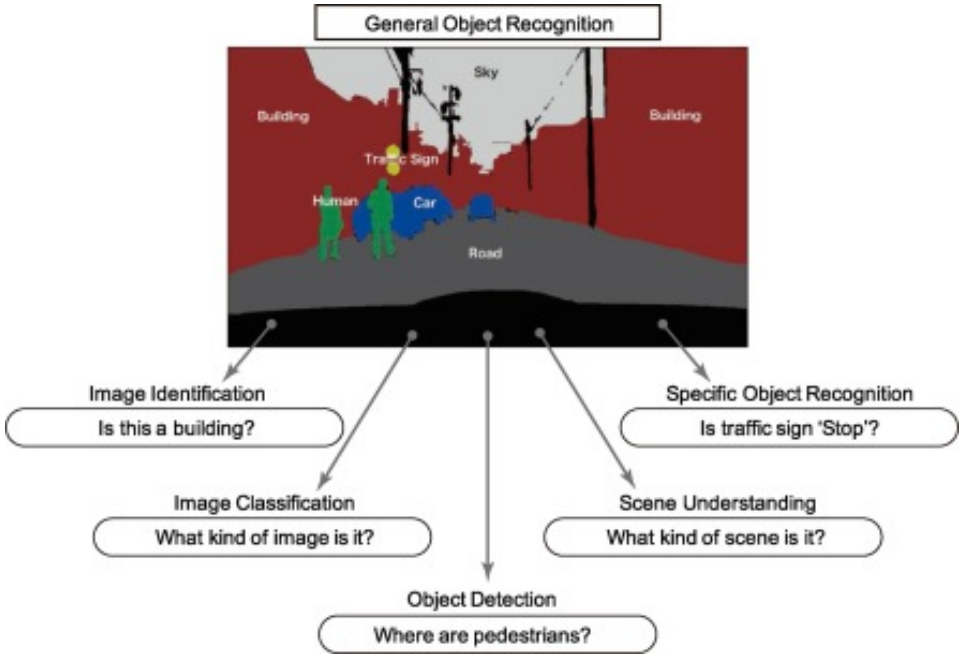


Figura 2.2: Segmentação geral de reconhecimento de objetos [11]

aplicações deste tipo de rede, são visíveis resultados bastante positivos no setor da visão computacional, em comparação com aproximações de ML, um campo mais vasto e onde está inserido o ramo de DL.

Dentro desta área, são várias as investigações, [11][12][13][14], que permitem o estudo da mesma e fazem parte do desenvolvimento de projetos que poderão tanto ser aplicados a nível académico ou industrial.

2.2 Inteligência artificial

A inteligência artificial é um termo que se refere a quando as máquinas são capazes de aprender, tomar decisões e simular pensamentos ou ações humanas. Este termo, oficialmente introduzido em 1956 [15], é bastante generalizado, isto porque qualquer técnica implementada que desse origem à replicação de ações humanas, era considerada inteligência artificial.

Com o passar dos anos e evolução da tecnologia, foi introduzido um ramo pertencente à inteligência artificial conhecido como ML. Composto por várias técnicas que levam à aprendizagem da máquina diretamente a partir dos dados sem depender de uma equação predeterminada como modelo, este método veio revolucionar várias áreas como a área do reconhecimento de voz [16], de objetos [17][18], da previsão de clima [19], de preços de mercado [20], da deteção de fraudes [21], entre outras. Mais recentemente, na década de 2010, foi desenvolvida uma sub-categoria no ML classificada como DL ou *deep machine learning*. Este termo é utilizado quando um sistema aprende a executar tarefas de classificação diretamente a partir de imagens,

textos ou sons. Este conjunto de técnicas permitiu trabalhar com um conjunto de dados muito superior ao de ML e assim obter resultados mais fiáveis.

Das técnicas referidas e do tipo de projeto a implementar, os métodos são baseados maioritariamente nos seguintes processos:

1. Recolha de informação;
2. Análise e pré-processamento de dados;
3. Aplicação de uma técnica de inteligência artificial;
4. Realização de treinos, validação e teste do algoritmo;
5. Aplicação do algoritmo.

Enquanto que na programação tradicional o programador efetua totalmente o código e processos para solucionar um problema com utilização de dados recolhidos, a técnica implementada em ML resume-se a usar um modelo que, com base em treinos, fará o reconhecimento de padrões nos dados e conseguirá dar resposta a informações que nunca foram anteriormente apresentadas.

2.2.1 *Machine Learning*

ML é um método que se baseia na ideia de que os sistemas podem aprender, identificar padrões e tomar decisões com pouca intervenção humana. A maior parte das indústrias que trabalham com grandes quantidades de dados, dão uso à tecnologia de ML porque permite fornecer resultados de forma rápida e precisa, mesmo com categorias distintas. Os serviços financeiros, com a previsão de valores de mercado, o governo, com prevenção de fraudes, a área da saúde, com a identificação de tendências ou situações de alerta e a área dos transportes, com identificação de rotas e deteção de objetos, são setores que utilizam bastante esta tecnologia.

Dos vários modelos de aprendizagem aplicados, segundo [22] existem três categorias principais e técnicas, visualizadas na Figura 2.3: aprendizagem supervisionada, aprendizagem não supervisionada e a aprendizagem por reforço.

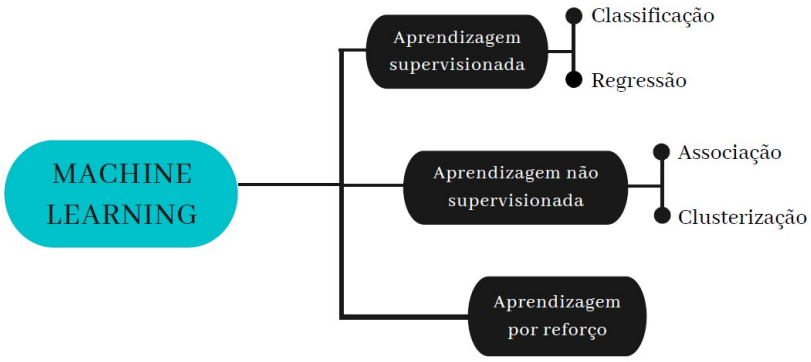


Figura 2.3: Grupos de aprendizagem em ML

Aprendizagem supervisionada

Os métodos de aprendizagem supervisionada funcionam com a utilização de dados e exemplos conhecidos. Neste caso, o programador ajuda na criação do modelo com a definição de classes e de exemplos de cada classe para que, posteriormente, haja uma comparação quando é apresentado algum dado novo. Um exemplo comum é o cálculo da velocidade de um veículo numa auto-estrada.

Com determinados parâmetros de entrada como hora atual, tipo de veículo e volume de veículos atuais nas vias, o algoritmo é treinado com essa informação e posteriormente é capaz de determinar possíveis velocidades médias de um automóvel. Segundo [23], neste tipo de aprendizagem existem duas categorias relevantes, que são: classificação e regressão. Na Figura 2.4 pode-se ver uma simples comparação entre estas duas categorias.

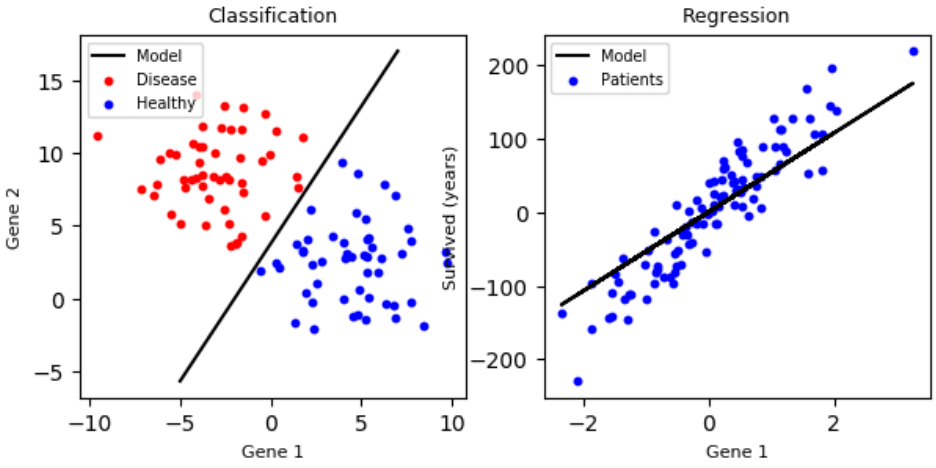


Figura 2.4: Comparação entre classificação e regressão [24]

Classificação é um tipo de aprendizagem em que o algoritmo é treinado para identificar a qual categoria pertence uma determinada amostra, com base no conjunto de dados de formação. Alguns problemas de classificação são binários, havendo classes de verdadeiro e falso e um exemplo disso é a classificação da velocidade de um veículo como "excesso de velocidade"ou "não excesso de velocidade". Para além da classificação binária, existe também a classificação categórica, uma classificação bastante utilizada para, por exemplo, classificar um veículo como "carro", "camião", "mota", entre outros.

Como referido anteriormente, existe a categoria regressão. O objetivo deste tipo de aprendizagem é desenvolver uma função de relação entre entradas e saídas para ajudar a máquina a perceber as alterações que ocorrem na saída dependendo das entradas. Em outros termos, com a regressão, o algoritmo pretende prever um valor futuro e um dos exemplos em que é aplicado este tipo de aprendizagem é, dada uma informação sobre volume e velocidade numa determinada secção de uma auto-estrada, pode prever-se uma velocidade média para um próximo período de tempo.

Aprendizagem não supervisionada

A aprendizagem não supervisionada, como o nome indica, não necessita de controlo para saber se o resultado está certo ou errado, isto porque neste tipo de aprendizagem, o objetivo é criar um conjunto de classes consoante padrões ou semelhanças entre objetos. O algoritmo é desenvolvido para aprender sozinho, através da observação e da descoberta, sem classes conhecidas como acontece na aprendizagem supervisionada. Exemplo deste método é o agrupamento de filmes numa plataforma de *streaming*. O algoritmo não sabe o significado do tipo de filmes, porém, através da análise do que o cliente costuma assistir, o algoritmo vai criando conjuntos de classes dependendo das semelhanças dos vários tipos, como duração, género de filme, atores, sem conhecimento do que se trata cada um dos tópicos. A associação e *clustering* são duas das técnicas mais aplicadas neste tipo de aprendizagem [25].

A associação, como o nome indica, permite correlacionar particulares tendências num conjuntos de dados que representam os padrões mais significativos. Um exemplo da aplicação deste método é a utilização de vários dados de um acidente rodoviário, como associação entre idades dos condutores, níveis de álcool, hora e local do acidente. Esta associação permite informar aos agentes, locais e horários onde é exigido um maior controlo da alcoolemia.

Para além da associação, a *clustering* é um método também bastante utilizado na aprendizagem não-supervisionada. Neste tipo de aprendizagem, um conjunto de dados são segmentados de maneira a criar vários grupos com base nas semelhanças entre eles, como se pode ver no exemplo da Figura 2.5.

Um analisador que supervisiona e tenta identificar a agressividade da condução com os dados gravados de uma auto-estrada em hora de ponta é um outro exemplo

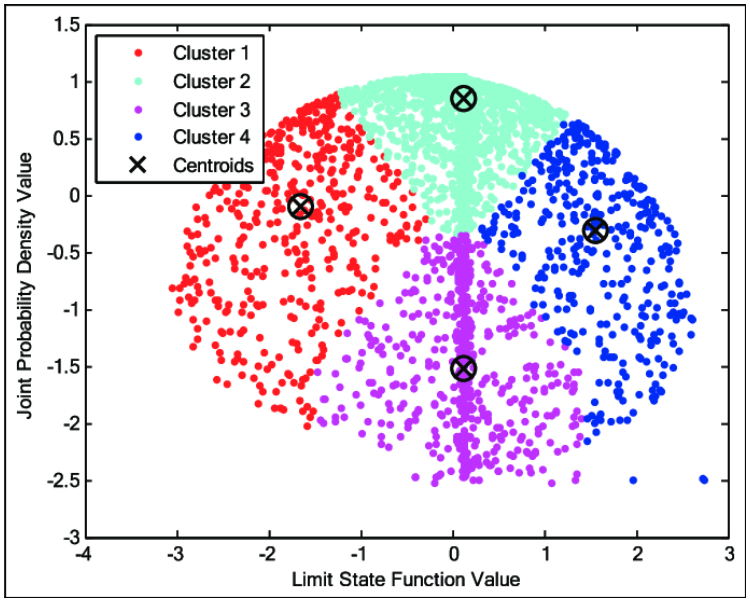


Figura 2.5: Exemplo de *clustering* [26]

de *clustering*. Com alguma informação dada e também conhecimentos próprios, o engenheiro tem o objetivo de categorizar o tipo de condução como agressiva, lenta ou normal com base, por exemplo, em acelerações e desacelerações.

Aprendizagem por reforço

Os algoritmos de aprendizagem reforçada funcionam sobre o princípio de tentativa-erro e são utilizados com mais frequência em robótica, jogos de vídeo e navegação. Neste tipo de aprendizagem, composta por três componentes, o agente, o ambiente e as ações, o algoritmo pretende desenvolver as ações que geram melhores recompensas. Neste caso, o agente define a política de recompensas, porém, não dá ao modelo quaisquer sugestões de como resolver o problema, ficando ao cargo do algoritmo descobrir como realizar tarefas de maneira a maximizar a recompensa.

2.2.2 *Deep Learning*

DL é um campo da tecnologia de ML que se inspira na funcionalidade das células cerebrais. Geralmente neste tipo de tecnologia são utilizadas arquiteturas de redes neuronais artificiais, nas quais podem existir centenas de camadas de processamento. O termo *"deep"* refere-se propositadamente ao número de camadas na rede e quantas mais camadas tiver, mais profunda será a mesma. A necessidade de se fornecer recursos manualmente no caso de ML é a principal diferença entre ambas as tecnologias, pois, no caso de DL, o sistema aprende com base em exemplos e trabalha normalmente com quantidades superiores de dados. O uso de maiores quantidades

de dados para treinar o algoritmo permite ao sistema ser o mais preciso possível na classificação.

Nas redes neuronais artificiais, podem ser distinguidas três camadas: a camada de entradas, a camada oculta e a camada de saídas. A camada de entrada é responsável pela obtenção de dados de fontes externas e colocação dos mesmos nas camadas ocultas. Nas camadas ocultas, é feita a maior parte do processamento, através de conexões entre nós e por fim, na camada de saída é apresentado o resultado final. Na Figura 2.6 pode visualizar-se a arquitetura de uma rede neuronal.

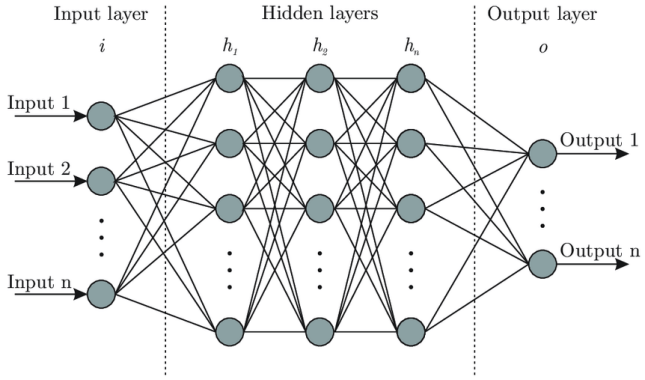


Figura 2.6: Arquitetura de uma rede neuronal [27]

Quando se aplica um algoritmo em DL, é necessário perceber que tipo de arquitetura de rede neuronal se vai utilizar, estando disponibilizadas redes CNN, e redes neuronais recorrentes, *Recurrent Neural Networks* (RNN) [28].

Redes neuronais convolucionais

As CNN são das arquiteturas mais utilizadas na atualidade quando se trata de classificação de imagens. A sua composição com filtros e atribuição de pesos eliminam informação irrelevante e determinam diferenças entre imagens. Na Figura 2.7 está apresentado um exemplo de arquitetura CNN, que permite pré-processamentos mais simples em comparação com outros algoritmos de classificação.

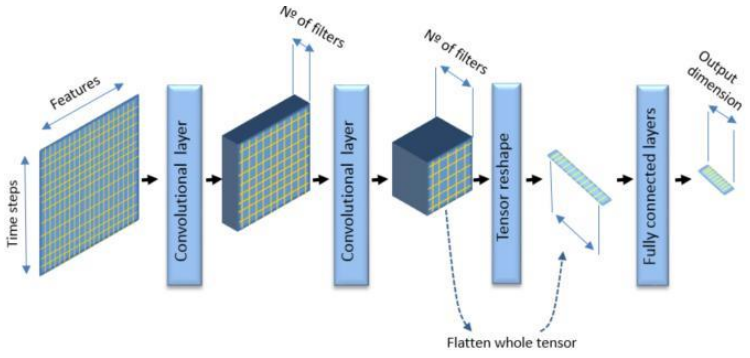


Figura 2.7: Arquitetura de uma rede neuronal convolucional [29]

Para a classificação de imagens, existem várias arquiteturas distintas baseadas no modelo CNN, das quais as mais conhecidas são: GoogleNet, AlexNet e ResNet.

Redes neuronais recorrentes

A RNN é um tipo de rede neuronal que usa dados sequenciais ou dados de séries temporais. Como funcionamento, esta rede utiliza os dados da saída de uma etapa anterior para alimentar a entrada da etapa atual, formando de certa forma, um ciclo, como está exibido na Figura 2.8.

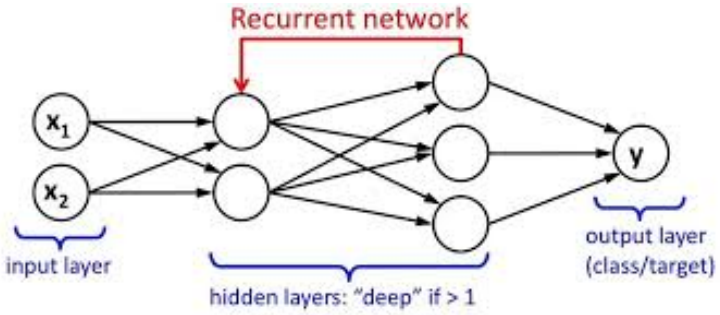


Figura 2.8: Arquitetura de uma rede neuronal recorrente [30]

Entre as várias RNN, uma das mais utilizadas e que será posteriormente explicada, é a rede LSTM.

Métodos de DL

Nas redes neuronais, é necessário que se faça um treino da rede e, para isso, são utilizados algoritmos de treino. Dos algoritmos existentes, os mais relevantes são o algoritmo com base na regra do perceptrão e da técnica de retropropagação. A estrutura básica do perceptão, criada por McCulloch-Pitts em 1943, pode ser observada na Figura 2.9, o qual é composto por um determinado número de entradas, uma saída, um peso w_0 referente à polaridade θ, pesos w_i referentes às entradas x_i e uma função de ativação em degrau.

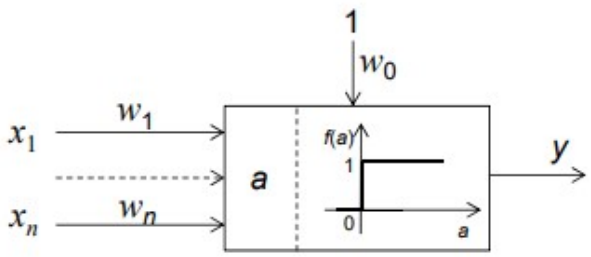


Figura 2.9: Estrutura básica do perceprão [31]

Em 1960 Rosenblatt propôs uma regra de aprendizagem do perceptrão com 4 passos distintos que acabaram por permitir ao algoritmo convergir para o resultado desejado. Os passos são:

1. Inicialização: Atribuídos pesos iniciais $w_1, w_2, ..., w_n$ e polaridade θ valores aleatórios no intervalo de -0.5 e 0.5;
2. Ativação: Calcular a saída atual para a iteração 1, aplicando as entradas e saída desejada;
3. Treino dos pesos: Atualizar os pesos do perceptrão;
4. Iteração: Incrementar a iteração de 1 e retornar ao passo 2 para repetir o processo até à convergência.

A utilização do perceptrão em monocamada fica limitada quando os problemas não são linearmente separáveis, com uma função de ativação em degrau ou linear. Para esses problemas é necessário utilizar redes neuronais multicamadas e, por vezes, a técnica de retropropagação. Num treino em multicamadas, o algoritmo tem duas fases, onde a primeira é conhecida como propagação e a segunda fase como retropropagação. Na primeira fase, após inseridos os dados de entrada, a resposta é propagada para as unidades das camadas seguintes até à camada de saída onde é obtida uma resposta da rede e calculado o erro. Após obtenção do erro, segue-se a segunda fase onde desde a camada de saída à camada de entrada são feitas alterações nos pesos da conexões até se obter o resultado desejado. A técnica de retropropagação torna-se assim bastante útil para problemas complexos até ao ponto de ser utilizada para a condução autónoma e previsão de trajetórias, como será explicada posteriormente com um exemplo no seguinte sub-capítulo.

2.2.3 Métodos de previsão de trajetórias

O desenvolvimento de métodos de previsão de trajetórias tem-se tornado bastante comum em várias organizações especializadas na condução autónoma. Visto que a previsão de tráfego se trata de um dos principais elementos quando um veículo se conduz autonomamente, são vários os parâmetros e métodos aplicados nesta categoria. Dependendo do tipo de aprendizagem, *machine* ou DL, o objetivo pretendido é o mesmo, ou seja, determinar trajetos de maneira a não ocorrer qualquer colisão entre os vários veículos. Entre os métodos existentes na inteligência artificial, serão referidos e explicados os que têm maior capacidade para obter os melhores resultados, como:

- Método baseado no teorema de *Bayes*;
- Método com base em algoritmos de retropropagação;

- Método focado na arquitetura de redes neuronais LSTM.

Teorema de *Bayes*

Thomas Bayes foi um famoso matemático e pioneiro no uso de probabilidades para a resolução de problemas que deu início a um teorema agora bastante utilizado, descrito por:

$$P(A \mid B) = \frac{P(B \mid A).P(A)}{P(B)} \tag{2.1}$$

Este teorema relaciona a probabilidade condicionada de $p(A|B)$ com $p(B|A)$ e afirma que com base na frequência de ocorrências de um determinado evento é possível realizar previsões, ou seja, o teorema de *Bayes* forma-se em torno de uma equação que mede a razão entre a probabilidade à *priori*, $p(A)$ e $p(B)$, que leva em conta o conhecimento de informações mais recentes de determinada situação, e a probabilidade à *posteriori*, $p(A|B)$ e $p(B|A)$, que considera dados adicionais relevantes em relação ao que se te trata. Para casos mais complexos, foram desenvolvidas também "redes bayesianas"que analisam várias variáveis ao mesmo tempo e ajudam na previsão de eventos.

Segundo os autores Li et al. (2019) [32], é possível desenvolver um sistema de previsão e coordenação de trajetórias com esta metodologia e será com base nesta investigação que será feito o estudo do teorema de *Bayes* na área da condução autónoma e da previsão de tráfego.

O principal objetivo é obter uma condição de distribuição multi-modal de trajetórias futuras consoante dados históricos que possa prever com precisão a trajetória de várias entidades. Assumindo que existem N entidades numa determinada situação, em que N pode variar em diferentes cenários de tráfego, os autores do projeto assumiram duas equações distintas, uma que funciona com base na informação histórica (2.2) e outra para trajetórias previsíveis (2.3).

$$T_{k-T_h+1:k} = \{t^j_{k-T_h+1:k} \mid t^j_k = (x^j_k, y^j_k), j = 1, ..., N\} \tag{2.2}$$

$$T_{k+1:k+T_f} = \{t^j_{k+1:k+T_f} \mid t^j_k = (x^j_k, y^j_k), j = 1, ..., N\} \tag{2.3}$$

Nestas equações, T_k e T_f são os comprimentos dos trajetos, x e y as coordenadas em 2D no espaço da imagem e k o *frame* do tempo atual. Desta forma, com estas equações é possível se obter uma condição de distribuição multi-modal de futuras trajetórias com dados do passado, com $p(T_{k+1:k+T_f} \mid T_{k-T_h+1:k})$.

O sistema implementado é formado por uma estrutura hierárquica, com um módulo de reconhecimento de coordenação a nível macro e um módulo de previsão

de padrões subtis a nível micro que resolve tarefas de geração probabilística, como se pode observar na Figura 2.10.

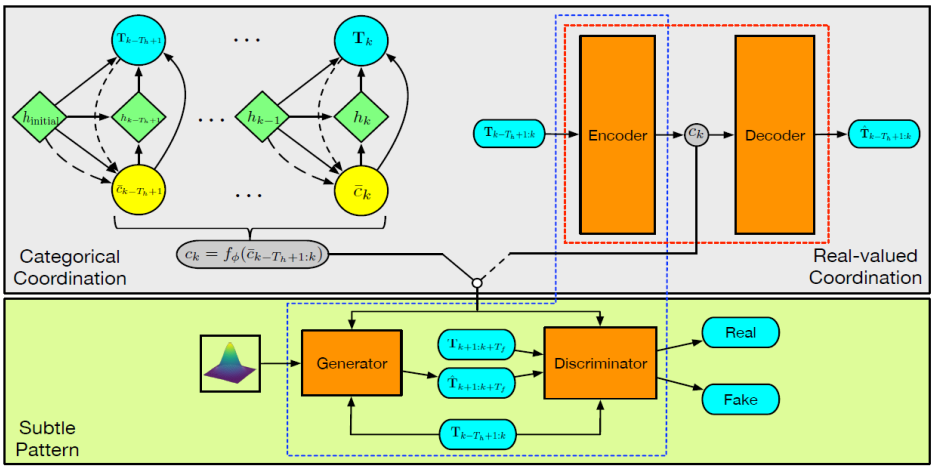


Figura 2.10: Sistema de coordenação e previsão de trajetórias [32]

De acordo com o tipo de objetivo, no módulo de reconhecimento de coordenação podem ser selecionados dois tipos de fórmulas, onde uma delas é composta por um classificador probabilístico baseado em características extraídas por *Variational Recurrent Neural Networks* (VRNN), que fornece uma distribuição discreta de coordenação, e outra que, com base em um modelo de estilo *Variational AutoEncoder* (VAE), fornece uma distribuição contínua da coordenação real. Quanto ao módulo de previsão de padrões, este é baseado numa *Coordination-Bayesian Conditional. Generative Adversarial Network* (C-BCGAN) em que o gerador toma como entrada as informações históricas, bem como um ruído de distribuição normal e gerar hipóteses de trajetórias.

A utilização deste teorema permite determinar um conjunto de trajetórias de várias entidades, como é possível verificar no projeto exemplificado. Assim, com a obtenção de resultados positivos por parte dos autores, este método torna-se viável e uma das opções para a realização deste projeto.

Algoritmo de retropropagação

Para além do uso do teorema de *Bayes* para previsão de cenários, são outras as técnicas e algoritmos usados para o mesmo fim, como algoritmo de retropropagação, mencionado anteriormente. Em DL, podem ser apresentadas várias camadas numa só arquitetura e com a aplicação da técnica de retropropagação, os sinais são propagados no sentido proativo (da entrada para a saída), mas também no sentido retroativo (com a retropropagação do erro de saída para as camadas anteriores).

Segundo os autores Chenxi Ding, Wuhong Wang, Xiao Wang e Martin Baumann [33], a expressão matemática de uma rede neuronal de retropropagação é definida como 2.4:

$$Y_j = f(\sum_{i=1}^{n} w_{ji}x_i - b_j) = f(n_j), \tag{2.4}$$

onde j representa o índice do neurónio em questão, i a referência a cada valor de entrada, x_i o valor de um parâmetro de entrada, w_{ji} o peso da conexão para os neurónios, b_j o valor limite que a saída pode ter, n_j o resultado do somatório das entradas multiplicadas pelos seus pesos e f é a função de ativação.

Para melhor perceção, na Figura 2.11 está exibido um simples modelo de retropopagação.

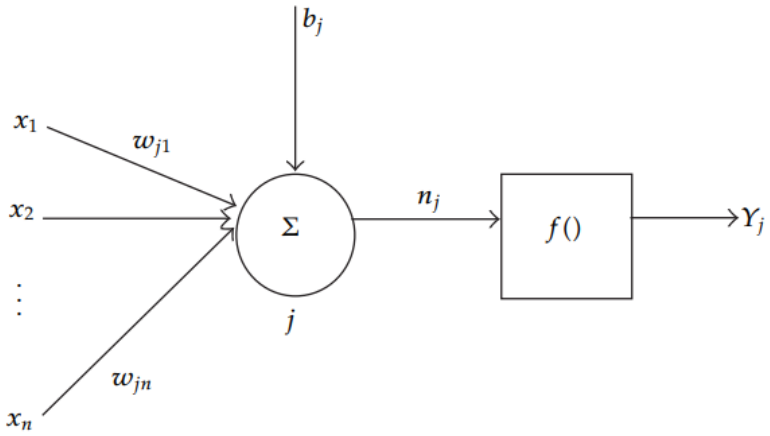


Figura 2.11: Típico modelo de retropropagação [33]

Esta investigação [33] fornece em detalhe a eficácia deste método para a previsão de trajetórias e mudança de faixa com base nos dados do veículo passado e com base na mesma, será realizada a explicação deste algoritmo tal como resultados da sua implementação.

Nesta pesquisa foram consideradas quatro variáveis de entrada: a posição, velocidade, aceleração e o tempo de avanço do veículo, uma saída que é a previsão do próximo estado e duas camadas ocultas. Segundo Ding et al. (2013) [33], foi utilizado um modelo da rede neuronal de retropropagação idêntico ao exibido na Figura 2.12.

Neste modelo, as variáveis correspondem à fórmula matemática explicada anteriormente, onde o vetor X é um vetor composto pelas entradas x_i, f_j as funções de ativação que neste caso é uma função sigmoide não linear na primeira camada e uma função linear na segunda e Y a saída. Com a realização de testes os autores foram capazes de obter resultados bastante satisfatórios, sendo que com um aumento de iterações a rede torna-se mais eficaz para com os resultados pretendidos e concluíram

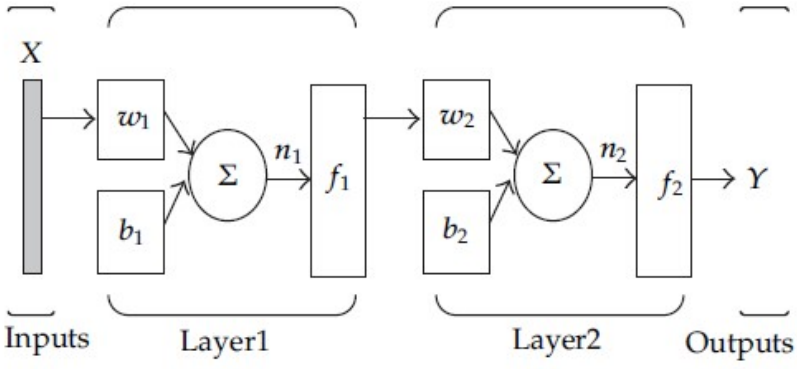


Figura 2.12: Modelo de rede neuronal de retropropagação utilizado [33]

que este método e modelo são boas bases para desenvolver modelos de mudança de vias mais complexos.

Arquitetura *Long Short-Term Memory* (LSTM)

Como particular implementação de redes neuronais recorrentes, a rede neuronal LSTM é bastante utilizada tanto na previsão de trajetórias de pedestres [34] como de veículos [35] em interseções, devido à sua eficiência.

A sua estrutura em cadeia contém quatro redes neuronais e blocos chamados de células, que permitem a inexistência de funções de ativação. Numa LSTM é utilizado o estado anterior e entrada atual, como numa rede neuronal recorrente comum, mas também o estado antigo das células de memória. A arquitetura de uma LSTM pode ser observada na Figura 2.13.

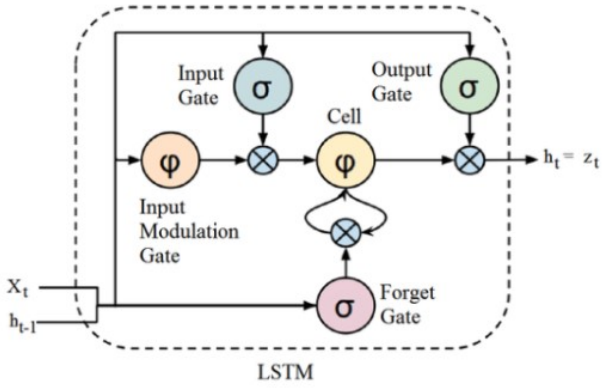


Figura 2.13: Arquitetura de uma LSTM [36]

Neste tipo de técnica, as células mantêm a informação e os *gates* fazem a manipulação de memória. Como se pode visualizar na figura acima, existem três *gates*:

input gate, *forget gate* e *output gate*, duas entradas, x_t e h_{t-1}, uma para o tempo específico e outra com dados da célula anterior e uma saída h_t.

No *input gate* ocorre a adição de dados úteis ao estado da célula que são filtrados e fornecidos os dados a serem lembrados e também a criação de um vetor que, com ajuda de uma função *tanh* que dá saída -1 ou $+1$, contém valores de x_t e h_{t-1}. De seguida é feita uma multiplicação entre os valores regulados e o vetor de forma a serem obtidas informações úteis.

No *forget gate*, são inseridos dados de ambas as entradas e com a multiplicação dos mesmos com matrizes de peso e passagem por uma função de ativação, resulta na decisão sobre se uma determinada informação é esquecida ou retida para uso futuro.

Quanto ao *output gate*, é gerado um vetor com a aplicação da função *tanh* na célula e dados atuais e realizada uma filtragem com uma função sigmóide nos dados a serem lembrados. Por fim, é feita uma multiplicação entre os valores do vetor e os valores regulados para serem enviados como uma saída.

Altché and Fortelle (2017) [37] e Deo and Trivedi (2018) [38] são alguns dos vários autores que utilizaram esta técnica para desenvolverem a sua investigação. Como estes trabalhos e todas as investigações referidas anteriormente, o objetivo de Deo and Trivedi (2018) foi obter um algoritmo capaz de prever trajetórias num determinado cenário de tráfego, mais precisamente, previsão de manobras. O modelo proposto pode ser observado na Figura 2.14.

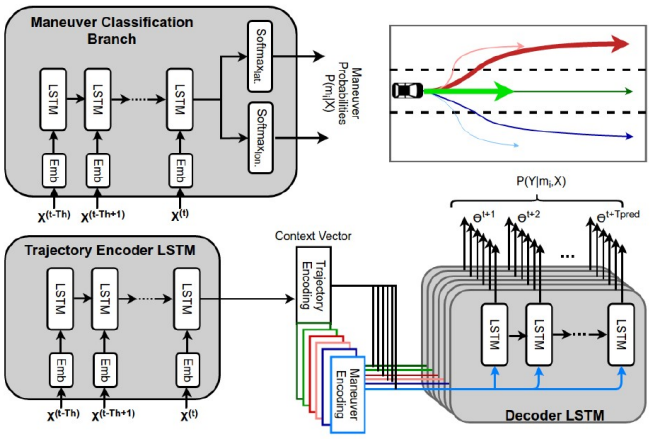


Figura 2.14: Modelo proposto usando LSTM [38]

Neste diagrama é possível observar um codificador, um descodificador e um classificador de trajetórias. O codificador de trajetórias LSTM codifica os dados históricos e as posições do veículo em estudo e os veículos adjacentes num vetor. O descodificador gera distribuições de manobras futuras do veículo em intervalos de tempo e o classificador atribui probabilidades de manobra.

A realização de testes e comparação com outros modelos, como uma mistura de modelos gaussiana com um campo aleatório de Markov e modelo com unidades recorrentes fechadas, permitiram concluir que o erro obtido com o método LSTM era inferior aos restantes e assim, um bom método para previsão de trajetórias.

2.3 *Software* MATLAB e Simulink

O MATLAB e Simulink são instrumentos de programação desenvolvidos pela *MathWorks* que são utilizados tanto a nível industrial como académico.

2.3.1 Introdução ao MATLAB

O MATLAB é uma ferramenta que permite o desenvolvimento de algoritmos em várias áreas de investigação como robótica, ML e DL, processamento de sinais, entres outras e o Simulink, uma ferramenta de simulação que funciona à base da programação gráfica, mais precisamente, de diagrama de blocos. A junção de ambos os instrumentos com a utilização do editor de texto no MATLAB, observado na Figura 2.15, simulação no Simulink e análise dos resultados obtidos no MATLAB e a disponibilização de cursos *on-ramp*, alguns gratuitos, faz com que sejam ferramentas bastante intuitivas e o desenvolvimento de um projeto seja facilitado.

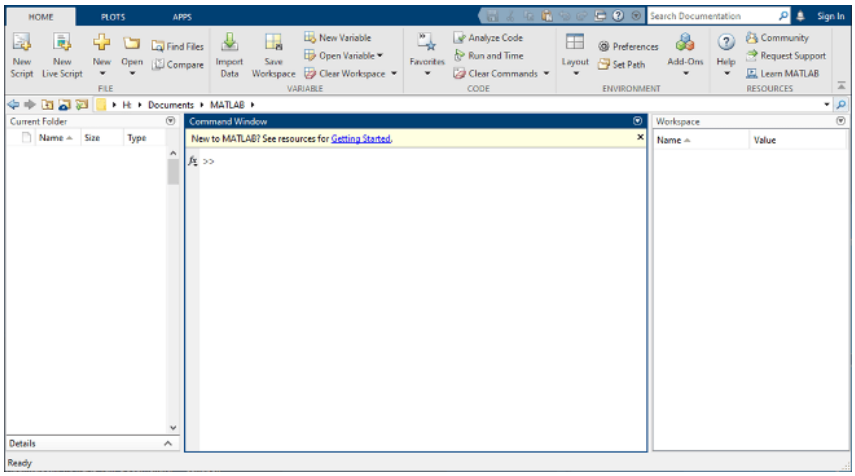


Figura 2.15: Ambiente de trabalho MATLAB [39]

2.3.2 *Toolboxes*

Toolboxes são bibliotecas auxiliares compostas por funções ou diagramas já definidos que facilitam no desenvolvimento de algoritmos e otimizam o tempo investido para realizar tarefas. No desenvolvimento de um algoritmo capaz de prever trajetórias veiculares é importante o estudo e utilização de bibliotecas de ML e DL devido à complexidade do tema e, por isso, serão explicadas as seguintes *toolboxes*:

- *Deep Learning Toolbox*;
- *Automated Driving Toolbox.*

Deep Learning Toolbox

A *Deep Learning Toolbox* é uma biblioteca bastante vasta que fornece uma estrutura para projetar e implementar redes neuronais, disponibilizando ao utilizador algoritmos, vários modelos pré-treinados e aplicações. Com esta biblioteca, é possível utilizar redes neuronais convolucionais, redes LSTM como foi referido anteriormente, alterar parâmetros das mesmas e até importar modelos de outras plataformas de ML como *PyTorch* e *TensorFlow* através do formato *Open Neural Network Exchange* (ONNX).

Automated Driving Toolbox

A *Automated Driving Toolbox* é uma das bibliotecas que melhor se encaixa para o desenvolvimento de um algoritmo relacionado com condução autónoma. Esta *toolbox* fornece algoritmos e ferramentas para projetar, simular e testar sistemas de condução autónoma. Algumas das suas utilidades são a simulação de cenários de tráfego, desenvolvimento e teste de algoritmos de processamento de visão e LiDar para a condução autónoma, localização de faixas, sinalização, veículos, entre outros para determinar colisões ou movimentos, planeamento de trajetórias e controlo veicular.

Capítulo 3

Dados e metodologia

O presente capítulo tem como propósito apresentar a origem dos dados, a sua composição e parâmetros, explicar métodos aplicados para a visualização das trajetórias, influências entre parâmetros e métodos de processamento que servirão para preparar os dados e serem aplicados por um algoritmo que determinará futuras trajetórias.

3.1 Estudo da base de dados

O desenvolvimento de um projeto para a previsão de trajetórias depende bastante dos dados iniciais disponíveis. A quantidade de informação, os diferentes parâmetros e o detalhe que se pode encontrar numa base de dados de trajetórias de veículos, influencia a variedade de técnicas a aplicar para obter o mesmo resultado e pode determinar a qualidade de um algoritmo.

Um dos conjuntos de dados mais utilizados nesta área de aplicação são os dados fornecidos pelo departamento de transportes dos Estados Unidos da América, os dados NGSIM [40]. Os dados fornecidos começam com a obtenção de imagens através de várias câmaras de vídeo sincronizadas que, posteriormente, são submetidas a códigos para a deteção dos veículos e determinação dos vários parâmetros de cada um.

O departamento de transporte americano, com o programa NGSIM, disponibiliza dados de várias estradas americanas, mais precisamente de secções da auto-estrada US-101 e da Interstate 80 e dados da Lankershim Boulevard e da Peachtree [40]. Para o processamento e aplicação de algoritmos de inteligência artificial a aplicar

no projeto foram submetidos os dados referentes à troço da Interstate 80. A sua estrutura e condições, ou seja, várias faixas, velocidades bastante díspares e falta de sinalização permite desenvolver ou aplicar algoritmos de diferentes complexidades e dar entrada na área da condução autónoma e previsão de trajetórias. Na Figura 3.1 é ilustrada a troço em causa.

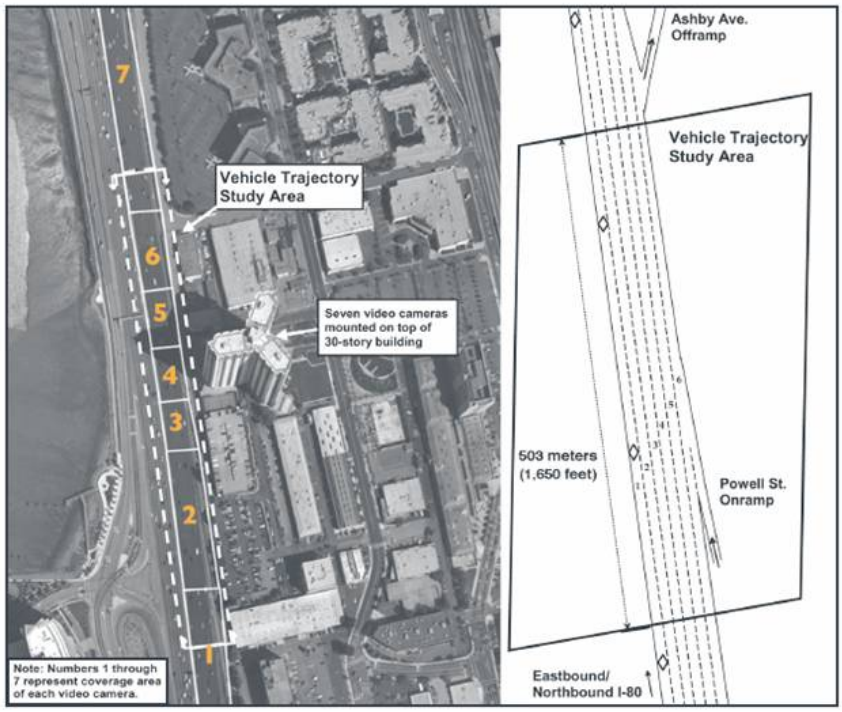


Figura 3.1: troço da auto-estrada Interstate 80 [41]

No dia 13 de Abril de 2005 foram adquiridos, pelos investigadores, os dados detalhados das trajetórias dos veículos. A área de estudo tem aproximadamente 500 metros e 6 faixas no total, sendo uma delas utilizada para a entrada e saída de veículos naquele troço e, como apoio a quem utilizar estes dados, estão também disponíveis documentos de informação e algumas gravações da auto-estrada obtidas por 7 câmaras de vídeo.

Quanto aos dados, são fornecidos três ficheiros `.csv` ou `.txt`, em que os valores estão separados por vírgulas, referentes a 15 minutos de gravação nas horas de maior congestão, totalizando 45 minutos de informação entre as 16h:00m e 16h:15m, as 17h:00m e 17h:15m e as 17h:15m e 17h:30m. Cada ficheiro de 15 minutos referente à I-80 é composto por 1 048 575 observações de 1725 veículos distintos em que cada observação é obtida a uma frequência de 10 Hertz (100 milissegundos) e é caracterizada inicialmente por 18 parâmetros. Na Tabela 3.1 estão listados e descritos todos os parâmetros fornecidos nos ficheiros `.csv` ou `.txt` da Interstate 80.

Tabela 3.1: Descrição dos parâmetros dos dados NGSIM

Parâmetro	Descrição
Vehicle_ID	Número de identificação do veículo.
Frame_ID	Número de identificação do *frame*.
Total_Frames	Número total de *frames* em que o veículo aparece neste conjunto de dados.
Global_Time	Tempo decorrido em milissegundos desde 1º de janeiro de 1970.
Local_X	Coordenada X do veículo (em pés) em relação à borda mais à esquerda do troço, no sentido da marcha.
Local_Y	Coordenada Y do veículo (em pés) em relação à borda de entrada do troço.
Global_X	Coordenada X do veículo (em pés) com base no CA State Plane III em NAD83.
Global_Y	Coordenada Y do veículo (em pés) com base no CA State Plane III em NAD83.
v_Length	Comprimento do veículo (em pés).
v_Width	Largura do veículo (em pés).
v_Class	Tipo de veículo: 1 - motociclo, 2 - automóvel, 3 - pesado
v_Vel	Velocidade instantânea do veículo (em pés/segundo).
v_Acc	Aceleração instantânea do veículo (em pés/segundo ao quadrado).
Lane_ID	Número da faixa em que se encontra o veículo.
Preceding	Identificação do veículo em frente na mesma faixa.
Following	Identificação do veículo atrás na mesma faixa.
Space_Headway	Espaço até ao veículo da frente (em pés).
Time_Headway	Tempo até ao veículo da frente (em segundos).

3.2 Tratamento e análise de dados

Antes de ser feito um processamento de dados para depois se utilizar um algoritmo e se fazer a previsão de trajetórias, é necessário entender o comportamento dos veículos e a variação de valores em determinadas situações.

Apesar da partilha de três horários diferentes foi obrigatória a escolha de apenas um dos ficheiros a utilizar neste projeto devido à elevada quantidade de observações a analisar. Posteriormente podem ser utilizados todos os ficheiros, porém, em função do curto espaço de tempo disponível e da capacidade da máquina, tiveram de ser feitas reduções de dados.

3.2.1 Conversão de unidades de medida

As bases de dados são de origem americana e, por isso, os dados estão sob o sistema imperial. Para haver uma melhor perceção das trajetórias e dos valores em cada momento, foram convertidas as unidades de alguns parâmetros do sistema imperial para sistema métrico. As localizações em X e em Y, a largura e comprimento do veículo e o espaçamento entre veículos foram convertidos de pés para metros, a

velocidade passou de pés por segundo (ft/s) para quilómetros por hora (km/h) e a aceleração começou a ser apresentada em metros por segundo ao quadrado (m/s^2) ao invés de pés por segundo ao quadrado (ft/s^2). Estes valores facilmente foram convertidos com a aplicação da seguinte fórmula (3.1):

$$metros = pés * 0.3048 \tag{3.1}$$

No caso da velocidade, foi posteriormente conveniente converter de m/s para km/h, com a simples multiplicação do novo valor do parâmetro por 3.6. Na Figura 3.2 é possível observar a tabela com os dados a trabalhar após a conversão.

i80

1048575x18 table

	1 Vehicle_ID	2 Frame_ID	3 Total_Frames	4 Global_Time	5 Local_X	6 Local_Y	7 Global_X	8 Global_Y	9 v_Length	10 v_Width	11 v_Class	12 v_Vel	13 v_Acc	14 Lane_ID	15 Preceding F
1	1	12	884	1.1134e+12	5.1462	14.6953	6.0428e+06	2.1331e+06	4.3586	1.9507	2	13.7160	0	2	0
2	1	13	884	1.1134e+12	5.1627	15.0763	6.0428e+06	2.1331e+06	4.3586	1.9507	2	13.7160	0	2	0
3	1	14	884	1.1134e+12	5.1789	15.4570	6.0428e+06	2.1331e+06	4.3586	1.9507	2	13.7160	0	2	0
4	1	15	884	1.1134e+12	5.1953	15.8383	6.0428e+06	2.1331e+06	4.3586	1.9507	2	13.7160	0	2	0
5	1	16	884	1.1134e+12	5.2115	16.2193	6.0428e+06	2.1331e+06	4.3586	1.9507	2	13.7160	0	2	0
6	1	17	884	1.1134e+12	5.2276	16.6003	6.0428e+06	2.1331e+06	4.3586	1.9507	2	13.7050	-0.0274	2	0
7	1	18	884	1.1134e+12	5.2438	16.9810	6.0428e+06	2.1331e+06	4.3586	1.9507	2	13.6941	-0.0244	2	0
8	1	19	884	1.1134e+12	5.2599	17.3602	6.0428e+06	2.1331e+06	4.3586	1.9507	2	13.7379	0.1676	2	0
9	1	20	884	1.1134e+12	5.2822	17.7391	6.0428e+06	2.1331e+06	4.3586	1.9507	2	13.9025	0.6736	2	0
10	1	21	884	1.1134e+12	5.2697	18.1243	6.0428e+06	2.1331e+06	4.3586	1.9507	2	14.2646	1.3503	2	0
11	1	22	884	1.1134e+12	5.2416	18.5245	6.0428e+06	2.1331e+06	4.3586	1.9507	2	14.8023	1.7191	2	0
12	1	23	884	1.1134e+12	5.1898	18.9455	6.0428e+06	2.1331e+06	4.3586	1.9507	2	15.3400	1.4539	2	0
13	1	24	884	1.1134e+12	5.1398	19.3828	6.0428e+06	2.1331e+06	4.3586	1.9507	2	15.7569	0.8656	2	0
14	1	25	884	1.1134e+12	5.1060	19.8285	6.0428e+06	2.1331e+06	4.3586	1.9507	2	15.9983	0.3566	2	0

Figura 3.2: Apresentação dos dados na tabela em MATLAB

3.2.2 Simulação das trajetórias

Com o fornecimento de diferentes conjuntos de dados e para ser feita a melhor escolha em relação ao ficheiro e suas trajetórias foi importante verificá-las, pois os dados fornecidos podem remeter a um momento de elevada congestão, com pouca variação de valores. Para isso, foi aplicado um código de visualização que utiliza como base o número do *frame*, a localização em X e Y, o número de identificação dos veículos, o seu comprimento, a sua largura e a sua classe.

No código de visualização das trajetórias está predefinida uma taxa de amostragem de 100 milissegundos e os valores para o comprimento e largura do troço. Os valores relativos à via sofreram alterações em função da conversão realizada anteriormente. Neste caso, para ser visível o conjunto total de trajetórias, foi alterado o valor inicial e final do troço para a sua totalidade, entre os 0 metros e os 500 metros e a largura, que é correspondente à soma da largura das vias, mais um ligeiro espaçamento para a descrição do gráfico.

Durante o processo de atualização é criado um bloco para simular graficamente o veículo em questão com diferentes tamanhos, consoante o comprimento e largura do mesmo, e diferentes cores dependendo da classe, neste caso amarelo para um

automóvel, verde para um veículo pesado e vermelho para um motociclo. É de realçar que os veículos apresentam a forma de retângulos transversais porque a escala está definida para amostrar o comprimento total do troço, com 500 metros. Assim, os veículos, com comprimentos entre os 4 e os 10 metros, representam entre 0.8% e 2% da escala, tornando-os nestas formas mais estreitas. Para melhorar a apresentação das trajetórias foram também adicionadas linhas referentes às várias vias existentes. Estas linhas não servem apenas para melhorias estéticas, mas ajuda na diferenciação das vias e como se comportam os veículos de cada uma. Na Figura 3.3, é possível observar o *frame* 557 do troço da I-80 e suas trajetórias com os vários blocos identificados com o número do veículo que estão a reproduzir.

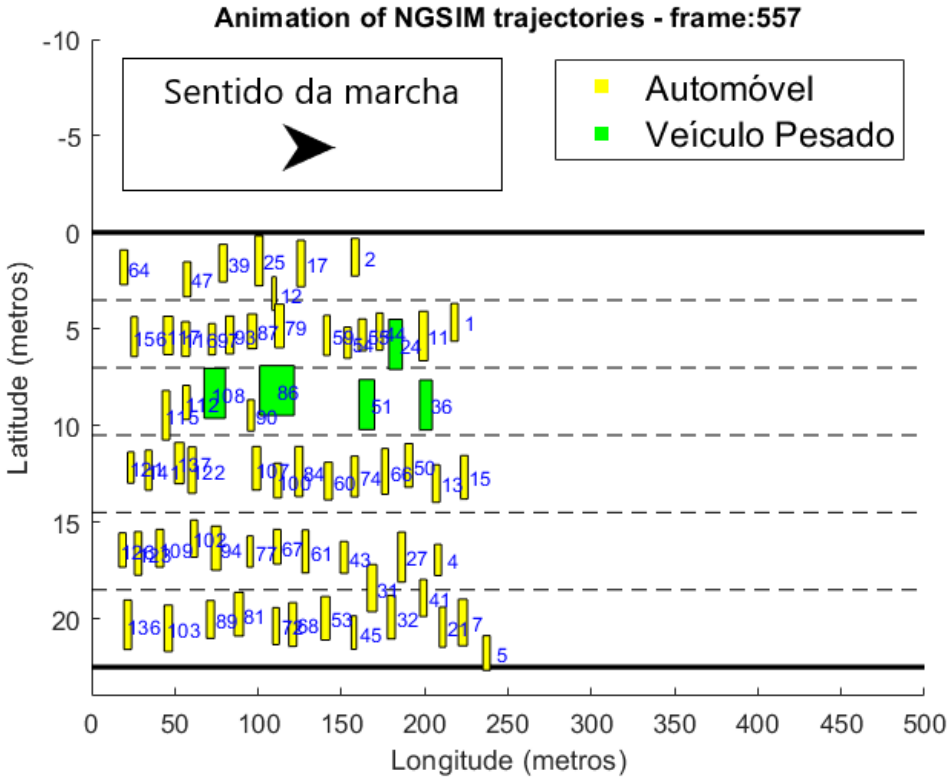


Figura 3.3: *Frame* 557 obtido com a compilação do código de visualização

Após a compilação do código com as várias bases de dados, foi determinado que o ficheiro composto pelos percursos entre 16h:00m e 16h:15m seria o melhor a ser utilizado. Esta conclusão deveu-se à menor congestão de veículos e por isso, mais variedade de valores dependendo do número da via. Enquanto que nos outros ficheiros, todas as vias tinham velocidades idênticas, neste em particular, as vias mais à esquerda apresentavam velocidades mais elevadas que as vias mais à direita e mudanças de faixa que seriam mais fáceis de prever, com os tipos de acelerações e espaçamentos entre veículos.

3.2.3 Análise dos parâmetros

A simulação das trajetórias ajudou bastante na escolha do melhor ficheiro para este tipo de projeto, porém, há a necessidade de se compreender melhor determinados valores, influências entre eles e determinadas situações como mudança de faixa. Para essa análise mais detalhada, foi desenvolvido um código que seleciona determinados parâmetros que se achem mais relevantes e permite ao utilizador observar analogias entre parâmetros com gráficos.

A utilização de 1 048 575 observações dificultou o desenvolvimento constante do projeto devido às capacidades do *hardware* que, neste caso, não eram as melhores para a compilação do código, mais especificamente, as características deste hardware são:

- Marca e modelo: MSI - GL62M 7REX
- Processador: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
- Memória RAM: 8GB DDR4
- Placa gráfica: NVIDIA GeForce GTX 1050 Ti
- Armazenamento: 256GB SSD + 1TB

O tempo de compilação foi bastante prolongado e uma das tarefas estipuladas era que fossem aplicadas técnicas e algoritmos pouco dispendiosos de tempo e eficazes. Por isso, e de forma a tornar mais rápido estes processos, resolveu manipular-se a base de dados de maneira a apresentar apenas as trajetórias das duas faixas que mostram ser mais fáceis para ocorrer ultrapassagens e haver maior distinção de valores de variáveis entre elas. Com esta ideia inicial foram determinadas as velocidades e acelerações médias em cada faixa, visto que estes dois parâmetros são dos principais e dos mais influentes para a previsão de trajetórias. Na Figura 3.4 está retratado um gráfico com as velocidades médias por cada faixa existente. É possível observar uma elevada desigualdade de valores entre a faixa 1 e a faixa 2 que, neste processo de seleção, é o que se procura entre faixas vizinhas.

Para além da velocidade, foi preciso visualizar os valores de aceleração para consolidar a escolha das duas melhores faixas. No gráfico apresentado na Figura 3.5 verificam-se vizinhanças com valores bastante distintos, comparado com o gráfico da velocidade anteriormente exposto.

Entre a faixa 6 e a faixa 5 é onde se observa a maior diferença de valores. A diferença da aceleração torna-se bastante percetível visto que a faixa 6 se trata de uma faixa para a entrada e saída de veículos e sendo também um horário de congestão, as faixas mais ao centro tornam-se mais lentas e podem chegar a ter uma média de aceleração negativa, como se pode perceber nas faixas 3, 4 e 5. Quanto à faixa 1 e 2, percebe-se que sendo a faixa 1 a mais rápida, os veículos que a utilizam

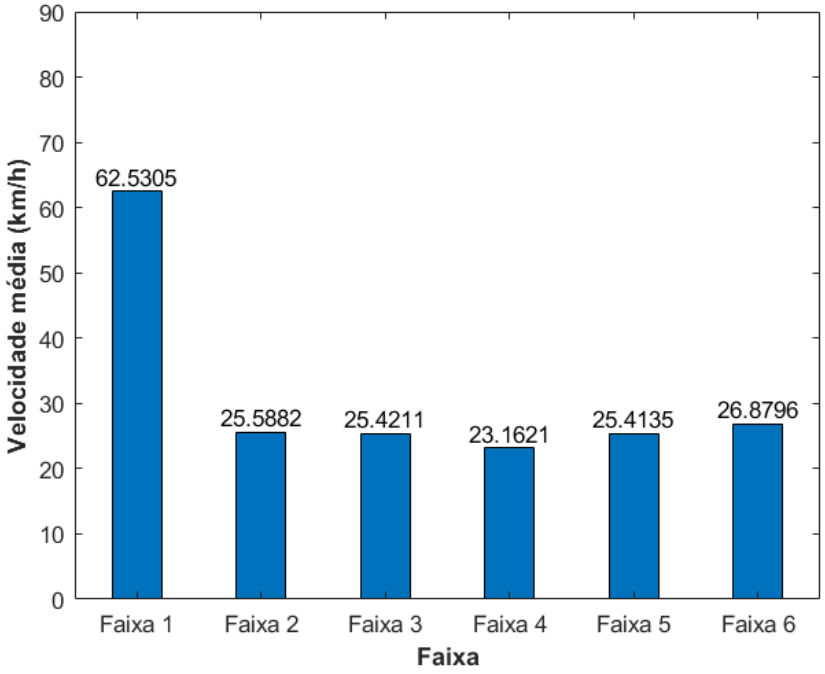


Figura 3.4: Velocidade média em cada faixa da I-80

ou que se mudam para essa faixa necessitam de maiores acelerações e sendo a faixa 2, uma mais central, terá valores médios mais perto de 0. Desta forma, e com a análise do gráfico da velocidade média e da aceleração média, foram escolhidas todas as observações pertencentes às faixas 1 e 2.

A seleção das duas faixas mais à esquerda permitiu assim reduzir consideravelmente o número de observações de 1 048 575 para 286 620 e o número de veículos de 1725 para 650. De notar que a nova quantidade de observações a utilizar continua suficiente para se aplicar técnicas e algoritmos de forma a se obter resultados bastante positivos para a previsão de trajetórias.

Após a redução da base de dados foi novamente feito um estudo dos vários parâmetros e comportamentos dos veículos, mais precisamente dos veículos da faixa 1 e da faixa 2 individualmente e dos veículos que alternam de faixa.

Intervalo de velocidades

Para ser feita uma comparação precisa, foi necessário realizar uma separação das observações por faixas. Esta separação de dados consiste na formação de uma nova matriz com os veículos que utilizam apenas a faixa 1, veículos que utilizam apenas a faixa 2 e veículos que utilizam ambas as faixas. Desta forma consegue entender-se de melhor como atuam os veículos de cada faixa. Na Figura 3.6 estão esboçados histogramas da quantidade de observações por intervalo de velocidade para os veículos que circulam apenas na faixa 1, apenas na faixa 2 e em ambas as faixas.

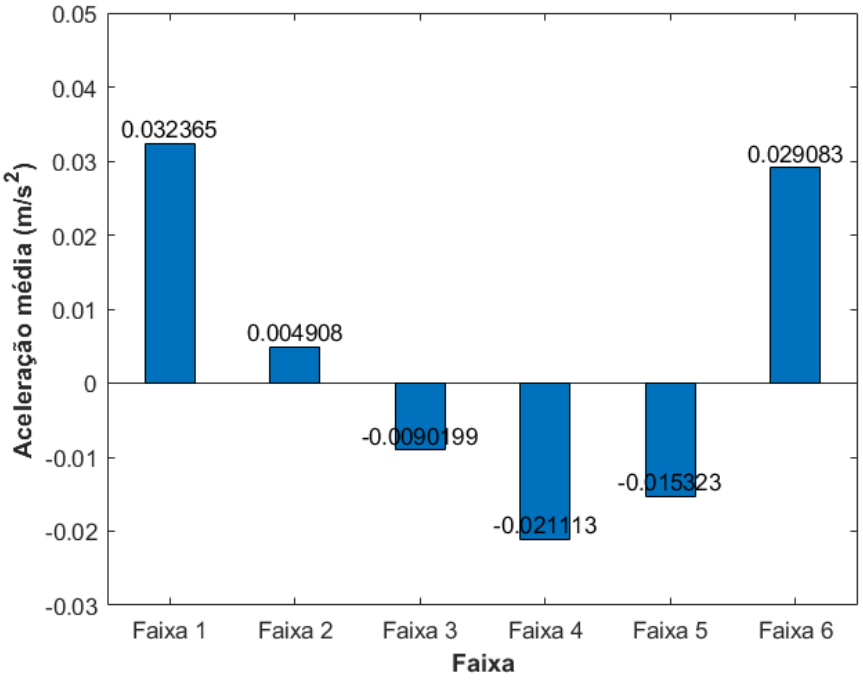


Figura 3.5: Aceleração média em cada faixa da I-80

Como se pode observar, existe uma grande diferença na quantidade de observações e na distribuição de velocidades entre os veículos que utilizam apenas a faixa 1, a faixa 2 ou que utilizam ambas as faixas.

Mais à esquerda está representado o gráfico referente à faixa 1. Neste gráfico, pode visualizar-se que o maior intervalo de velocidade para os veículos que utilizam apenas a faixa 1 é entre os 60 km/h e os 70 km/h. Apesar do intervalo entre estas velocidades ser visto como o intervalo com o maior índice de observações, não existe uma grande dispersão na quantidade das mesmas entre as velocidades de 40 km/h e 90 km/h.

Quanto ao gráfico ao centro, estão expostos os dados dos veículos que utilizam apenas a faixa 2. Ao contrário do primeiro gráfico, existe uma diferença substancial no que toca ao número de observações tanto entre intervalos de velocidade como valores totais. Percebe-se que existem poucos momentos em que os veículos ultrapassam a velocidade dos 40 km/h, estando as observações compostas principalmente por velocidades entre os 10 km/h e os 40 km/h. As baixas velocidades das viaturas resultam em tempos de passagem no troço de estudo superior aos veículos da faixa 1, que andam normalmente a velocidade superiores, e por isso, é que há um número tão elevado de observações correspondentes à faixa 2.

Terminando com o gráfico dos veículos que utilizam ambas as vias, entende-se que a quantidade de viaturas que mudam de faixa é significativamente pequena em

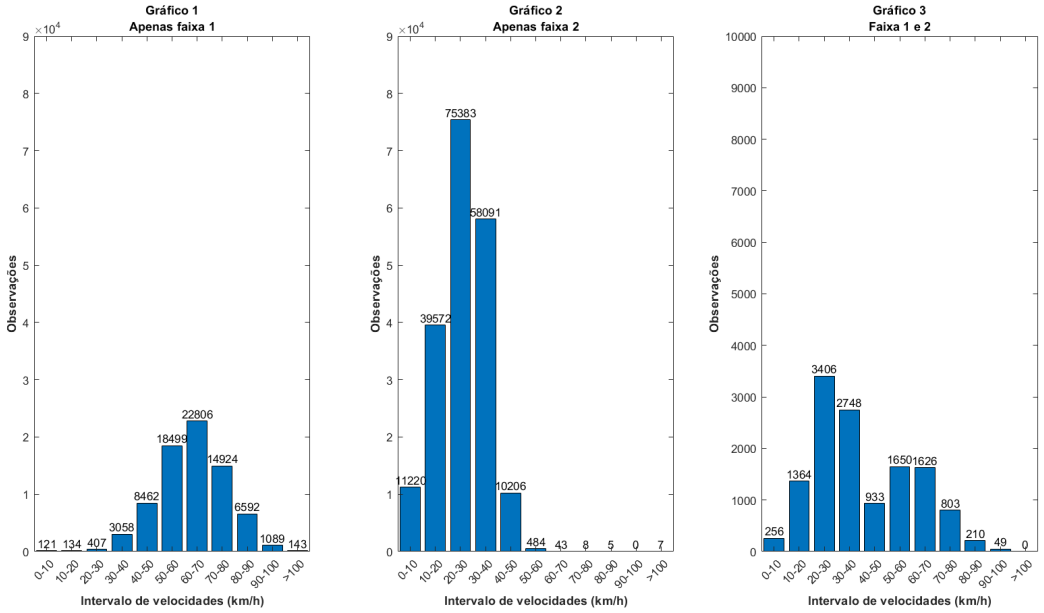


Figura 3.6: Distribuição de velocidade em cada faixa

comparação com os restantes veículos que não mudam de via. Ainda assim, percebe-se que a distribuição de observações por intervalo de velocidade é mais uniforme visto que foram submetidos a 2 tipos diferentes de movimentos no troço da I-80. De notar que a escala em Y deste último gráfico foram alterados para melhor interpretação.

Com estes gráficos é possível determinar entre que valores de velocidade um veículo pode tomar consoante o tipo de via que ocupa, porém, as trajetórias são compostas por diferentes parâmetros, cada um com uma determinada importância. Assim, serão apresentados e explicados outros parâmetros que se acham pertinentes como a aceleração, o espaçamento entre um determinado veículo e o veículo em frente e seu tempo.

Intervalo de acelerações

Perceber o intervalo de valores da aceleração em que normalmente os veículos atuam, pode ajudar numa futura previsão de trajetórias. Com a análise da Figura 3.5 já se teve conhecimento sobre a média de acelerações por cada faixa, sendo a faixa 1, como previsto, a com maior média positiva. Ainda assim, pretende-se conhecer melhor este parâmetro e por isso, a Figura 3.7 apresenta três gráficos com o número de observações por cada intervalo de acelerações dos três novos conjuntos de dados.

Os três gráficos apresentam o mesmo padrão, estando a maioria das observações entre os intervalos de acelerações de -1 (m/s^2) a 1 (m/s^2), porém existem diferenças nos gráficos que informam ao programador de que a faixa 1 é uma faixa mais ativa no que toca a acelerações e desacelerações. Como se pode observar no gráfico 1, e ao contrário do gráfico 2, o número de observações dos intervalos de -4 a -3 e 3 a 4

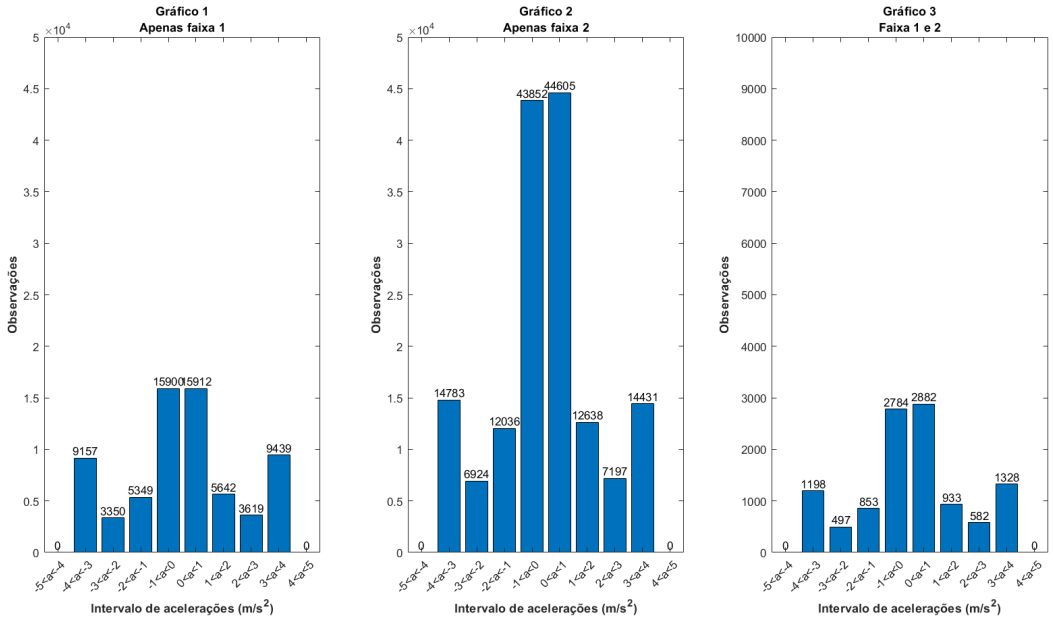


Figura 3.7: Distribuição de aceleração em cada faixa

é mais de metade das observações dos intervalos de -1 a 1. Esta leitura permite dar a conhecer de que os veículos maioritariamente fazem acelerações ou desacelerações pouco intensas ou bastante intensas.

Quanto ao gráfico 2, percebe-se rapidamente que a maioria se encontra entre os intervalos de tempo com acelerações ou desacelerações muito baixas e que a dispersão de observações pelos diferentes intervalos de acelerações estão distribuídas mais ou menos uniformemente, sendo que o número de observações mais alto sem a inclusão dos dois intervalos mais preenchidos, é um terço dos intervalos com mais observações. No terceiro gráfico o padrão é igual aos dois gráficos anteriores, no entanto a diferença entre os intervalos de valores de -5 a -4 e 4 a 5 para os intervalos de valor com mais observações é de metade, sendo possível supor que as observações dos intervalos com maiores acelerações e desacelerações sejam de quando o veículo muda de trajetória.

Espaçamento entre veículos

Outro parâmetro que é de certa forma importante para depois ser utilizado num algoritmo é o espaçamento entre veículos. Esta informação permite dar a conhecer ao utilizador sobre que valores são mais testemunhados em todas as observações recolhidas. Na Figura 3.8 podem ser verificados três gráficos sobre o espaçamento entre veículos que utilizam só a faixa 1 ou 2 e os que utilizam ambas as faixas.

Após a análise dos gráficos percebe-se que o distanciamento entre veículos que utilizam a faixa mais à esquerda é principalmente superior a 55 metros. Este valor deve-se à ocorrência de velocidades superiores nesta via, que é a via mais rápida, e

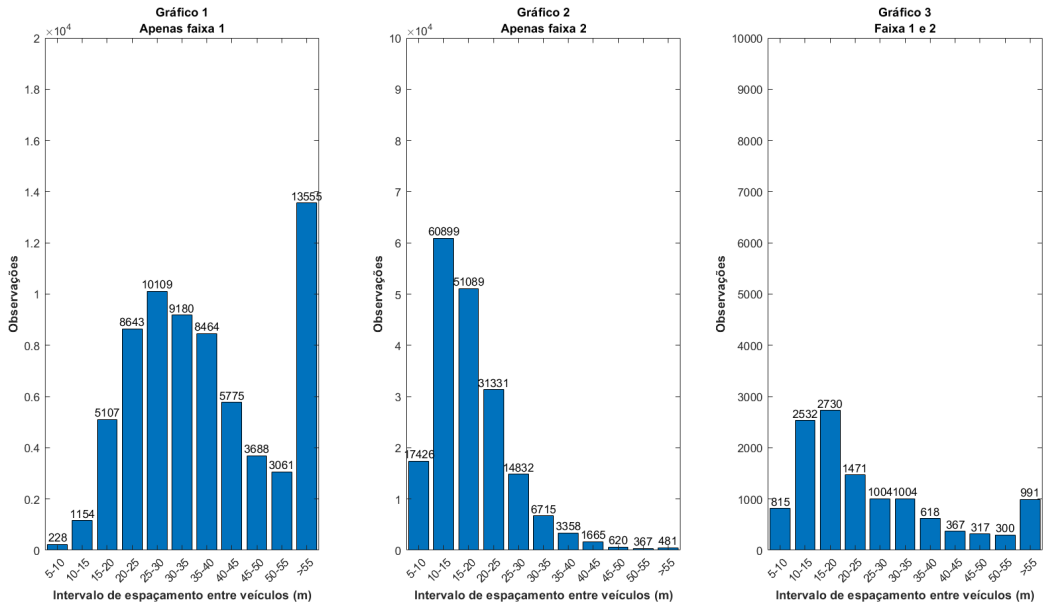


Figura 3.8: Distribuição do espaçamento entre veículos em cada faixa

por isso, os condutores aumentam a sua distância de segurança. Ainda assim, são várias as observações em que os veículos estão entre os 20 e os 40 metros de distância, valores razoáveis visto que, na maioria das observações, a velocidade nesta via é entre os 60 km/h e os 70 km/h.

Com o estudo do gráfico 2 entende-se que a maioria dos veículos deixa um espaço entre os 10 e os 20 metros de distância do veículo em frente. A elevada congestão e consequentemente, as baixas velocidades, fazem com que os condutores necessitem de menos distância de segurança e por isso, apresentam baixos valores de distâncias entre eles. Já no gráfico 3, com a junção de veículos que utilizam ambas as vias, é normal que haja uma dispersão de observações mais homogénea, apresentando pouco mais de um terço do intervalo com maiores observações, o intervalo com espaçamento superior aos 55 metros.

Tempo entre veículos

Para terminar a leitura dos parâmetros que se acham mais cruciais para o desenvolvimento de um algoritmo de previsão de trajetórias, está o parâmetro do tempo que decorre entre a passagem de um veículo e o seu precedente.

Tal como os parâmetros anteriores, vão ser amostrados e analisados 3 gráficos referentes aos veículos que utilizam apenas a faixa 1 e 2 e os veículos que utilizam ambas. Os gráficos em questão estão representados na Figura 3.9.

Ao contrário dos parâmetros apresentados anteriormente, os três gráficos revelam informações muito parecidas. Após uma verificação à distribuição de observações, pode concluir-se que os veículos estão geralmente intervalados entre 2 e 6 segundos.

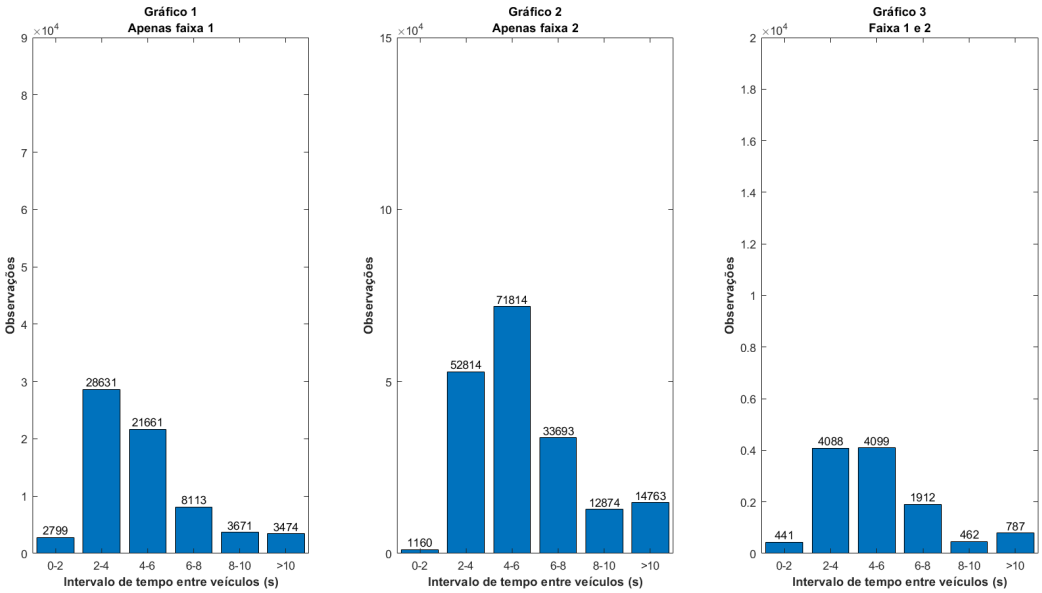


Figura 3.9: Distribuição do tempo entre veículos em cada faixa

A causa deve-se provavelmente à velocidade percorrida em cada faixa, isto é, como observado no gráfico anterior, na faixa 1 existe um maior espaçamento entre veículos, porém, e com informações obtidas no gráfico 1 da Figura 3.6, as velocidades são também superiores. Com isto, consolida-se a teoria de que os veículos da via 1 percorrem o trajeto a velocidades muito idênticas, acontecendo o mesmo para os veículos da via 2 e os que utilizam ambas as vias.

3.2.4 Parâmetro distância lateral

A leitura dos vários parâmetros apresentados anteriormente deu para perceber de forma clara como a maioria dos veículos se move no troço de 500 metros da auto-estrada. Ainda assim, achou-se pertinente adicionar novos parâmetros que não existiam antes e que podem ser cruciais para a obtenção de melhores resultados. Os parâmetros em questão é a distância a que um determinado veículo está em relação aos veículos da outra faixa e, no caso de ser um veículo que muda de faixa, a determinação da distância lateral em relação ao veículo da nova faixa quando ocorre a mudança da mesma. Estes parâmetros permitem fornecer ao programador informação sobre a distância normal a que um veículo se encontra quando pretende fazer uma mudança de trajetória e que podem ser posteriormente aplicados em técnicas e algoritmos.

Para a determinação deste parâmetro foi realizada uma leitura *frame* a *frame* do novo conjunto de dados referentes às duas faixas apenas. Uma vez que por *frame* estão amostrados vários veículos, é realizado um processo de leitura e cálculo individualmente a cada veículo.

Primeiramente é percebida a faixa em que o veículo se encontra e após se obter essa informação, são analisados sequencialmente todos os veículos que se situam na faixa lateral e com um processo de verificação e seleção, é determinada a menor distância ao veículo de estudo naquele instante. Dependendo da via em que se encontra, a distância obtida é adicionada numa nova coluna da matriz de dados. No caso deste estudo, foi criada a coluna 19 para a distância à esquerda e a coluna 20 para a distância à direita.

Depois de processados todos os dados das distâncias laterais, pretendeu-se obter os valores de quando é mudado o valor da variável *Lane_ID*. Este processo é mais simples e menos demorado que o processo anterior, sendo feita uma análise individualmente a cada veículo e uma leitura de cada observação. Caso haja mudança de valor no parâmetro da faixa, é sinalizada a observação anterior e colocado numa nova coluna, neste caso, na coluna 21, o valor da distância da coluna 19 ou 20. Na Figura 3.10 pode ser visualizada a distribuição de observações por intervalo das distâncias laterais.

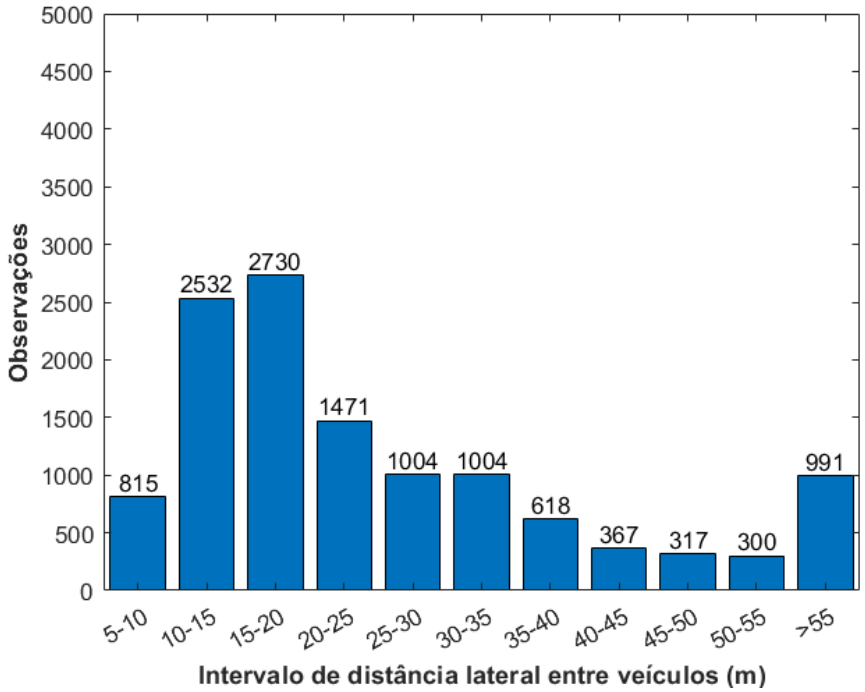


Figura 3.10: Distribuição da distância lateral quando há mudança faixa

Analisando o gráfico, é possível verificar que a distância lateral quando ocorre uma mudança de faixa foca-se principalmente entre os valores de 10 a 20 metros, um resultado expectável para as velocidades com que os veículos circulam.

3.3 Pré-processamento e escolha de parâmetros

O desenvolvimento prático de um projeto que funciona à base da inteligência artificial é por vezes demorado e pode chegar a ser bastante complexo consoante a sua área e metodologia. Para facilitar o desenvolvimento de um algoritmo capaz de cumprir os objetivos do programador, ajuda seguir um conjunto de processos, nomeadamente uma recolha de dados, uma planificação e uma preparação dos mesmos, a escolha do melhor algoritmo a utilizar com a realização de treinos, testes e ajustes e por fim, a aplicação do algoritmo no produto final.

De forma a dar início ao corpo do projeto, foi feita a escolha de um veículo que cumprisse os critérios, neste caso, que fizesse mudanças de faixas e que apresentasse bons valores nos diferentes parâmetros de que é composto para depois serem submetidos aos algoritmos desenvolvidos.

A escolha dos dados é determinante e, por isso, é aconselhável fazer uma visualização das várias trajetórias e definir assim o melhor conjunto de variáveis referentes a um determinado veículo antes de se prosseguir com a apresentação dos algoritmos.

Para a observação das várias trajetórias, foi usado o código de visualização mas com ligeiras alterações. As suas modificações foram consequência da redução de veículos para apenas os que utilizam a faixa 1 e 2 do troço e, com uma análise às várias viaturas, foi determinado como veículo de teste, o veículo com o número de identificação 757. A transição de faixas e as condições em que é submetido, neste caso, velocidades e distâncias dos veículos em redor foram determinantes para a sua escolha. Na Figura 3.11 pode ser identificado o veículo em questão.

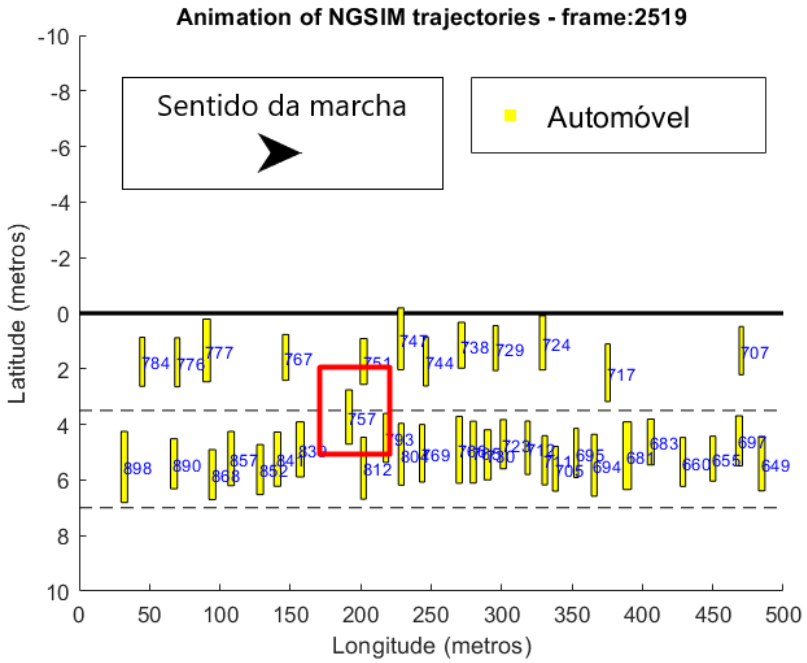


Figura 3.11: Identificação do veículo escolhido no *frame* 2519

Depois de determinado o veículo para os seus dados serem futuramente submetidos a treinos com os algoritmos, foram visualizados dois dos parâmetros que mais interessam neste tipo de cenários. As variáveis são diretamente a localização e a velocidade da viatura. A previsão da futura localização é o objetivo principal deste projeto, sendo que a velocidade será um parâmetro de apoio para consolidar a qualidade dos algoritmos Na Figura 3.12 está exibido o gráfico correspondente à localização do veículo no troço e na Figura 3.13 o gráfico correspondente à velocidade do mesmo veículo.

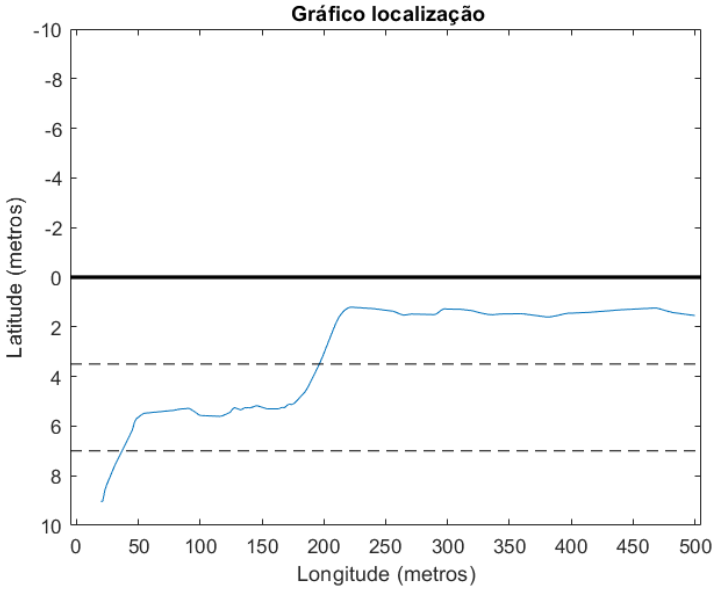


Figura 3.12: Gráfico da localização do veículo no troço

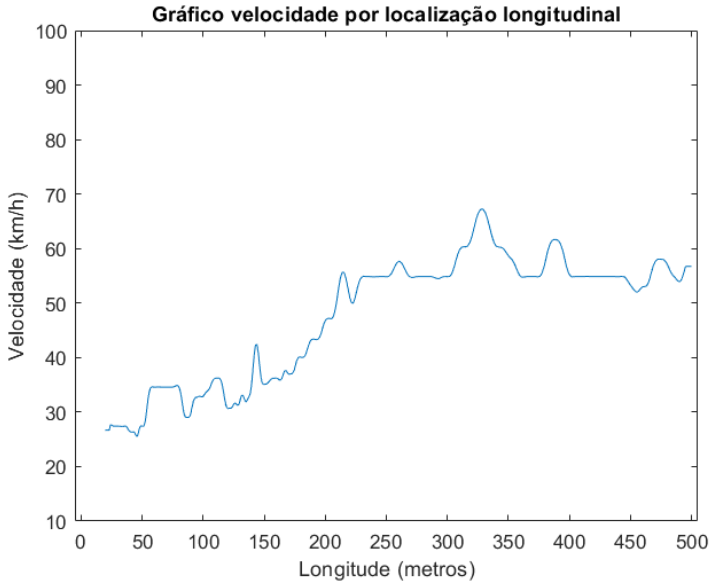


Figura 3.13: Gráfico da velocidade do veículo no troço

3.4 Arquitetura dos Algoritmos

A arquitetura de uma rede neuronal pode variar bastante consoante o tipo de tarefa a que vai ser sujeita e, com a previsão de valores futuros de trajetórias como objetivo deste projeto, é necessário a utilização de camadas capazes de tal. Neste caso, serão explicadas as várias configurações, as várias camadas utilizadas e a composição das arquiteturas, em código MATLAB, como está explícito no Anexo A.

3.4.1 Configurações e opções de treino

Na programação de um algoritmo de inteligência artificial existem várias opções que podem colocar em causa a sua qualidade. A escolha da melhor quantidade de dados para treino e para teste e a quantidade de neurónios em algumas camadas são alguns dos fatores que influenciam bastante a aptidão do algoritmo.

As configurações determinadas para complementar o algoritmo são, portanto, as seguintes:

- **Percentagem de dados para treino:** 75%;
- **Percentagem de dados para teste:** 25%;
- **Número de entradas (numEntradas):** 1, Localização X e Velocidade, separadamente;
- **Número de saídas (numRespostas):** 1, Localização X e Velocidade, separadamente;
- **Número de camadas ocultas:**
  - **Camada oculta 1 (numCamadas1):** 300;
  - **Camada oculta 2 (numCamadas2):** 100;
- **Probabilidade *dropout* (*dropoutProb)*:** 0.2.

Uma vez que estão feitas as configurações iniciais para a distribuição de dados e para as várias camadas que se irão utilizar, é necessário incluir as opções de treino, caso contrário, ocorrerão problemas de processamento. Uma vez que durante o treino e obtenção de resultados, serão feitas modificações, iniciou-se o teste dos algoritmos com as seguintes opções de treino: escolha da função *Adaptive Moment Estimation* (ADAM) para otimizar o treino, colocação de um limite de iterações com a opção *MaxEpochs* para 200, e para visualizar o treino da rede, a colocação da opção *plot* para *training-progress*. A definição destas opções está visível na Figura 3.14.

```
%% Opções de Treino

options = trainingOptions('adam', ...
            'MaxEpochs',200, ...
            'Plots','training-progress');
```

Figura 3.14: Opções de treino em MATLAB

3.4.2 Camadas

A composição dos vários modelos desenvolvidos resume-se à utilização das camadas *Sequence Input*, *Dropout*, *LSTM*, *Bidirectional Long Short-Term Memory* (Bi-LSTM), *Fully Connected Layer* e *Regression*.

- ***Sequence Input:*** prepara uma sequência de dados consoante o número de entradas definidas (numEntradas). Na maioria dos modelos, com o propósito de prever valores futuros, esta é a a camada inicial, uma vez que prepara os dados sequencialmente para as próximas camadas;
- ***Dropout:*** é uma camada que define aleatoriamente os neurónios que serão ignorados durante o processo de treino consoante a probabilidade estabelecida (*dropoutProb*), ajudando a evitar *overfitting*, ou seja, ter menos desvios causados por erros e uma reta de previsão mais fluída;
- ***LSTM*:** camada composta por vários neurónios e que é bastante utilizada na área de inteligência artificial. Possui conexões de *feedback* e a sua configuração traduz-se resumidamente pelo número de neurónios que o programador pretende (*NumCamadas*);
- ***Bi-LSTM:*** camada que utiliza duas LSTM, cada uma com dados de diferentes estados. Neste tipo de rede, a saída pode obter informações de estados passados e futuros simultaneamente sendo a sua configuração idêntica à rede *LSTM*, onde é apenas necessário colocar o número de camadas ocultas (*NumCamadas*) a utilizar.
- ***Fully Connected Layer:*** é uma camada colocada normalmente nos finais dos modelos de redes neuronais, depois de uma camada *LSTM*, por exemplo. Esta camada cria conexões com os neurónios da camada anterior e combina-os de forma a obter, dependendo da escolha do programador, um determinado número de saídas. Neste projeto, é necessário incluir o número de respostas (numRespostas), que acaba por ser o número de saídas do modelo;
- ***Linear Regression:*** utilizada como camada final em todos os modelos que irão ser desenvolvidos, a *Linear Regression* calcula o erro quadrático médio

(RMSE) entre os valores pretendidos e obtidos e não necessita de qualquer variável de entrada.

3.4.3 LSTM-1

A arquitetura LSTM-1 é das mais simples a ser utilizada, isto porque apresenta apenas 4 camadas principais, Figura 3.15. Como todos os modelos, inicia-se com uma *Sequence Input* e, até à *Linear Regression*, são compostas 2 camadas diferentes, uma primeira camada *LSTM* composta por 300 neurónios e uma camada *Fully Connected Layer*.

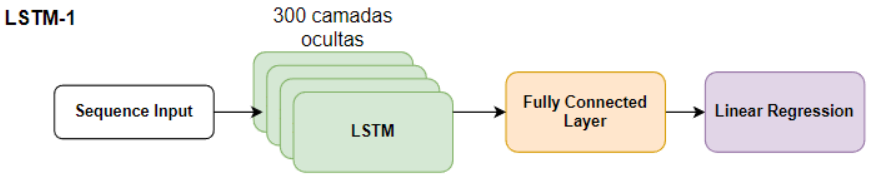


Figura 3.15: Arquitetura LSTM-1

3.4.4 LSTM-2

A LSTM-2, como o nome indica, é composta por duas camadas *LSTM*, Figura 3.16. Estas duas camadas estão sequencialmente ligadas após a camada de *Sequence Input*, e termina com a camada *Fully Connected Layer* e uma *Linear Regression*. De notar que como no modelo LSTM-Drop-2, as *LSTM* apresentam valores diferentes de neurónios, com 300 na primeira e 100 na segunda.

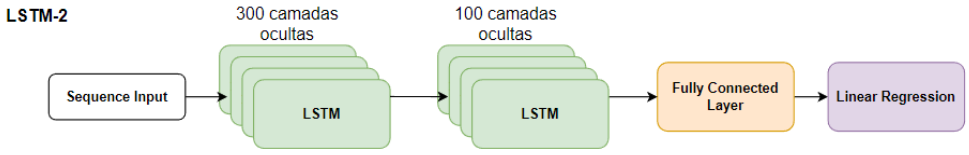


Figura 3.16: Arquitetura LSTM-2

3.4.5 LSTM-Drop-1

O modelo LSTM-Drop é um modelo desenvolvido com foco principal na camada *LSTM* e na camada *Dropout* (Figura 3.17). Inicialmente é encontrada uma camada de entrada que redimensiona os dados e os prepara para a seguinte que, neste modelo, é uma camada *Dropout*. Esta camada, como foi explicada anteriormente, anula aleatoriamente uma quantidade de neurónios consoante a percentagem colocada. Posteriormente os dados são submetidos a uma camada *LSTM* com 300 neurónios,

sendo depois conectada a uma camada *Fully Connected Layer* e termina com uma *Linear Regression*.

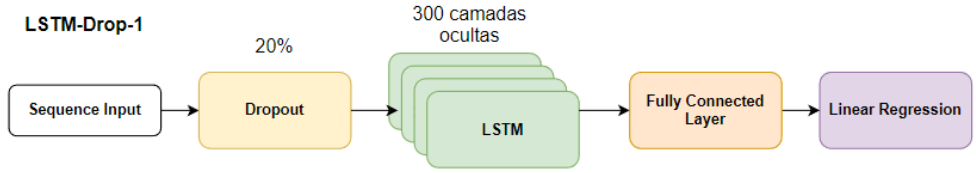


Figura 3.17: Arquitetura LSTM com *dropout*

3.4.6 LSTM-Drop-2

O modelo LSTM-Drop-2 é composto pelas mesmas camadas apresentadas no modelo anterior, porém, a sua quantidade é diferente (Figura 3.18). Ao invés de ocorrer a ligação entre a camada *LSTM* e a camada *Fully Connected Layer*, como na arquitetura anterior, existe mais uma camada *dropout* com a mesma probabilidade de atualização de neurónios para nulo e uma camada *LSTM* com, neste caso, 100 neurónios.

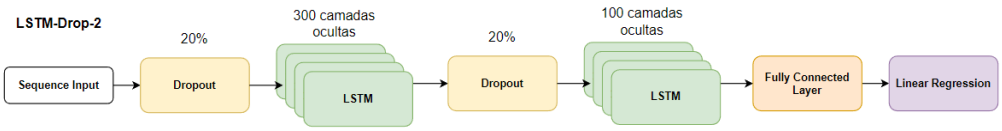


Figura 3.18: Arquitetura LSTM-Drop-2

3.4.7 Modelos Bidirecionais

Depois de desenvolvidas as arquiteturas com utilização a camadas *LSTM*, foram consideradas as camadas *Bi-LSTM*. Sendo a rede Bi-LSTM, uma rede que processa informação nos dois sentidos, podem ser obtidos melhores desempenhos por parte destes modelos. Desta forma, as arquiteturas Bi-LSTM-1, Bi-LSTM-2, Bi-LSTM-Drop-1, Bi-LSTM-Drop-2 estão apresentadas nas Figuras 3.19, 3.20, 3.21 e 3.22, respetivamente.

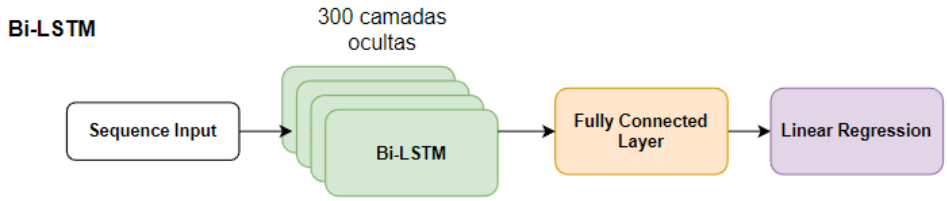


Figura 3.19: Arquitetura Bi-LSTM-1

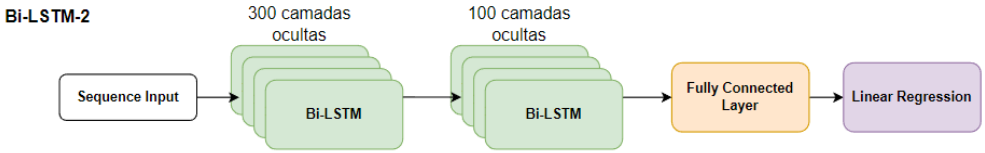


Figura 3.20: Arquitetura Bi-LSTM-2

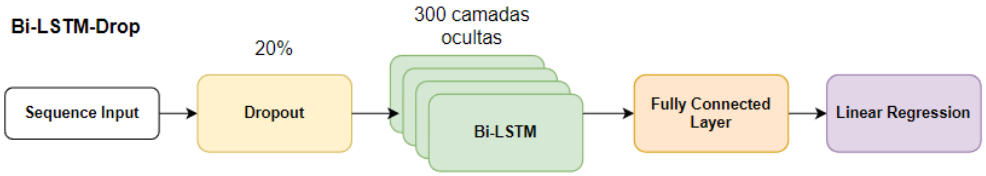


Figura 3.21: Arquitetura Bi-LSTM-Drop-1

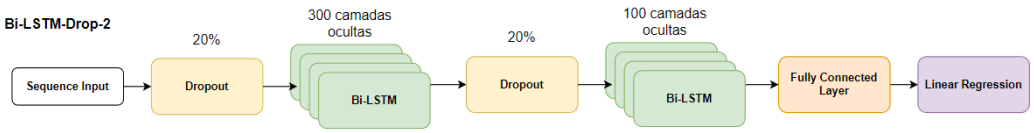


Figura 3.22: Arquitetura Bi-LSTM-Drop-2

3.4.8 Regression

Por fim está apresentado o modelo *regression*. Esta arquitetura é das mais simples a implementar e servirá como base de comparação para os modelos referidos anteriormente. Como se pode observar na Figura 3.23, este modelo inicia-se com uma *Sequence Input*, seguida de duas camadas *Fully Connected Layer*, onde a primeira é composta por 300 neurónios e a segunda pelo número de respostas, ou seja, 1, terminando assim com uma *Linear Regression*.

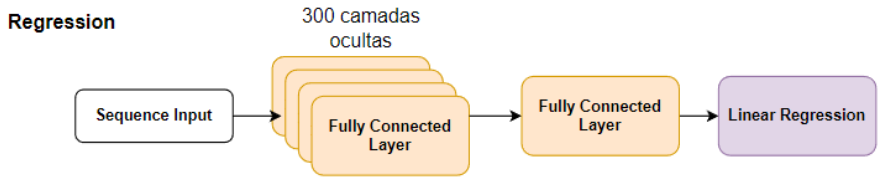


Figura 3.23: Arquitetura Regression

Capítulo 4

Testes e resultados

Este capítulo será principalmente focado na amostragem dos erros e dos resultados obtidos com cada modelo exemplificado no capítulo anterior.

4.1 Cálculo dos erros

Para ser desenvolvido um algoritmo eficiente, é necessário realizar um conjunto de treinos e testes com os vários modelos de redes neuronais em vista a obter primeiramente o RMSE de cada uma, que é definido pela diferença entre a saída calculada pelo algoritmo e o vetor de saída desejado.

Antes de ser calculado o erro, é necessário fazer um cálculo prévio do valor previsto, isto é, após a revelação do valor de previsão x^{prev}, é realizada uma multiplicação do valor obtido pelo desvio padrão dos dados para treino, considerado neste trabalho como σ e adicionada a média desses mesmos dados, μ. Esta equação, (4.1), é a seguinte:

$$x^{prev} = x^{prev}.\sigma + \mu \tag{4.1}$$

com σ representada na equação (4.2) e μ na fórmula (4.3), onde N é o número total de dados utilizados para treino:

$$\sigma = \sqrt{\frac{1}{N-1}.\sum_{i=1}^{N}(x_i^{treino} - \mu)^2}, \tag{4.2}$$

$$\mu = \frac{1}{N} . \sum_{i=1}^{N} x_i^{treino} \tag{4.3}$$

Quanto ao cálculo do erro final, este é obtido através da fórmula (4.4):

$$RMSE = \sqrt{\frac{1}{n} . \sum_{i=1}^{n} (x_i^{prev} - x_i^{real})^2} \tag{4.4}$$

onde n é o número total de dados para teste, x_i^{prev} é o resultado final dos valores previstos, e x_i^{real} o valor real do parâmetro em cada *frame* de teste.

A determinação de um modelo capaz de prever a localização de um veículo é o objetivo principal deste projeto e, por isso, foi realizado em primeiro lugar o cálculo do erro em relação à localização do veículo escolhido por *frame*. Na Tabela 4.1 estão apresentados os valores obtidos do erro de cada modelo pelo *frame* de previsão em atraso. O cálculo do erro foi determinado em duas diferentes situações, uma primeira em que o algoritmo não tem qualquer conhecimento dos dados de treino e faz uma previsão de valores num tamanho idêntico ao tamanho desse mesmo conjunto e uma segunda situação em que o algoritmo tem um atraso e por *frame*, faz um ajuste para que os dados previstos se adequem aos valores reais.

Tabela 4.1: Resultados do cálculo do erro RMSE, no parâmetro localização

	RMSE da localização (m)	
Modelos	**Sem conhecimento**	**Com conhecimento**
LSTM-1	0.1018	0.0319
LSTM-2	0.1043	0.0401
LSTM-Drop-1	0.1421	0.1217
LSTM-Drop-2	0.1056	0.1048
Bi-LSTM-1	1.5934	0.6941
Bi-LSTM-2	1.1182	0.4547
Bi-LSTM-Drop-1	1.1996	0.6249
Bi-LSTM-Drop-2	0.2336	0.2114
Regression	0.6538	0.0261

Como se pode observar nesta primeira tabela, existe uma grande variedade de valores entre os vários modelos testados. Neste caso, fez-se uma análise individualmente ao testes feitos com e sem conhecimentos dos dados reais e uma distribuição de cores nos três melhores resultados, mais precisamente a verde a melhor medida, a laranja a segunda melhor e a vermelho a terceira melhor medida.

Analisando a tabela, verifica-se que não existe um modelo capaz de obter o melhor resultado em ambas as situações simultaneamente, porém, determina-se com facilidade os modelos com menor erro em ambos os processos. Enquanto que as

arquiteturas que utilizam uma rede Bi-LSTM não mostram vantagens na sua utilização, pois em nenhuma das situações, qualquer um dos modelos não obteve nenhum dos três melhores resultados, os modelos constituídos por simples redes *LSTM*, sem utilização de uma camada *dropout*, obtiveram resultados bastante positivos.

O modelo LSTM-1 e o modelo LSTM-2 foram os mais bem sucedidos, com o modelo LSTM-1 a obter o melhor resultado sem conhecimento de qualquer dado real e o segundo melhor valor com conhecimento. Quanto ao modelo LSTM-2, este acaba por determinar a segunda melhor solução sem conhecimento dos dados de treino e o terceiro melhor valor com conhecimento. Quanto ao melhor valor calculado com conhecimentos dos dados de treino, este foi obtido com o processamento do algoritmo Regression. Para além destes dados estarem distribuídos por diferentes arquiteturas, entre os três melhores valores não houveram grandes dispersões, uma vez que os valores variaram entre os 0.1018 e os 0.1056 para os erros obtidos sem conhecimento dos dados de treino e entre 0.0261 e 0.0401 para os valores de erro com conhecimento dos dados de treino.

Como referido anteriormente, o cálculo prévio de valores da localização poderiam ser insuficientes para a escolha de um algoritmo e, por isso, incluiu-se o cálculo de valores futuros de velocidade, um parâmetro com maiores variações, para solidificar a escolha de uma arquitetura. Desta forma, na Tabela 4.2 estão listados os valores obtidos do erro em relação à velocidade do veículo.

Tabela 4.2: Resultados do cálculo do erro, RMSE, no parâmetro velocidade

	RMSE da velocidade (km/h)	
Modelos	**Sem conhecimento**	**Com conhecimento**
LSTM-1	2.7677	1.3800
LSTM-2	2.9699	1.2900
LSTM-Drop-1	2.2326	1.7014
LSTM-Drop-2	2.0177	1.6575
Bi-LSTM-1	8.5458	4.2671
Bi-LSTM-2	7.2188	4.3054
Bi-LSTM-Drop-1	8.5956	4.2901
Bi-LSTM-Drop-2	8.3113	5.0476
Regression	3.9919	0.9711

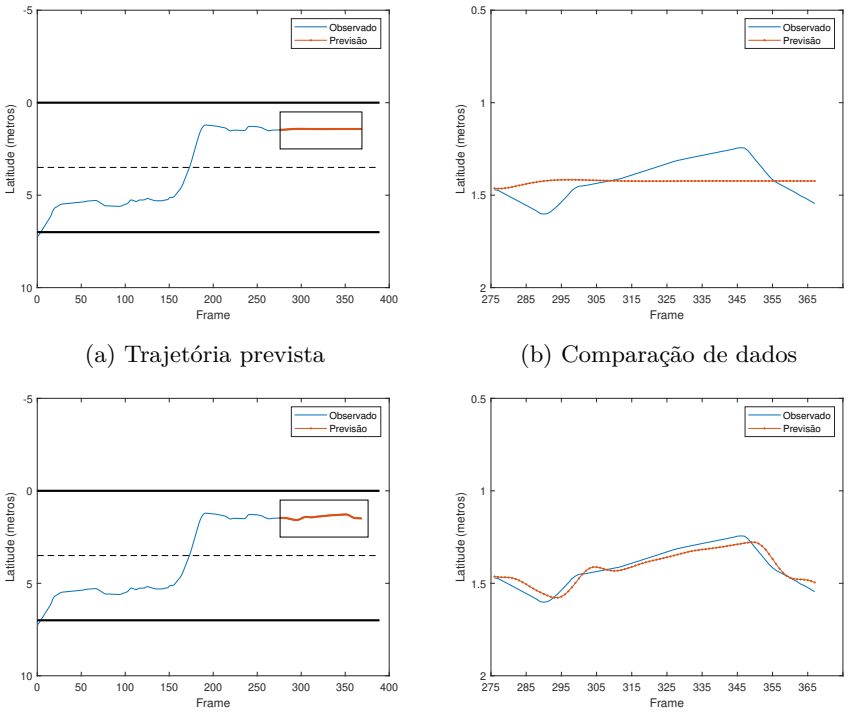
Feita uma análise à tabela anterior, vêm-se algumas semelhanças em relação à Tabela 4.1. Essas semelhanças são precisamente a distribuição de bons e maus resultados, ou seja, os maus resultados foram novamente calculados por parte das arquiteturas compostas por Bi-LSTM e os bons resultados obtidos pelas arquiteturas *LSTM* mais simples. Outra semelhança é o valor obtido pelo modelo Regression na situação com conhecimento de valores reais que, novamente, calculou o melhor resultado. Desta vez, a arquitetura LSTM-1 obteve o terceiro melhor valor em

ambas as situações e a arquitetura LSTM-2 foi a segunda melhor a prever os valores de velocidade com conhecimento dos dados.

Quanto aos cálculos de erro na situação em que os algoritmos não têm qualquer conhecimento de valores reais, o melhor resultado obtido foi através do modelo LSTM-Drop-2 e a segunda arquitetura mais bem sucedida foi a LSTM-Drop-1, não havendo ainda assim, muita grande dispersão entre os três melhores resultados.

4.2 Demonstração e discussão de resultados

Como todos os processos até agora desenvolvidos, a amostragem de resultados também foi realizada com o programa MATLAB. Estes resultados estão ilustrados em forma de gráficos, em que nas alíneas (a) e (c), a azul, estão apresentados os dados utilizados para treino das trajetórias dos veículos e a laranja a previsão proposta pelos algoritmos desenvolvidos e, nos gráficos laterais, (b) e (d), uma ampliação da zona selecionada e a comparação entre valores reais e valores previstos. Entre a Figura 4.1 e a Figura 4.9 estão representados os resultados obtidos com a implementação dos diferentes modelos utilizados.



(a) Trajetória prevista

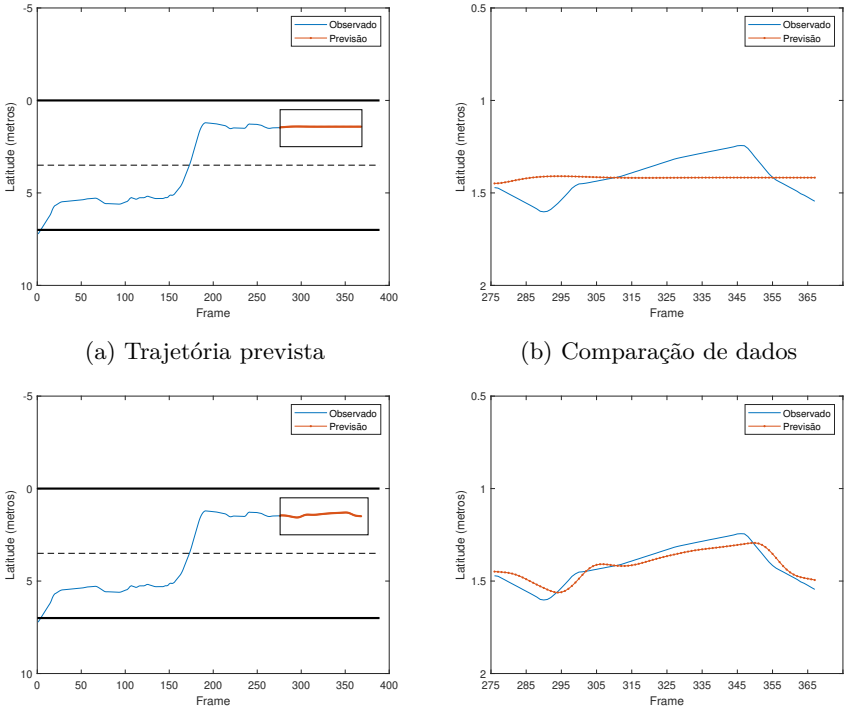
(b) Comparação de dados

(c) Trajetória prevista

(d) Comparação de dados

Figura 4.1: Modelo LSTM-1 sem conhecimento (a) e (b) e com conhecimento (c) e (d)

Com uma análise aos gráficos representados com as alíneas (a) e (c), compostos por 75% dos dados usados para treino e 25% dos dados previstos, e dos gráficos (b)



(a) Trajetória prevista

(b) Comparação de dados

(c) Trajetória prevista

(d) Comparação de dados

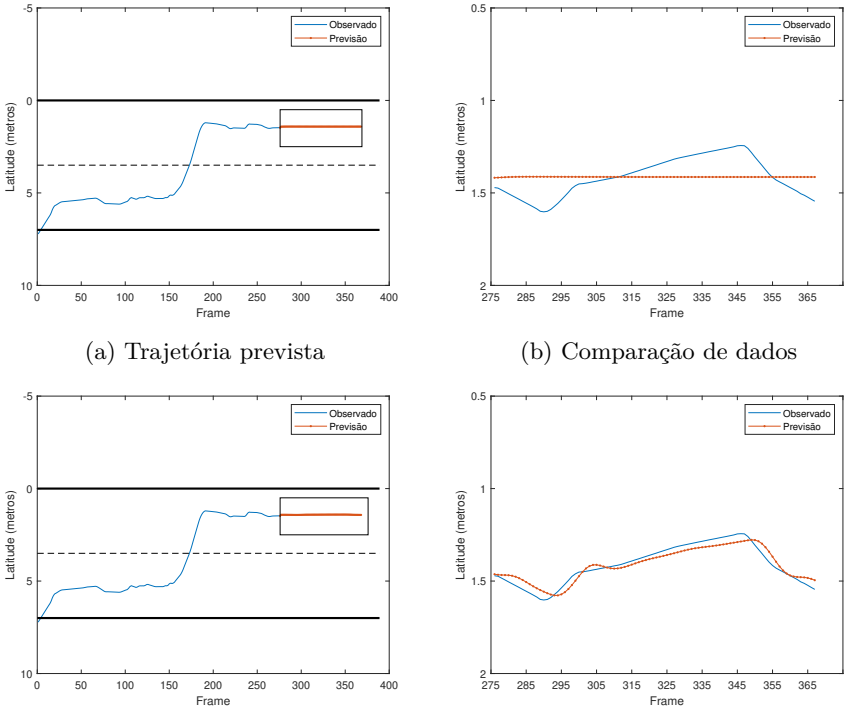
Figura 4.2: Modelo LSTM-2 sem conhecimento (a) e (b) e com conhecimento (c) e (d)

e (d), de comparação dos dados reais e dos dados previstos pelos algoritmos, ambos em função dos *frames*, percebe-se que com conhecimento dos dados reais, ainda que com atraso, os valores previstos seguem padrões muito idênticos, em comparação com as aplicações sem conhecimento de dados, onde os resultados tomam quase todos uma resposta linear, sem grandes variações.

De notar que na Figura 4.3b e 4.4b consegue notar-se o efeito da camada *dropout*. Enquanto que nas Figuras 4.1.b e 4.2.b observa-se uma ligeira curvatura nos *frames* iniciais. Nas arquiteturas compostas pela camada *dropout*, esta curvatura é atenuada.

Em relação aos gráficos associados aos algoritmos compostos por redes bidirecionais nas Figuras 4.5, 4.6 e 4.7, apesar de não ter ocorrido uma mudança de faixa, não foram obtidos os resultados mais adequados, ficando aquém nesta utilização e previsão mais simples de dados futuros. Porém, como se pode observar na Figura 4.8, com a arquitetura Bi-LSTM-Drop-2 resultaram valores bastante positivos em comparação com os outros modelos, algo que não seria de esperar e que poderá ser um tema de estudo num trabalho futuro.

Por fim e como modelo base de comparação entre algoritmos, a arquitetura Regression obteve resultados bastante positivos com conhecimento de valores reais. A sua previsão sem conhecimento de dados também não foge muito aos valores reais



(a) Trajetória prevista

(b) Comparação de dados

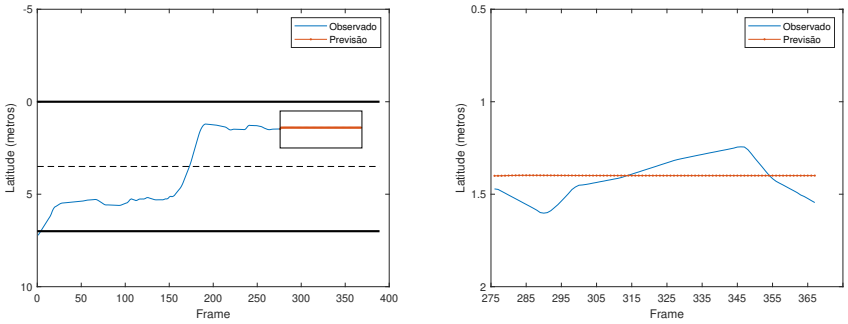
(c) Trajetória prevista

(d) Comparação de dados

Figura 4.3: Modelo LSTM-Drop-1 sem conhecimento (a) e (b) e com conhecimento (c) e (d)

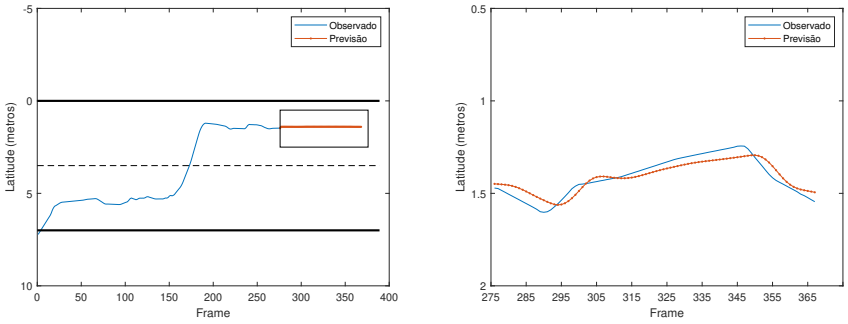
e tornou-se assim uma arquitetura, apesar de simples, bastante útil para comparação e referência de modelos.

Desta forma, e tendo como base de análise as Tabelas 4.1 e 4.2, compostas pelos erros obtidos com as simulações das várias arquiteturas e os gráficos das localizações apresentados desde a Figura 4.1 à Figura 4.9, consideram-se os modelos com rede LSTM e sem *dropout* como os melhores para a previsão de valores de localização, sem conhecimento. Quando se pretende fazer uma previsão com conhecimento dos dados reais, com atraso de resposta, o modelo Regression é a melhor escolha.



(a) Trajetória prevista

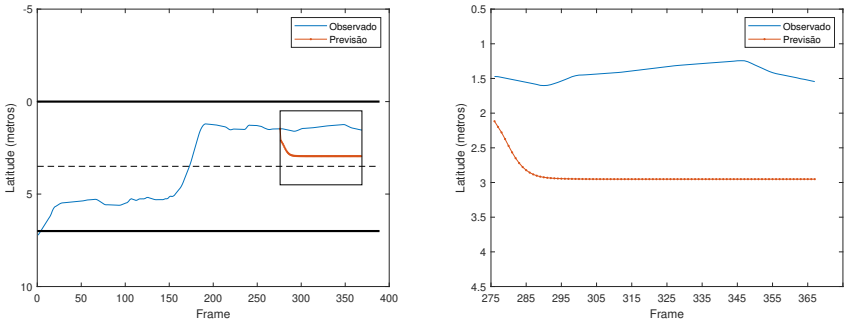
(b) Comparação de dados



(c) Trajetória prevista

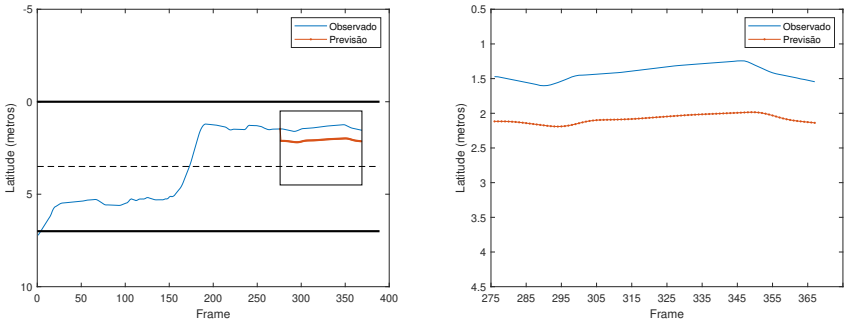
(d) Comparação de dados

Figura 4.4: Modelo LSTM-Drop-2 sem conhecimento (a) e (b) e com conhecimento (c) e (d)



(a) Trajetória prevista

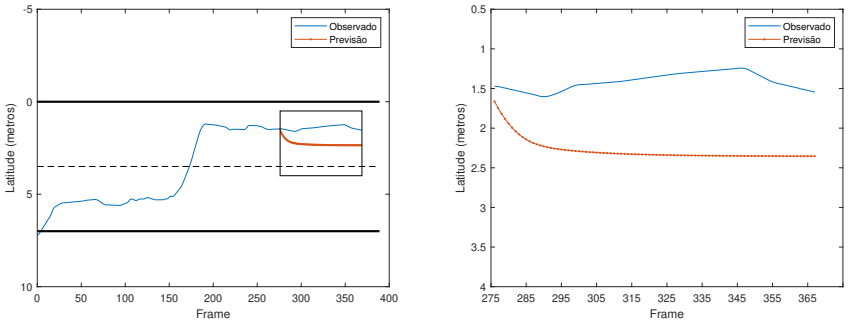
(b) Comparação de dados



(c) Trajetória prevista

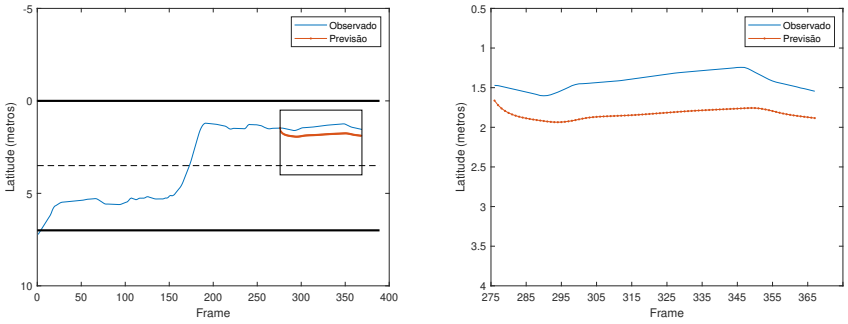
(d) Comparação de dados

Figura 4.5: Modelo Bi-LSTM-1 sem conhecimento (a) e (b) e com conhecimento (c) e (d)



(a) Trajetória prevista

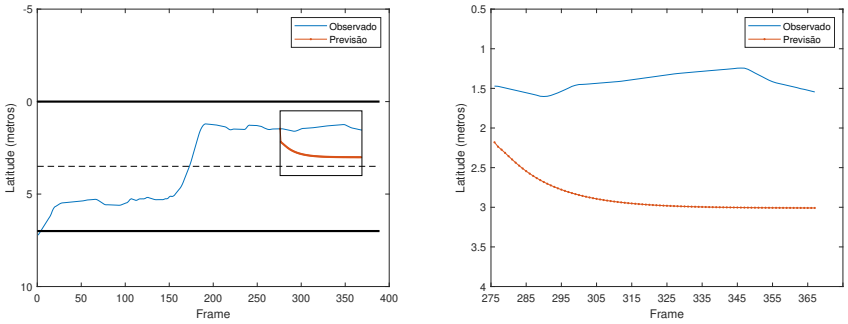
(b) Comparação de dados



(c) Trajetória prevista

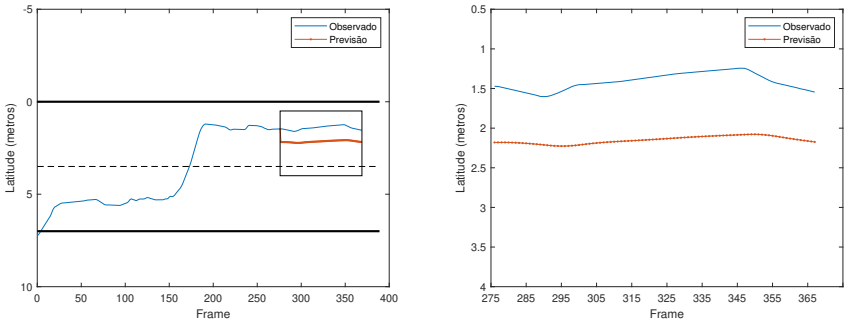
(d) Comparação de dados

Figura 4.6: Modelo Bi-LSTM-2 sem conhecimento (a) e (b) e com conhecimento (c) e (d)



(a) Trajetória prevista

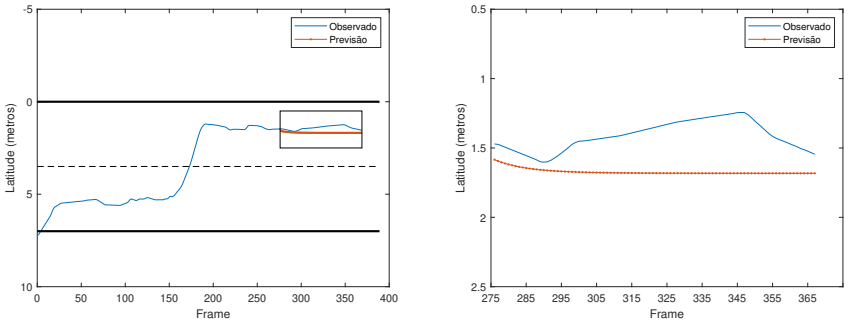
(b) Comparação de dados



(c) Trajetória prevista

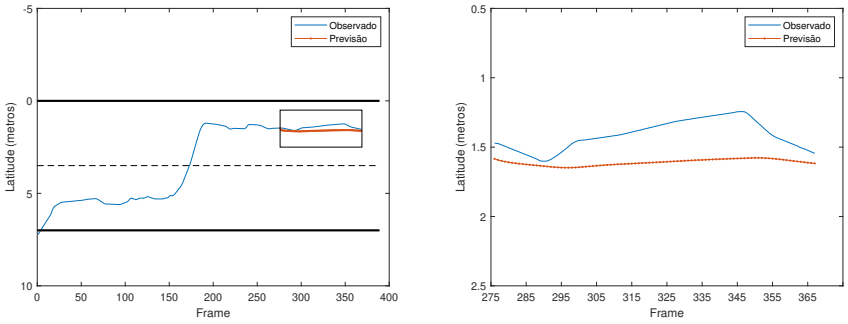
(d) Comparação de dados

Figura 4.7: Modelo Bi-LSTM-Drop-1 sem conhecimento (a) e (b) e com conhecimento (c) e (d)



(a) Trajetória prevista

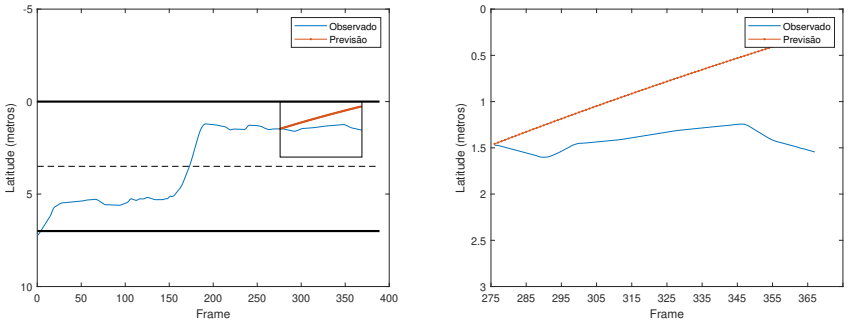
(b) Comparação de dados



(c) Trajetória prevista

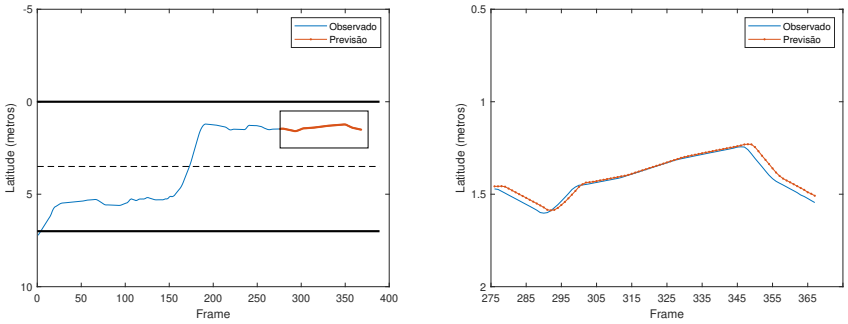
(d) Comparação de dados

Figura 4.8: Modelo Bi-LSTM-Drop-2 sem conhecimento (a) e (b) e com conhecimento (c) e (d)



(a) Trajetória prevista

(b) Comparação de dados



(c) Trajetória prevista

(d) Comparação de dados

Figura 4.9: Modelo Regression sem conhecimento (a) e (b) e com conhecimento (c) e (d)

Capítulo 5

Conclusão e trabalhos futuros

A realização desta dissertação permitiu não só aprofundar todo um conjunto de conhecimentos em relação à IA e áreas de aplicação, mas também dar a conhecer metodologias, técnicas e a complexidade que é a condução autónoma.

Uma vez que, no programa MATLAB, a existência de informação e projetos é substancialmente maior no setor da visão computacional que noutras áreas, o objetivo principal desta dissertação foi desenvolver uma base de algoritmo em IA que permitisse dar entrada na área de previsão de trajetórias. Com este objetivo, foi determinado como conjunto de tarefas, a realização de um pré-processamento dos dados fornecidos e o estudo, desenvolvimento, teste, avaliação e determinação do melhor modelo para a previsão futura de valores.

Obtidos os resultados das diferentes propostas e feita uma comparação entre elas, foi possível determinar a melhor arquitetura neste tipo de aplicação. Neste trabalho estão aplicados métodos mais simples e, por isso, existe margem para serem desenvolvidos diferentes algoritmos mais complexos e capazes de trabalhar com um maior conjunto de parâmetros que, neste projeto, foram principalmente dois, a localização e a velocidade. Apesar de, neste caso, não ter sido desenvolvido um método muito complexo, foi implementado um pré-processamento que serviria como apoio à criação de uma nova técnica de previsão. Este pré-processamento mais composto, que não foi explicado nesta dissertação, poderá vir a ser apoio para trabalhos futuros pois, devido à pouca disponibilidade de tempo, não foi possível utilizar.

Desta forma, como trabalho futuro, sugerem-se duas situações distintas. Um primeiro cenário é a utilização de mais parâmetros para influenciar os resultados finais

como as distâncias a que o veículo principal se encontra dos veículos em seu redor, a sua aceleração e utilização de determinados parâmetros dos veículos envolventes. Depois destas melhorias, um aumento do conjunto de dados com a utilização de mais faixas, uma vez que neste projeto foram feitas previsões com apenas as duas faixas mais à esquerda. Outro cenário é o desenvolvimento de um código capaz de trabalhar com a informação determinada com o código de pré-processamento de dados elaborado. Este pré-processamento tem uma base idêntica a vários projetos, onde a estrada é dividida em várias parcelas e cada parcela tem, ou não, a identificação do veículo. Essa identificação e localização seria uma ajuda para determinar o tipo de ambiente em que o veículo se encontra e tomar as melhores decisões.

Com os objetivos cumpridos e com um melhor conhecimento nesta área, que ainda está numa fase de evolução, considera-se um êxito o trabalho realizado durante esta etapa, onde fica disponibilizada uma base de aplicações que servem de entrada para quem quer ingressar na condução autónoma com o *software* MATLAB.

Referências

[1] "Mobility's future: An investment reality check | McKinsey." [Citado na página 5]

[2] "SAE Levels of Driving Automation™ Refined for Clarity and International Audience." [Citado na página 5]

[3] K. B. Isa and A. B. Jantan, "An Autonomous Vehicle Driving Control System," p. 12. [Citado na página 6]

[4] E. Alcala, L. Sellart, V. Puig, J. Quevedo, J. Saludes, D. Vazquez, and A. Lopez, "Comparison of two non-linear model-based control strategies for autonomous vehicles," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, (Athens, Greece), pp. 846–851, IEEE, June 2016. [Citado nas páginas vii e 7]

[5] G. Lewis, "Object Detection for Autonomous Vehicles," p. 6. [Citado na página 6]

[6] C. Premebida, G. Melotti, and A. Asvadi, "RGB-D Object Classification for Autonomous Driving Perception," in *RGB-D Image Analysis and Processing* (P. L. Rosin, Y.-K. Lai, L. Shao, and Y. Liu, eds.), pp. 377–395, Cham: Springer International Publishing, 2019. Series Title: Advances in Computer Vision and Pattern Recognition. [Citado na página 6]

[7] A. Ghosal and M. Conti, "Security issues and challenges in V2X: A Survey," *Computer Networks*, vol. 169, p. 107093, Mar. 2020. [Citado na página 7]

[8] "Self-driving sector contends its cars can prevent many more crashes than insurance study says," *Reuters*, June 2020. [Citado na página 7]

[9] A. Philipp and D. Göhring, "Interaction-aware Prediction of Urban Traffic Scenarios," p. 8 Seiten, 2022. Artwork Size: 8 Seiten Publisher: Freie Universität Berlin. [Citado na página 7]

[10] P. Bautista-Camino, A. I. Barranco-Gutiérrez, I. Cervantes, M. Rodríguez-Licea, J. Prado-Olivarez, and F. J. Pérez-Pinal, "Local Path Planning for Autonomous Vehicles Based on the Natural Behavior of the Biological Action-Perception Motion," *Energies*, vol. 15, p. 1769, Feb. 2022. [Citado na página 8]

[11] H. Fujiyoshi, T. Hirakawa, and T. Yamashita, "Deep learning-based image recognition for autonomous driving," *IATSS Research*, vol. 43, pp. 244–252, Dec. 2019. [Citado nas páginas vii, 8 e 9]

[12] Shaun Fernandes, Dhruv Duseja, and R. Muthalagu, "Application of Image Processing Techniques for Autonomous Cars," *Proceedings of Engineering and Technology Innovation*, Dec. 2020. [Citado na página 9]

[13] V. Viswanathan and R. Hussein, "Applications of Image Processing and Real-Time embedded Systems in Autonomous Cars: A Short Review," p. 15, 2017. [Citado na página 9]

[14] A. Jain, H. Reddy, and S. Dubey, "Automated Driving Vehicle Using Image Processing," *International Journal of Computer Sciences and Engineering*, vol. 2, p. 3, 2014. [Citado na página 9]

[15] M. Haenlein and A. Kaplan, "A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence," *California Management Review*, vol. 61, pp. 5–14, Aug. 2019. [Citado na página 9]

[16] S. Dua, S. S. Kumar, Y. Albagory, R. Ramalingam, A. Dumka, R. Singh, M. Rashid, A. Gehlot, S. S. Alshamrani, and A. S. AlGhamdi, "Developing a Speech Recognition System for Recognizing Tonal Speech Signals Using a Convolutional Neural Network," *Applied Sciences*, vol. 12, p. 6223, June 2022. [Citado na página 9]

[17] Y. Park, L. M. Dang, S. Lee, D. Han, and H. Moon, "Multiple Object Tracking in Deep Learning Approaches: A Survey," *Electronics*, vol. 10, p. 2406, Oct. 2021. [Citado na página 9]

[18] A. Rangesh and M. M. Trivedi, "No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles using Cameras & LiDARs," Feb. 2019. arXiv:1802.08755 [cs]. [Citado na página 9]

[19] G. Hemalatha, K. Srinivasa Rao, and D. Arun Kumar, "Weather Prediction using Advanced Machine Learning Techniques," *Journal of Physics: Conference Series*, vol. 2089, p. 012059, Nov. 2021. [Citado na página 9]

[20] K. S. R. Vanukuru, "Stock Market Prediction Using Machine Learning," 2018. Publisher: Unpublished. [Citado na página 9]

[21] B. Stojanović, J. Božić, K. Hofer-Schmitz, K. Nahrgang, A. Weber, A. Badii, M. Sundaram, E. Jordan, and J. Runevic, "Follow the Trail: Machine Learning for Fraud Detection in Fintech Applications," *Sensors*, vol. 21, p. 1594, Feb. 2021. [Citado na página 9]

[22] A. D. Babu, "Artificial Intelligence vs Machine Learning vs Deep Learning (AI vs ML vs DL)," Nov. 2019. [Citado na página 10]

[23] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, "Supervised Machine Learning," p. 112. [Citado na página 11]

[24] "Between Classification and Regression How to know which is which? | LaptrinhX." [Citado nas páginas vii e 11]

[25] S. Dridi, "Unsupervised Learning - A Systematic Literature Review," preprint, Open Science Framework, Apr. 2022. [Citado na página 12]

[26] W. Peng, X. Huang, X. Zhang, L. Ni, and S. Zhu, "A time-dependent reliability estimation method based on surrogate modeling and data clustering," *Advances in Mechanical Engineering*, vol. 11, p. 168781401983987, Apr. 2019. [Citado nas páginas vii e 13]

[27] F. Bre, J. M. Gimenez, and V. D. Fachinotti, "Prediction of wind pressure coefficients on building surfaces using artificial neural networks," *Energy and Buildings*, vol. 158, pp. 1429–1441, Jan. 2018. [Citado nas páginas vii e 14]

[28] A. Mathew, P. Amudha, and S. Sivakumari, "Deep Learning Techniques: An Overview," in *Advanced Machine Learning Technologies and Applications* (A. E. Hassanien, R. Bhatnagar, and A. Darwish, eds.), vol. 1141, pp. 599–608, Singapore: Springer Singapore, 2021. Series Title: Advances in Intelligent Systems and Computing. [Citado na página 14]

[29] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017. [Citado nas páginas vii e 14]

[30] "Recurrent Neural Network," May 2019. [Citado nas páginas vii e 15]

[31] R. d. S. Barbosa, "Princípios de Redes Neuronais," p. 67. [Citado nas páginas vii e 15]

[32] J. Li, H. Ma, W. Zhan, and M. Tomizuka, "Coordination and Trajectory Prediction for Vehicle Interactions via Bayesian Generative Modeling," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, (Paris, France), pp. 2496–2503, IEEE, June 2019. [Citado nas páginas vii, 17 e 18]

[33] C. Ding, W. Wang, X. Wang, and M. Baumann, "A Neural Network Model for Driver's Lane-Changing Trajectory Prediction in Urban Traffic Flow," *Mathematical Problems in Engineering*, vol. 2013, pp. 1–8, 2013. [Citado nas páginas vii, 19 e 20]

[34] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, NV, USA), pp. 961–971, IEEE, June 2016. [Citado na página 20]

[35] A. Zyner, S. Worrall, J. Ward, and E. Nebot, "Long short term memory for driver intent prediction," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, (Los Angeles, CA, USA), pp. 1484–1489, IEEE, June 2017. [Citado na página 20]

[36] "Capítulo 51 - Arquitetura de Redes Neurais Long Short Term Memory (LSTM)," Aug. 2019. [Citado nas páginas vii e 20]

[37] F. Altché and A. de La Fortelle, "An LSTM Network for Highway Trajectory Prediction," Jan. 2018. Number: arXiv:1801.07962 arXiv:1801.07962 [cs]. [Citado na página 21]

[38] N. Deo and M. M. Trivedi, "Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1179–1184, June 2018. arXiv:1805.05499 [cs]. [Citado nas páginas vii e 21]

[39] "Desktop Basics - MATLAB & Simulink." [Citado nas páginas vii e 22]

[40] U. D. O. T. F. H. Administration, "Next Generation Simulation (NGSIM) Vehicle Trajectories and Supporting Data," 2017. Type: dataset. [Citado na página 25]

[41] "Interstate 80 Freeway Dataset, FHWA-HRT-06-137." [Citado nas páginas vii e 26]

Anexo A

Código Projeto em MATLAB

```
%% Modelação e previsão de trajetórias com recurso a inteligência artificial

    clc
    close all
    clear all

%% Carregar dados

    load('i80.mat')

% Configurações
    numEntradas = 1;
    numRespostas = 1;
    numCamadas1 = 300;
    numCamadas2 = 100;
    dropoutProb = 0.2;

%% Escolha do veículo 757 e parâmetros

    veic_teste=trajectories(trajectories(:,1)==757,:);
    parametro=veic_teste(:,5).';
    %localizacao=veic_teste(:,12).';
```

```
    comprimento=length(parametro)+20;

%% Criação da rede

    num_steps_train=floor(.75*numel(parametro));
    train=parametro(1:num_steps_train+1);
    test=parametro(num_steps_train+1:end);

    med=mean(train);
    sigma=std(train);
    std_train_dados=(train-med)/sigma;

    XTrain=std_train_dados(1:end-1);
    YTrain=std_train_dados(2:end);

%% Escolha do modelo a treinar

%%          Modelos com LSTM
    layersSetting = "withoutDropout";
    %layersSetting = "2withoutDropout";

    %layersSetting = "withDropout";
    %layersSetting = "2withDropout";

%%          Modelos com BiLSTM
    %layersSetting = "biwithoutDropout";
    %layersSetting = "2biwithoutDropout";

    %layersSetting = "biwithDropout";
    %layersSetting = "2biwithDropout";

%%          Modelo com Regression
    %layersSetting = "simpleRegression";

%% Definição dos modelos

    switch layersSetting
        case "withoutDropout"                           %LSTM-1
            layers = [ ...
                sequenceInputLayer(numEntradas)
```

```
        lstmLayer(numCamadas1)
        fullyConnectedLayer(numRespostas)
        regressionLayer];

case "2withoutDropout"                          %LSTM-2
    layers = [ ...
        sequenceInputLayer(numEntradas)
        lstmLayer(numCamadas1)
        lstmLayer(numCamadas2)
        fullyConnectedLayer(numRespostas)
        regressionLayer];

case "withDropout"                              %LSTM-Drop-1
    layers = [ ...
        sequenceInputLayer(numEntradas)
        dropoutLayer(dropoutProb)
        lstmLayer(numCamadas1)
        fullyConnectedLayer(numRespostas)
        regressionLayer];

case "2withDropout"                             %LSTM-Drop-2
    layers = [ ...
        sequenceInputLayer(numEntradas)
        dropoutLayer(dropoutProb)
        lstmLayer(numCamadas1)
        dropoutLayer(dropoutProb)
        lstmLayer(numCamadas2)
        fullyConnectedLayer(numRespostas)
        regressionLayer];

case "biwithoutDropout"                         %Bi-LSTM-1
    layers = [ ...
        sequenceInputLayer(numEntradas)
        bilstmLayer(numCamadas1)
        fullyConnectedLayer(numRespostas)
        regressionLayer];

case "2biwithoutDropout"                        %Bi-LSTM-2
    layers = [ ...
        sequenceInputLayer(numEntradas)
```

```
                bilstmLayer(numCamadas1)
                bilstmLayer(numCamadas2)
                fullyConnectedLayer(numRespostas)
                regressionLayer];

        case "biwithDropout"                                %Bi-LSTM-Drop-1
            layers = [ ...
                sequenceInputLayer(numEntradas)
                dropoutLayer(dropoutProb)
                bilstmLayer(numCamadas2)
                fullyConnectedLayer(numRespostas)
                regressionLayer];

        case "2biwithDropout"                               %Bi-LSTM-Drop-2
            layers = [ ...
                sequenceInputLayer(numEntradas)
                dropoutLayer(dropoutProb)
                bilstmLayer(numCamadas1)
                dropoutLayer(dropoutProb)
                bilstmLayer(numCamadas2)
                fullyConnectedLayer(numRespostas)
                regressionLayer];

        case "simpleRegression"                             %Regression
            layers = [ ...
                sequenceInputLayer(numEntradas)
                fullyConnectedLayer(numCamadas1)
                fullyConnectedLayer(numRespostas)
                regressionLayer];
    end

%% Opções de Treino

    options = trainingOptions('adam', ...
                'MaxEpochs',200, ...
                'Plots','training-progress');

%% Treino e teste da rede - Com conhecimento

    net=trainNetwork(XTrain,YTrain,layers,options);
```

```
    std_test_data=(test-med)/sigma;
    XTest=std_test_data(1:end-1);
    num_steps_test=numel(XTest);

    net=predictAndUpdateState(net,XTrain);
    [net,YPred]=predictAndUpdateState(net,YTrain(end));


    for i=2:num_steps_test
    %Seleção dos dados XTest para ter conhecimento
        [net,YPred(:,i)]=predictAndUpdateState(net,XTest(:,i-1));
    end
    YPred=YPred*sigma+med;
    YTest=test(2:end);

% Cálculo do erro
    rmseC=sqrt(mean((YPred-YTest).^2));

%% Figura 1 - Gráfico 75% da trajetória + 25% previsão com conhecimento
    figure
    plot(parametro)
    hold on
    idx=num_steps_train:(num_steps_train+num_steps_test);
    plot(idx,[parametro(num_steps_train),YPred],'.-')
    ylim([-5 10])
    line([0 comprimento],[0 0],'Color','k','LineStyle','-','LineWidth',2)
    line([0 comprimento],[3.5 3.5],'Color','k','LineStyle','--', 'LineWidth',0.5)
    line([0 comprimento],[7 7],'Color','k','LineStyle','-','LineWidth',2)
    set(gca, 'ydir', 'reverse');
    xlabel('Frame');
    ylabel('Latitude (metros)');

    legend(["Observado" "Previsão"])
    set(gcf,'color','w');
    rectangle('Position',[276 0.5 93 2])

%% Figura 2 - Gráfico previsão com conhecimento e erro
    figure
    subplot(2,1,1)
```

```
    plot(YTest)
    hold on
    plot(YPred,'.-')
    xticklabels({'275','285','295','305','315','325','335','345','355','365'})
    hold off
    legend(["Observado" "Previsão"])
    xlabel('Frame');
    ylabel('Latitude (metros)');
    set(gca, 'ydir', 'reverse');
    ylim([0.5 2.5])

    subplot(2,1,2)
    stem(YPred-YTest)
    xlabel("Frame")
    ylabel("Error")
    title("RMSE="+rmseC)
    set(gcf,'color','w');


%% Reset rede - Previsão Sem conhecimento
%Reset da rede
    net=resetState(net);
    net=predictAndUpdateState(net,XTrain);
    [net,YPred]=predictAndUpdateState(net,YTrain(end));

    for i=2:num_steps_test
    %Seleção dos dados YPred para não ter conhecimento
        [net,YPred(:,i)]=predictAndUpdateState(net,YPred(:,i-1));
    end
    YPred=sigma*YPred+med;

% Cálculo do erro sem conhecimento
    rmseS=sqrt(mean((YPred-YTest).^2));

%% Figura 3 - Gráfico 75% da trajetória + 25% previsão sem conhecimento
    figure
    plot(parametro)
    hold on
    idx=num_steps_train:(num_steps_train+num_steps_test);
    plot(idx,[parametro(num_steps_train),YPred],'.-')
```

```
    ylim([-5 10])
    line([0 comprimento],[0 0],'Color','k','LineStyle','-','LineWidth',2)
    line([0 comprimento],[3.5 3.5],'Color','k','LineStyle','--', 'LineWidth',0.5)
    line([0 comprimento],[7 7],'Color','k','LineStyle','-','LineWidth',2)
    set(gca, 'ydir', 'reverse');
    xlabel('Frame');
    ylabel('Latitude (metros)');
    legend(["Observado" "Previsão"])
    set(gcf,'color','w');
    rectangle('Position',[276 0.5 93 2])

%% Figura 4 - Gráfico previsão sem conhecimento e erro
    figure
    subplot(2,1,1)
    plot(YTest)
    hold on
    plot(YPred,'.-')
    hold off
    legend(["Observado" "Previsão"])
    xlabel('Frame');
    ylabel('Latitude (metros)');
    set(gca, 'ydir', 'reverse');
    ylim([0.5 2.5])
    xticklabels({'275','285','295','305','315','325','335','345','355','365'})

    subplot(2,1,2)
    stem(YPred-YTest)
    xlabel("Frame")
    ylabel("Error")
    title("RMSE="+rmseS)
    set(gcf,'color','w');
```