Universidade do Minho
Mestrado Integrado em Engenharia Informática
Classificadores e Sistemas Conexionistas

Deep Learning - Previsão do Fluxo de Trânsito

Joel Morais, A70841
Tiago Fraga, A74092

30 de Maio de 2019

Conteúdo

Capítulo 1

Introdução

Com a realização deste trabalho prático pretende-se conceber e implementar modelos de *Deep Learning* baseado em Redes Neuronais *LSTM* com o objetivo do mesmo ser capaz de prever o fluxo de trânsito e incidentes numa cidade que seja alvo de estudo.

A segurança rodoviária é um problema que tem vindo a aumentar ao longo do tempo com o número de acidentes e de mortes na estrada provocadas por possíveis congestionamentos e distrações dos condutores nas estradas, ou por outro lado, pelas condições atmosféricas que se verificam no momento.

Uma possível solução, que será o foco principal o projeto, é na previsão de fluxo de tráfego rodoviário.

Com esta previsão podemos auxiliar a Câmara Municipal do Porto, bem como diversas entidades (autoridades rodoviárias, pedestres, etc), de modo a ajudar na prevenção dos acidentes rodoviários.

Para este estudo ser efetuado foi-nos fornecido um conjunto de dados em bruto: Dados Climatéricos, Incidentes e Fluxo de Tráfego.

Este projeto foi dividido em quatro fases. A primeira fase consiste na definição da metodologia a adotar, a segunda consiste no tratamento e seleção dos dados que serão fornecidos à terceira fase que será a criação do modelo de *Deep Learning*. Por fim, será feita uma análise dos resultados obtidos.

Capítulo 2

Caso de Estudo

2.1 Fontes de Informação

2.1.1 Descrição dos DataSets

Para a cidade do Porto, foram-nos fornecidos três *DataSets* com informações distintas.

O primeiro *DataSet*, que daqui para a frente denominamos como **Traffic Flow**, continha informação sobre o fluxo de trânsito nas oito principais estradas da cidade: Avenida Gustavo Eiffel, Ponte do Freixo, Ponte da Arrábida, Rotunda da Boavista, Avenida dos Aliados, Rua Conde de Vizela, Rua Nova de Alfandega e a Rua da Constituição. A informação presente no *DataSet* contem vários atributos, que iremos detalhar mais á frente.

O segundo *DataSet*, denominado por **Weather** contem informações detalhadas sobre o estado do tempo em toda a cidade do Porto.

O terceiro *DataSet*, denominado por **Incidents**, contem informação sobre vários incidentes ocorridos em inúmeras estradas da cidade.

Todos *DataSets* continham informações sobre a data e a hora que ocorriam os eventos, portanto denotamos logo que este iria ser o atributo mais importante de forma a unir toda a informação.

2.1.2 Atributos

Cada um dos *DataSets* vinha originalmente com vários atributos, no entanto numa pré-análise definimos que nem todos eram importantes para o estudo em questão. Desta forma, e antes de qualquer tratamento dos dados, foi feita uma filtragem dos dados com que íamos trabalhar e os que iam ser descartados, para cada um dos ficheiros.

Fluxo de Trânsito

O ficheiro original com a informação sobre o fluxo de transito contém onze atributos:

- city_name - Nome da cidade;
- road_num - Numero da rua;
- road_name - Nome da rua;
- functional_road_class_desc - Descrição do tipo de rua;
- current_speed - Velocidade que se circula na estrada sem trânsito;
- free_flow_speed - Velocidade a que se está a circular na estrada num determinado momento;
- speed_diff - Diferença entre os dois atributos anteriores;
- current_travel_time - Tempo que demora a percorrer a estrada nem trânsito;
- free_flow_travel_time - Tempo que se demorou a percorrer a estrada num determinado momento.
- time_diff - Diferença entre os dois atributos anteriores;
- creation_date - Data que foram recolhidos os dados;

Dos atributos descritos, decidimos remover os seguintes: **city_name -** Visto já sabermos qual a cidade que estamos a trabalhar nao era necessário; **functional_road_class_desc -** Não vamos agrupar os dados por tipos de ruas;

Tempo

O ficheiro original com a informação do tempo contém doze atributos:

- city_name - Nome da cidade;
- weather_description - Descrição do estado do tempo;
- temperature - Temperatura;
- atmospheric_pressure - Pressão atmosférica;
- humidity - Humidade;
- wind_speed - Velocidade do vento;

- cloudiness - Nebulosidade;
- rain - Chuva;
- current_luminosity - Luminosidade;
- sunrise - Hora do nascer do sol;
- sunset - Hora do por do sol;
- creation_date - Data que foram recolhidos os dados;

Dos atributos descritos, decidimos remover os seguintes: **city_name -** Visto já sabermos qual a cidade que estamos a trabalhar nao era necessário; **temperature -** Não consideramos a temperatura relevante para analisar o fluxo de trânsito ou os incidentes; **atmospheric_pressure -** Não consideramos a pressão atmosférica relevante para analisar o fluxo de trânsito ou os incidentes; **rain -** Não consideramos o atributo *chuva* relevante para analisar o fluxo de trânsito ou os incidentes, visto que verificamos que este atributo tinha sempre o valor zero;**sunrise e sunset -** Não consideramos estes atributos relevantes para analisar o fluxo de trânsito ou os incidentes, visto que já temos o atributo de luminosidade;

Incidentes

O ficheiro original com a informação dos incidentes contém onze atributos:

- city_name - Nome da cidade;
- description - Descrição do incidente;
- cause_of_incident - Causa do incidente;
- from_road - Rua do inicio do incidente;
- to_road - Rua do fim do incidente;
- affected_roads - Ruas afetadas pelo incidente;
- incident_category_desc - Descrição da categoria do incidente;
- magnitude_of_delay_desc - Descrição do atraso do incidente;
- length_in_meters - Tamanho da fila provocada pelo incidente;

- delay_in_seconds - Atraso em segundos provocado pelo incidente;
- incident_date - Data que foram recolhidos os dados;

Dos atributos descritos, decidimos remover os seguintes: **city_name -** Visto já sabermos qual a cidade que estamos a trabalhar nao era necessário; **cause_of_incident e affected_roads -** Pois tinha muitos valores em falta; **incident_category_desc e magnitude_of_delay_desc -** A gama de valores disponíveis era muito curta para fornecer ao modelo(3 valores distintos).

2.2 Metodologia

Após a filtragem dos dados é importante definir o rumo de projecto e que dados irão ser calculados.

Face aos *DataSets* disponíveis, foi decidido que a primeira abordagem seria a junção dos dados climatéricos com o ficheiro do fluxo de transito e com o ficheiro de incidentes, desta forma obtínhamos dois ficheiros com onze e dez atributos respetivamente.

O objetivo do grupo com foi definido anteriormente era desenvolver dois modelos de *deep learning* capazes de absorver cada um dos ficheiros e calcular o pretendido.

Feita a junção de informação ficamos apenas com dois ficheiros principais, que daqui para a frente denominamos como **Fluxo de Trânsito** e **Incidentes**.

2.2.1 Fluxo de Trânsito

O modelo de *Deep Learning* que pretendemos para analisar e prever os dados deste ficheiro tem como objetivo prever o atributo **time_diff** ou **speed_diff**.

Para isso irão ser criados **blocos de informação correspondentes a 24 horas com o intuito de prever a hora a seguir** a esse bloco. Face aos recursos computacionais disponíveis, acreditamos ser a abordagem correta de forma a nao entrar em sobrecarga.

Como foi referido no sub-tópico acima este ficheiro contém informação sobre as oito principais ruas do porto, como tal o nosso objetivo prende-se a aplicar o modelo de *Deep Learning* desenvolvido a cada uma das ruas e verificar o resultado final para a previsão do *time_diff* e do *speed_diff*.

2.2.2 Incidentes

O modelo de *Deep Learning* que pretendemos para analisar e prever os dados deste ficheiro tem como objetivo prever o atributo **delay_in_seconds**.

Por limitações de tempo e de recursos, adotamos a metodologia criada para o primeiro ficheiro, ou seja, criamos um bloco de informação de 24 horas e procuramos prever os incidentes para a próxima hora a seguir a esse bloco.

Neste ficheiro procuramos nao dar relevância às ruas, mas sim à cidade do Porto como um só. Desta forma, no tratamento dos dados irão ser retirados os nomes das ruas e a informação contida no ficheiro será relativa à cidade do Porto.

2.3 Tratamento de Dados

Concluída a etapa de modelação da metodologia do projeto, e os ficheiros a trabalhar possuírem os atributos corretos a serem trabalhados, começamos a fase de tratamento dos dados.

Todo o tratamento de dados do projeto foi efetuado com o auxilio da plataforma **Knime**.

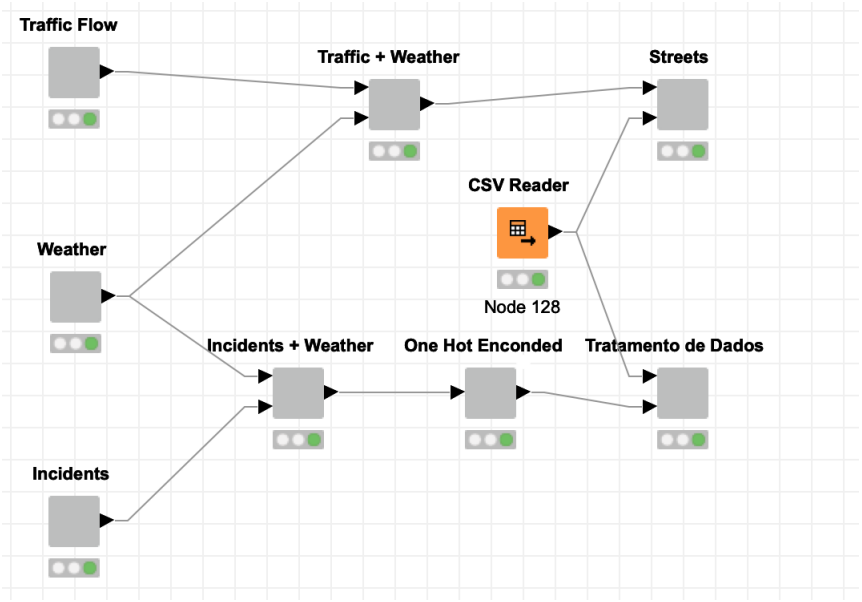

Figura 2.1: Workflow geral.

Como é possivel ver na imagem acima, começamos por fazer um tratamento individual a cada um dos ficheiros para posteriormente serem agregados como já foi descrito.

O tratamento inicial que foi feito a cada ficheiro consistiu no rearranjo da hora, e na eliminação dos atributos que nao pretendíamos usar para o estudo.

De forma a obtermos um intervalo temporal coerente, decidimos arredondar os valores das datas para ficarem num intervalo de hora a hora, ou seja, as horas cujos minutos eram inferiores a 30 arredondavam para a hora abaixo enquanto que o oposto arredonda para a hora acima.

Posto isto, é feita a junção de informação entre as três fontes de informação, resultando nos dois ficheiros que irão ser trabalhados.

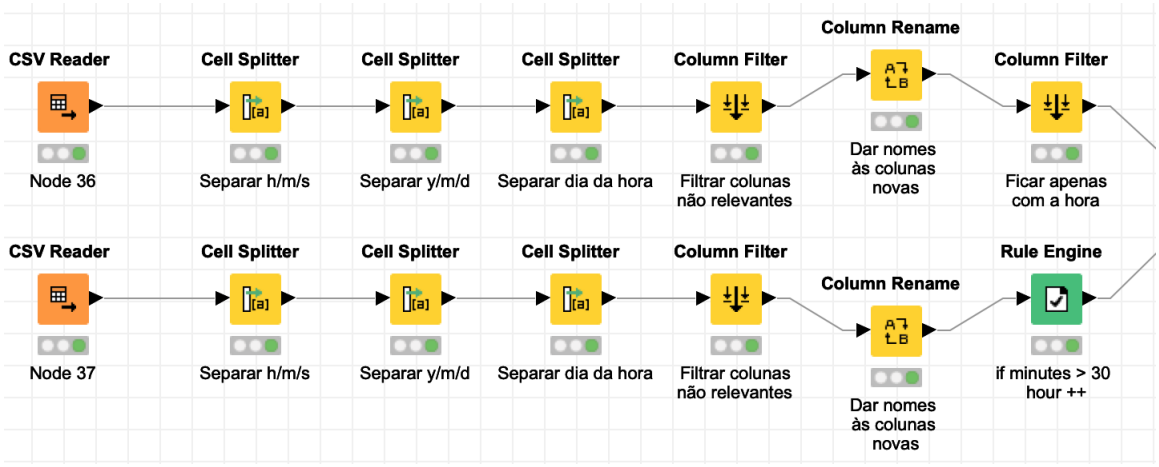

Figura 2.2: Eliminação de atributos e arredondamento da hora.

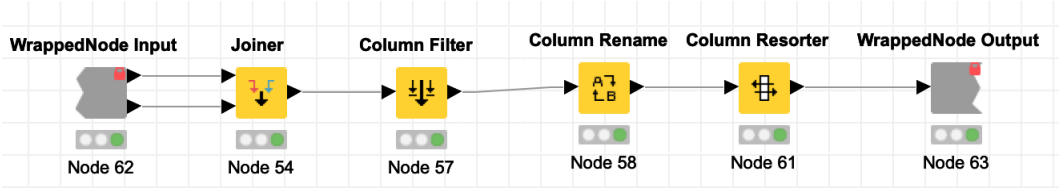

Figura 2.3: Junção dos datasets.

2.3.1 Fluxo de Trânsito

Concluída a parte de tratamento dos dados gerais, é o momento de fazer um trabalho criterioso com o ficheiro que contém a informação do fluxo de transito.

Como foi referido anteriormente, pretendemos estudar o fluxo na perspetiva individual das principais ruas da cidade do porto, como tal, a primeira tarefa foi dividir os dados por ruas.

Para cada uma destas ruas o tratamento dos dados foi bastante semelhante.

A primeira etapa consistiu em fazer o *One Hot Encode* de forma a permutar os dados nominais para valores numéricos.

Posteriormente foi feita a verificação das datas com um ficheiro que possuía todas as horas desde a data do primeiro registo até ao ultimo, deste modo conseguimos perceber que informação estava a faltar neste intervalo.

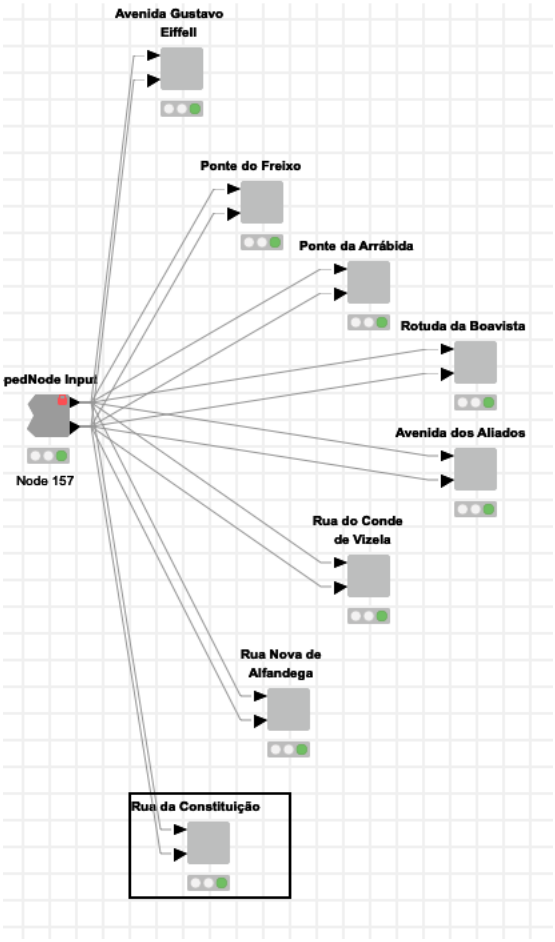

Figura 2.4: Divisão dos dados.

Perante esta análise pudemos concluir que a informação começava a 24 de Julho de 2018 e terminava a 1 de Março de 2019. No entanto, existia um enorme buraco de valores entre Dezembro de 2018 e Janeiro de 2019, portanto decidimos remover estes meses do estudo. Os meses de Julho de 2018 e Março de 2019 também foram removidos pois não possuíamos informação completa dos meses. Desta forma, o nosso estudo baseou-se de 1 de Agosto de 2018 ate 30 de Novembro de 2018 e no mês de Fevereiro de 2019.

Na terceira etapa deste tratamento, foi feita a verificação dos valores em falta nas entradas de informação que chegaram até esta fase. A reposição dos valores em falta foi obtida indo buscar os valores da hora anterior.

Por fim, foi feita a normalização dos dados para uma gama de valores entre -1 e 1. Como estamos a lidar com valores números para melhor a precisão do modelo de *Deep Learning* esta etapa foi

crucial na obtenção de bons resultados, pois porque numa fase inicial foi uma etapa que passamos a frente e estava a prejudicar a precisão do modelo.

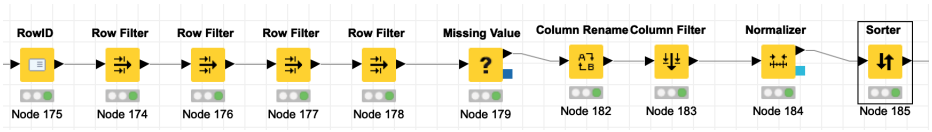

Figura 2.5: Normalização dos dados.

2.3.2 Incidentes

O tratamento dos dados para este ficheiro foi em tudo semelhante ao que foi descrito em cima.

Em primeiro lugar seguimos a metodologia descrita em cima, e começamos por remover o nome das ruas de forma a tratar os dados presentes como sendo pertencentes à cidade do porto no geral.

Em segundo lugar foi feito o *One Hot Encode*, e neste caso, a maior parte dos atributos eram nominais portanto foi uma etapa importante

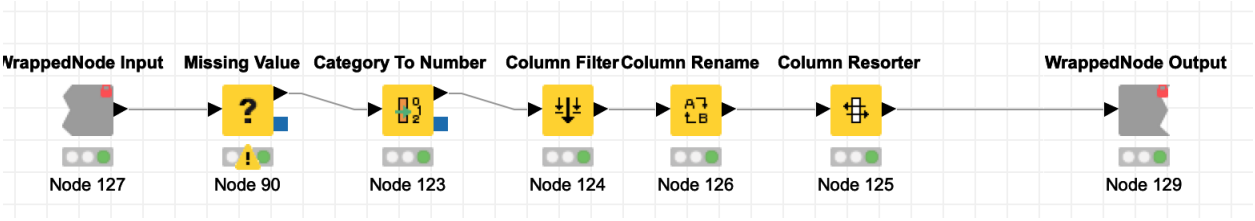

Figura 2.6: One Hot Encode.

Por fim, foram seguidas as mesmas etapas que foram descritas em cima após o *One Hot Encode*.

De salientar que o intervalo temporal estudado também foi desde 1 de Agosto até 30 de Novembro de 2018, bem como o mês de Fevereiro de 2019.

Capítulo 3

Modelo de Deep Learning

Terminada a fase de tratamento dos dados e estando os mesmos em condições de serem fornecidos ao modelo de *Deep Learning*, foi possivel começar a criação do mesmo.

3.1 Descrição do Modelo

Em primeiro lugar, foi necessário ter o seguinte software instalado nas máquinas dos elementos do grupo.

1. Python 3.7
2. NumPy
3. Pandas
4. TensorFlow
5. Sklearn

Foi desenvolvida uma rede LSTM de modo a conseguirmos fazer a nossa previsão necessária.

Em suma, **o nosso objetivo será fazer uma previsão de hora em hora, dadas as 24 horas anteriores**.

Começamos pela recolha dos dado, importando do ficheiro *csv*, todos os dados. O caso apresentado será relativamente à Rotunda da Boavista, havendo a possibilidade de adaptarmos o modelo para outras ruas:

```
data = pd.read_csv('Porto_Data/WeatherTraffic/Rotunda_da_Boavista.csv', header=0, sep=',' )
data_train = data[0:2600]
data_test = data[2600:3600]
```

Antes de começarmos a definir a rede é necessário mais um passo intermédio, ou seja, passar de um problema de sequência temporal para um problema supervisionado, de modo a conseguirmos fazer uma previsão, tendo em conta alguns componentes.

```
def to_supervised(df, tamanho, intervalo_previsao):
data = df.values
X, y = list(),list()

for pos in range(len(data)):
    tamanho_fim = pos + tamanho
    inicio_previsao = tamanho_fim
    fim_previsao = tamanho_fim + intervalo_previsao
    if fim_previsao < len(data):
        X.append(data[pos:tamanho_fim,:])
        y.append(data[inicio_previsao:fim_previsao,5])

X = np.reshape(np.array(X),(len(X),tamanho,11))
y = np.reshape(np.array(y),(len(y),1))

return X, y
```

O nosso problema, passando a supervisionado, vai estar dividido em blocos de 24 horas, (ou seja, do 0 ao 23) de modo a conseguirmos prever a hora seguinte.

- Batch_Size = 32
- TimeSteps = 24
- Features = 11
- Multi_Steps = 1

A escolha das *TimeSteps* pelo valor de 24 deve-se ao facto de querermos avaliar um dia inteiro, de modo a conseguirmos prever a próxima hora, daí o *Multi_Steps* ser a 1, podendo ser assim possível a previsão da próxima hora. O nosso *DataSet* é constituído por 11 *features* (atributos) que irão ser tomadas em consideração para fazermos a previsão.

Com uma *Batch_Size* de 32, estamos a dizer que após 32 amostras, os pesos da nossa rede *LSTM* vão ser atualizados, ou seja, a memória vai ser reiniciada. No contexto da rede que está a ser desenvolvida, após um mês a aprendizagem irá ser reiniciada.

3.2 Rede LSTM

Vamos começar a desenvolver a nossa rede LSTM para o problema em causa. Irá ser *stateful* e reconfigurar explicitamente o estado interno no final de cada época para cada amostra gerada.

As dimensões dos dados de input foram definidos como (*length,nr_feature*), atributos estes que já foram previamente explicados.

A primeira camada da rede vai utilizar 25 neurónios, que ao testando, fomos vendo que seria um número razoável para aprender este problema.

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(15,input_shape=(length,nr_features)))
```

De seguida definimos mais uma camada *Dense*, que recebe como argumento a variável *Multi_Steps*, que será 1.

Ao correrrmos cada época, queremos apresentar a *mean squared error* para conseguirmos perceber quanto de erro estamos a obter com a nossa previsão, tal como a *accuracy* e o erro absoluto médio. Isto tudo é feito através de:

```
model.add(tf.keras.layers.Dense(multi_steps))
model.compile(
    loss=tf.keras.losses.mse,
    optimizer=tf.train.AdamOptimizer(0.001),
    metrics=['accuracy','mae'])
```

Antes de passarmos à previsão da hora, é necessário fazer um *fit* do modelo, tendo controlo assim sobre o número de épocas de treino em que o modelo vai utilizar, sendo possível também ir alterando a batch_size, cada vez que se procede a esta ação de forma a melhoras os valores da precisão.

```
model.fit(X,y,shuffle=False,batch_size=batch_siz,epochs=1000,verbose=2)
```

Por fim, e já com o tratamento dos dados, em conjunto com o tratamento de toda a rede, estão todas as condições reunidas para procedermos à previsão do atributo pretendido para a próxima hora.

```
yhat=model.predict(X_test)
```

```
print('Expected:%s'% y_test[0])
print('Predicted:%s'% yhat[0])
```

Capítulo 4

Resultados Obtidos

Concluído o processo de criação do modelo de *Deep Learning*, foi possivel obter os resultados e proceder à analise dos mesmos.

Durante o processo de tuning do modelo foram feitas várias alterações nos parâmetros tais como, *batch size*, numero de neurónios, numero de camadas, quantidade de épocas de treino, tipo de otimizador usado, etc.

Pudemos concluir que quanto mais pequeno fosse o *batch size* , melhor eram os resultados da previsão. Para um valor de 32 ou 64, a *accuracy* do modelo era bastante boa, a partir desses valor começava a piorar.

Em relação ao numero de épocas de treino, pudemos afirmar que quanto maior for melhor é para a aprendizagem do modelo, no entanto limitamos este valor a 1000 para nao haver sobrecarga da máquina e para que o treino nao se torne muito custoso em termos de tempo.

Após estas etapas, fizemos o calculo do *MAE* (*Mean Absolute Error*) para a previsão do time_diff e do speed_diff para as oito principais ruas do porto. Relembrando que estes ficheiros contêm informação das condições atmosféricas e do fluxo de trânsito.

Para este cálculo foram usadas 1000 épocas de treino, o *batch size* com o valor de 32, os *timesteps* com o valor de 24 e as *features* com o valor de 11.

	time_diff	speed_diff
Avenida Gustavo Eiffel	0.0202	0.0213
Ponte do Freixo	0.0289	0.0293
Ponte da Arrábida	0.0191	0.0101
Rotunda da Boavista	0.0221	0.0430
Avenida dos Aliados	0.0270	0.0279
Rua Conde Vizela	0.0431	0.0435
Rua Nova Alfândega	0.0280	0.0263
Rua da Constituição	0.0244	0.0256

Tabela 4.1: Previsão dos valores para as 8 principais ruas do Porto.

Como é possivel verificar pela tabela, os valores do *time_diff* e do *speed_diff* situam-se sempre a rondar o valor **0.02** o que nos permite afirmar que o modelo construído é bastante robusto para várias fontes de informação. Apenas para os valores da Rua Conde de Vizela os valores foram dobrados, sendo que consideramos que se deve ao facto de que os dados que foram fornecidos para treino da rede possuam valores muito diferentes dos que foram passados como dados de teste e desta forma a rede nao conseguiu obter um valor tao bom de *accuracy*.

Para o segundo ficheiro, dos dados dos incidentes de trânsito com as condições atmosféricas, procedemos à previsão com os mesmos valores acima descritos e obtivemos um *mae* de **0.0821** para a previsão do *delay in seconds* da hora seguinte às vinte e quatro passadas como teste.

Capítulo 5

Conclusões e Trabalho Futuro

Após a análise dos resultados podemos afimar que o modelo de *Deep Learning* com redes neuronais *LSTM*, é robusto e capaz de prever com enorme exatidão o que lhe é proposto.

Se relembrarmos os tópicos apresentados no subcapítulo da metodologia, verificamos que para a primeira fonte de informação relativa aos dados sobre o fluxo de transito e as condições climatéricas, conseguimos cumprir todos os pressupostos.

Fizemos a analise das principais ruas do porto, e foi possivel obter resultados bastante satisfatórios usando o mesmo modelo para as diversas estradas.

Desta forma, consideramos que esta etapa do trabalho foi concluída com bastante sucesso.

No entanto, os pressupostos que inicialmente idealizamos para estudar a segunda fonte de informação, ou seja, os dados dos incidentes de transito com os dados climatéricos nao foram atingidos na integra.

Inicialmente, idealizamos fazer um estudo rua a rua como foi feito para a primeira fonte de informação, no entanto, por escassez de tempo e recursos tivemos de adaptar esta abordagem, adotando a que explicitamos no presente relatório.

Contudo, consideramos que esta abordagem generalizada, sobre o estudo da cidade do porto como um só e nao por ruas obteve bastantes bons resultados, visto que com dados sobre as vinte e quatro horas de registo de incidentes conseguimos prever o tempo de espera numa fila para a próxima hora, com um erro de 0.08.

Com esta gama de valores nunca podemos afirmar que são resultados insatisfatórios, no entanto, e como trabalho futuro idealizamos que poderá ser concretizada a nossa ideia inicial, bem como a junção de todas as fontes de informação numa só fonte e desta maneira construir um modelo geral e robusto que seja capaz de calcular todas as necessidades que se sejam visíveis para a redução da sinistralidade nas estradas da cidade do Porto.

Apêndice A

Anexo

Apresentamos o código das funções implementadas.

```
1  import tensorflow as tf
2  import numpy as np
3  from random import randint
4  from numpy import array
5  from numpy import argmax
6  import pandas as pd
7  from sklearn.model_selection import train_test_split
8
9
10 # Converter para supervised learning
11 def to_supervised(df, tamanho, intervalo_previsao):
12     data = df.values
13     X, y = list(),list()
14
15     for pos in range(len(data)):
16
17         tamanho_fim = pos + tamanho
18
19         inicio_previsao = tamanho_fim
20
21         fim_previsao = tamanho_fim + intervalo_previsao
22
23
24         if fim_previsao < len(data):
25             X.append(data[pos:tamanho_fim,:])
26             y.append(data[inicio_previsao:fim_previsao,5])
27
28
```

```
29
30      X = np.reshape(np.array(X),(len(X),tamanho,11))
31      y = np.reshape(np.array(y),(len(y),1))
32
33      return X, y
34
35
36  data = pd.read_csv('Porto_Data/WeatherTraffic/Rotunda_da_Boavista.csv',
        header=0, sep=',' )
37  print(data.shape)
38  data_train = data[0:2600]
39  data_test = data[2600:3600]
40
41
42  length = 24
43  nr_features = 11
44  multi_steps = 1
45
46
47  X,y = to_supervised(data_train, length, multi_steps)
48
49
50  model = tf.keras.Sequential()
51  model.add(tf.keras.layers.LSTM(15,input_shape=(length,nr_features)))
52  model.add(tf.keras.layers.Dense(multi_steps))
53  model.compile(
54          loss=tf.keras.losses.mse,
55          optimizer=tf.train.AdamOptimizer(0.001),
56          metrics=['accuracy','mae'])
57
58  print(model.summary())
59
60  model.fit(X,y,shuffle=False,epochs=10,verbose=2)
61
62
63  X_test,y_test = to_supervised(data_test,length,multi_steps)
64
65
66  yhat=model.predict(X_test)
67
68  print('Expected:%s'% y_test[0])
69  print('Predicted:%s'% yhat[0])
```