Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico Di Milano

Network Data Analysis Lab
Project 9: Traffic Forecasting

Team: Group 8

Presenters: Alexander Stefitz, Enrico Gregorini, Lorenzo Prada

Academic Year: 2022/2023

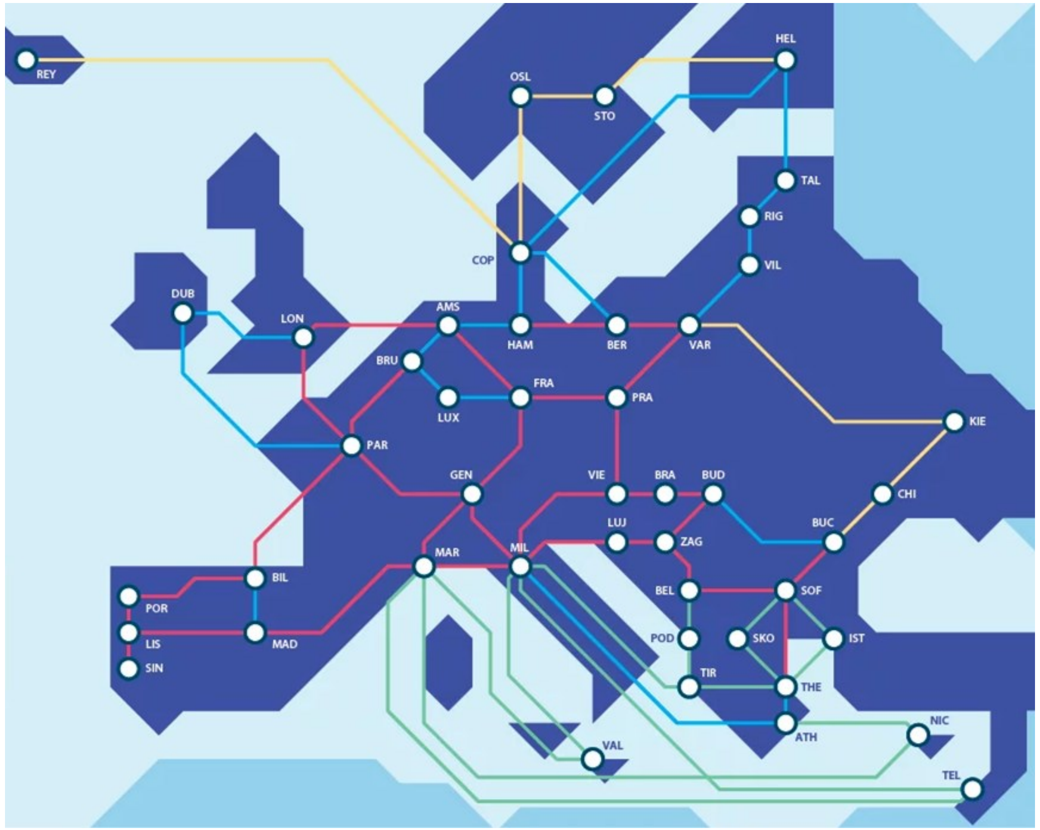
Introduction

- Network Traffic changes quickly during the day
- Processing and transporting resources can be scaled
- A good traffic prediction may save a lot of resources

- How do we perform a good traffic prediction and how can we use it to save resources?
  - Machine learning algorithms
  - Automatically turn on/off interfaces for traffic allocation based on the predictions

Dataset – Info

- GÉANT is the research network that carries traffic between universities and research institutes in Europe
- GÉANT is composed of 23 routers connected with 38 physical links
- GÉANT uses SONET technology to multiplex traffic with different bitrates into one optical signal
- Channel with the smallest bitrate that can be created in SONET is **50 Mb/s**



- <u>Each file in the dataset describes total traffic in kb/s between pairs of routers</u>
- <u>Dataset includes 2941 files: traffic at 15 minutes intervals for 1 month</u>

Dataset – Files

- In total there are 2941 .xml files, each containing the amount of traffic measured in the previous 15 minutes
- The measures has been taken from 01/01/2005 to 31/01/2005 included
- We do not have any topological information about the network
- We cannot assume any other information except the ones provided in the .xml files


```xml
<IntraTM ASID="20965">
<src id="12">
  <dst id="12">305258.2222</dst>
  <dst id="13">28801.8756</dst>
  <dst id="19">4077.1556</dst>
  <dst id="23">166.9067</dst>
  <dst id="8">1812.3022</dst>
</src>
<src id="13">
  <dst id="12">37182.8622</dst>
  <dst id="19">20624.0267</dst>
  <dst id="23">818.2044</dst>
  <dst id="8">10468.6044</dst>
</src>
</IntraTM>
```

Dataset – Missing Files

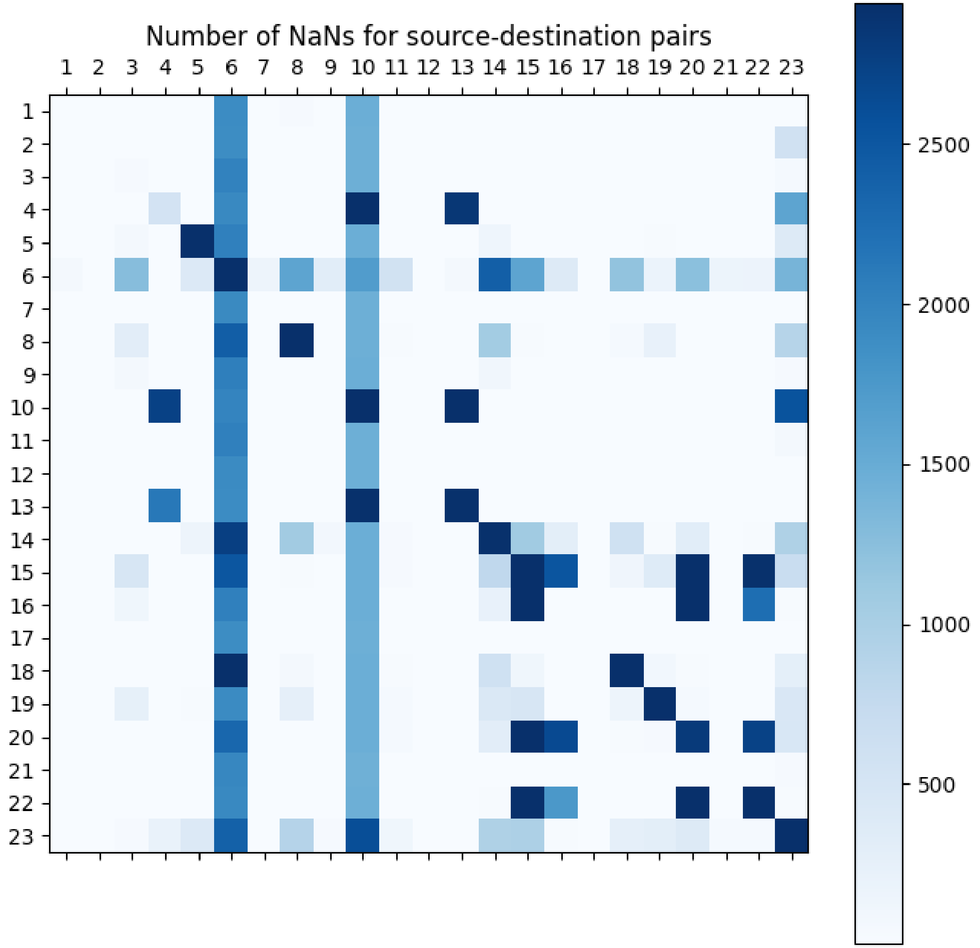
- 1 [day] = 24 [h] = 96 [sample a day]
- 31 [days] ⇒ 31*96 = 2976 [expected sample] != 2941 [dataset sample]

- There are 35 missing files
- Not much compared to the amount of data we have
- We filled missing values with NaNs
- All .xml are parsed and handled as NumPy array

```
Missing files: ['IntraTM-2005-01-01-00-00.xml' 'IntraTM-2005-01-01-00-15.xml'
 'IntraTM-2005-01-15-19-00.xml' 'IntraTM-2005-01-15-19-15.xml'
 'IntraTM-2005-01-15-19-30.xml' 'IntraTM-2005-01-15-19-45.xml'
 'IntraTM-2005-01-15-20-00.xml' 'IntraTM-2005-01-15-20-15.xml'
 'IntraTM-2005-01-15-20-30.xml' 'IntraTM-2005-01-15-20-45.xml'
 'IntraTM-2005-01-15-21-00.xml' 'IntraTM-2005-01-15-21-15.xml'
 'IntraTM-2005-01-15-21-30.xml' 'IntraTM-2005-01-15-21-45.xml'
 'IntraTM-2005-01-15-22-00.xml' 'IntraTM-2005-01-15-22-15.xml'
 'IntraTM-2005-01-15-22-30.xml' 'IntraTM-2005-01-15-22-45.xml'
 'IntraTM-2005-01-15-23-00.xml' 'IntraTM-2005-01-15-23-15.xml'
 'IntraTM-2005-01-15-23-30.xml' 'IntraTM-2005-01-15-23-45.xml'
 'IntraTM-2005-01-16-00-00.xml' 'IntraTM-2005-01-16-00-15.xml'
 'IntraTM-2005-01-16-00-30.xml' 'IntraTM-2005-01-16-00-45.xml'
 'IntraTM-2005-01-16-01-00.xml' 'IntraTM-2005-01-16-01-15.xml'
 'IntraTM-2005-01-16-01-30.xml' 'IntraTM-2005-01-24-02-00.xml'
 'IntraTM-2005-01-24-02-15.xml' 'IntraTM-2005-01-24-02-30.xml'
 'IntraTM-2005-01-28-00-45.xml' 'IntraTM-2005-01-28-03-45.xml'
 'IntraTM-2005-01-28-04-00.xml']
```

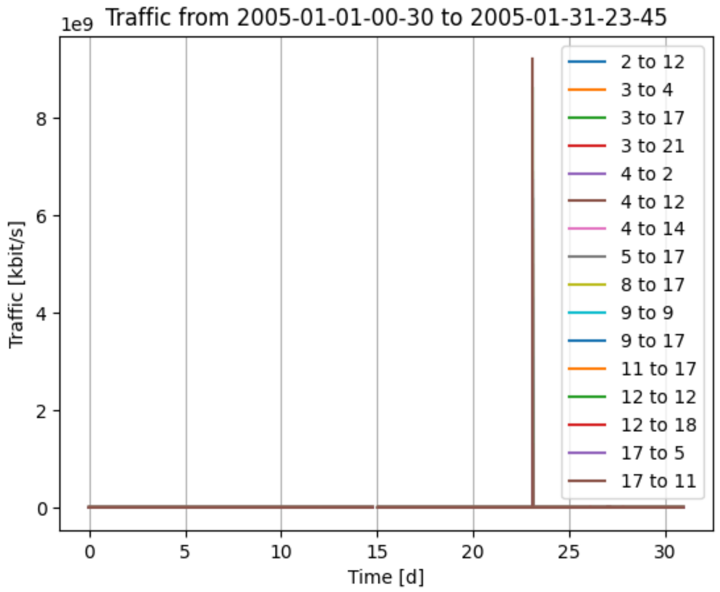
Dataset – Missing Data

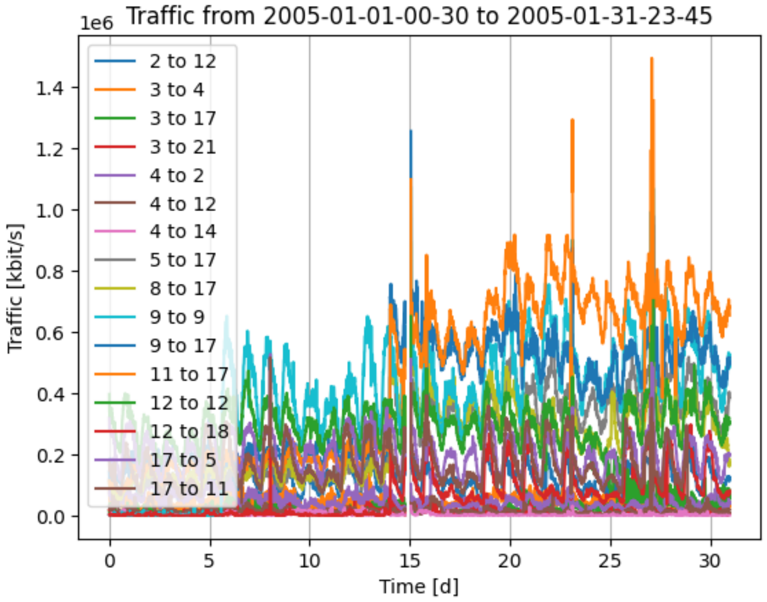
- The network consists of 23 routers
- This means we have 23x23 feasible pairs such that
$$(i, j) : i, j \in [1, 23]$$
- Not all the pairs have a significant amount of traffic in the whole month
- For each tuple we checked the numbers of NaNs and plotted it onto a matrix
- Discarded pairs that contain more than 100 NaNs



Data – Corrupted File

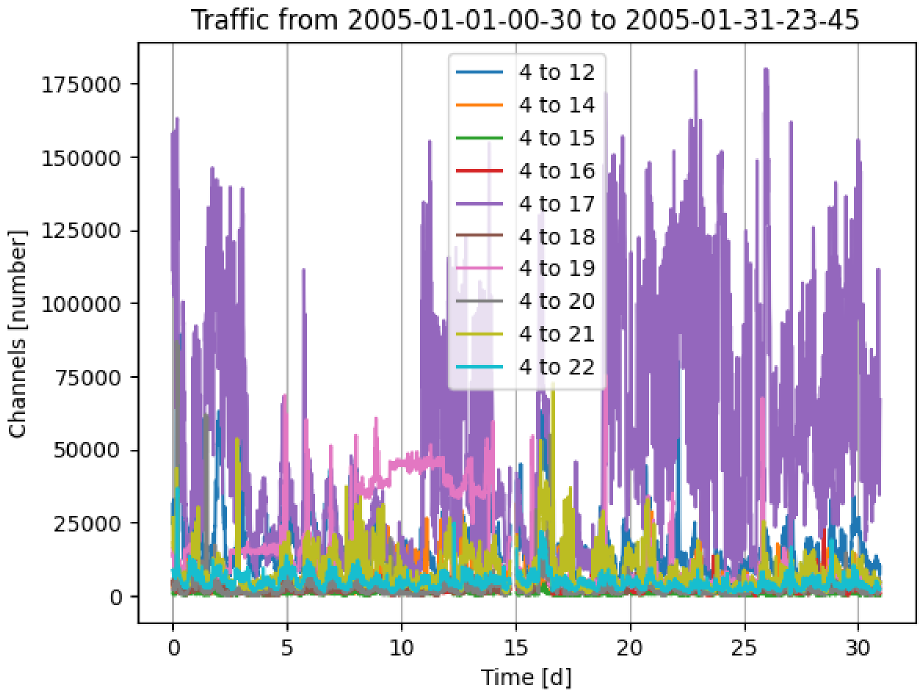
- Noticed that there are some huge outliers at a specific point (in the order of 10^9)
- So, we plotted the traffic trace of pairs containing that data
- They all overlap and overwhelm other data by a significant amount
- Since this is infeasible, especially in 2005, we removed these values and plotted the trace again
- The second graph looks as expected
- Since all that values come from the same file, we decided to classify that file as corrupted and discard it, replacing its values with NaNs

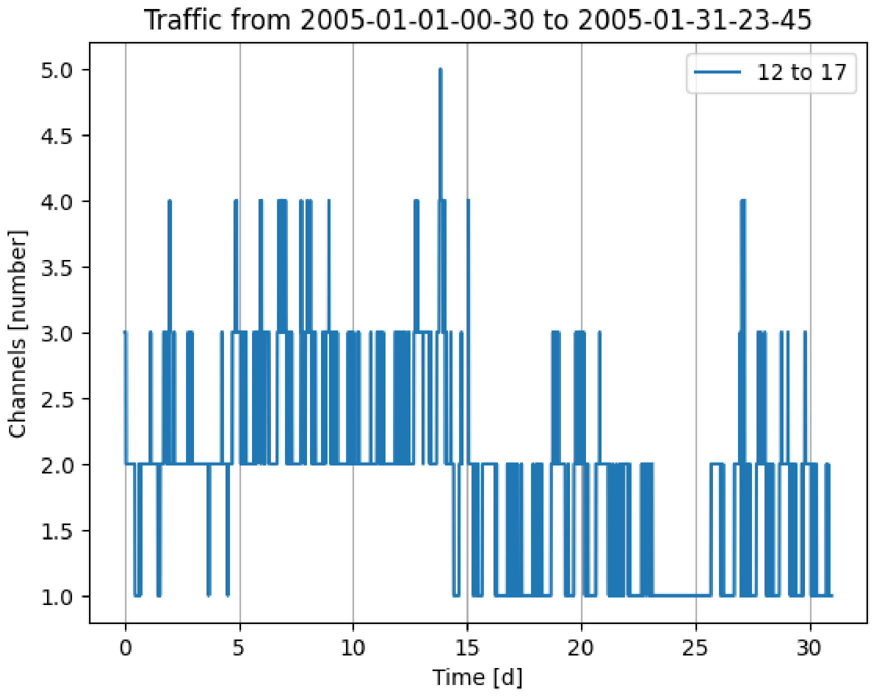




Pairs Selection – Graphical check

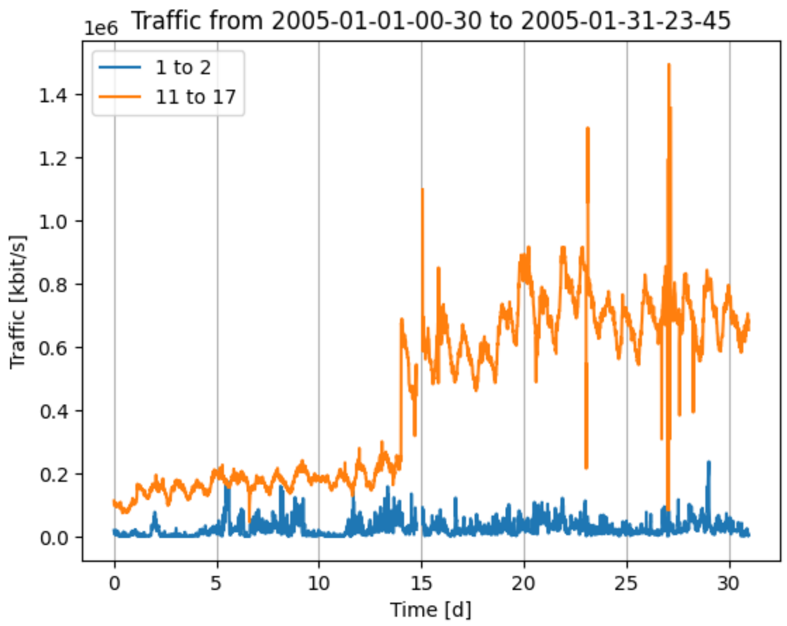
- After discard pairs with more than 100 NaNs we have 398 pairs
- We did a graphical check of the time series
- This is useful to get an idea of the traffic shape, the amount of data is too big to decide only on graphical evidence

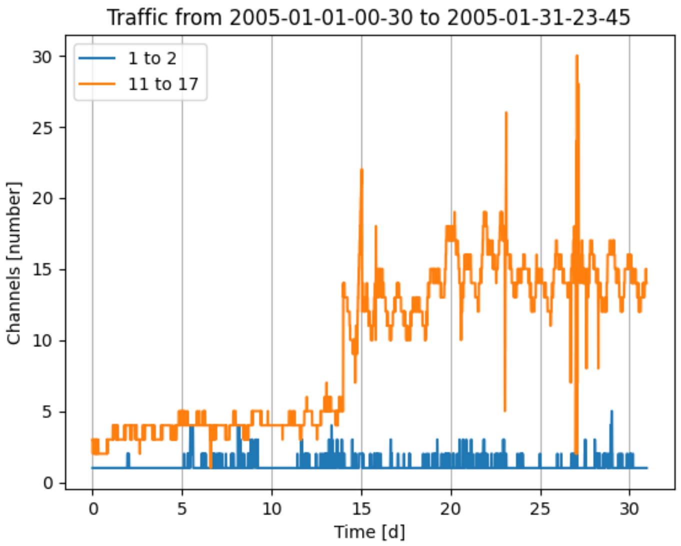




Pairs Selection – Variance check

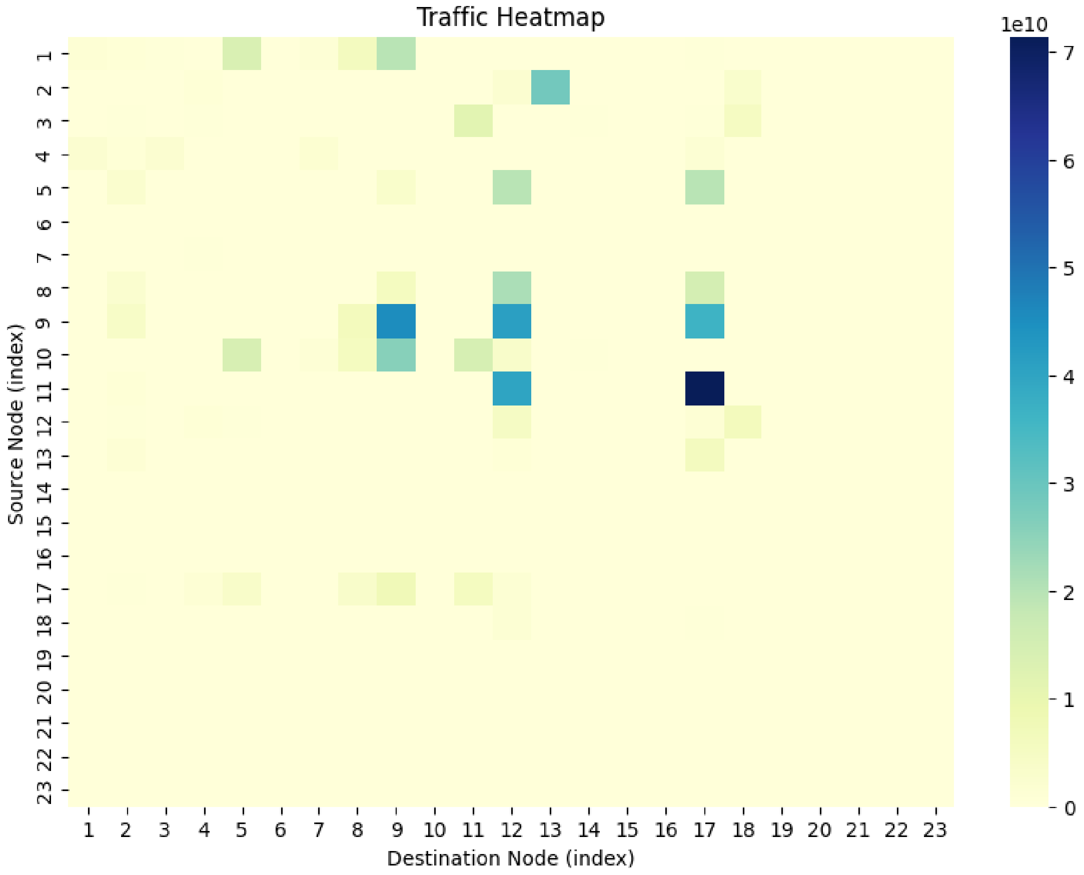
- We then checked the variance of the channels, ignoring the NaNs we added
- More than half of pairs have 0 variance, which means constant channels over time
- So, we manually checked the variances of the tuples and decide to discard tuples which have a variance < 0.1
- To decide this threshold we plotted different traffic traces with a known variance and looking at the shape of the graph we concluded the following
- Everything below 0.01 is almost constant traffic, so it can be discarded
- Everything above 1 clearly has a very fluctuant shape
- Looking at the shape of 0.01 graphs and 1 graphs we concluded that 0.1 could be a reasonable threshold
- **In the end, 52 remaining candidate pairs, which is a very reasonable amount of traffic for model fitting**





Pairs Selection – Filling the NaNs

- The remaining 52 pairs still contains a few NaNs
- We used *NumPy.interp()* to make a linear interpolation of the missing values
- The heatmap shows the traffic we got after filtering and reshaping for each tuple

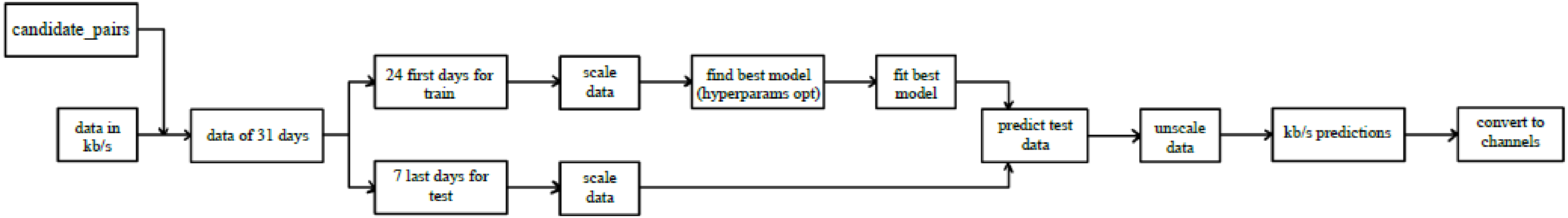


Pairs Selection – Overview

Traffic Prediction – ML algorithms

- **LSTM – Long Short-Term Memory ←**
- **Feature-based approach:**
  - LinReg – Linear Regression
  - ANN – Artificial Neural Network

LSTM – kb/s Prediction



- We got a subset of source-destination pairs called *candidate_pairs*
- Each one contains traffic for 31 days
- First 24 days are taken as train set and last 7 as test set
- This is done for each candidate pair and train and test split are concatenated
- Used train set to find scaling parameters, also used them for test data

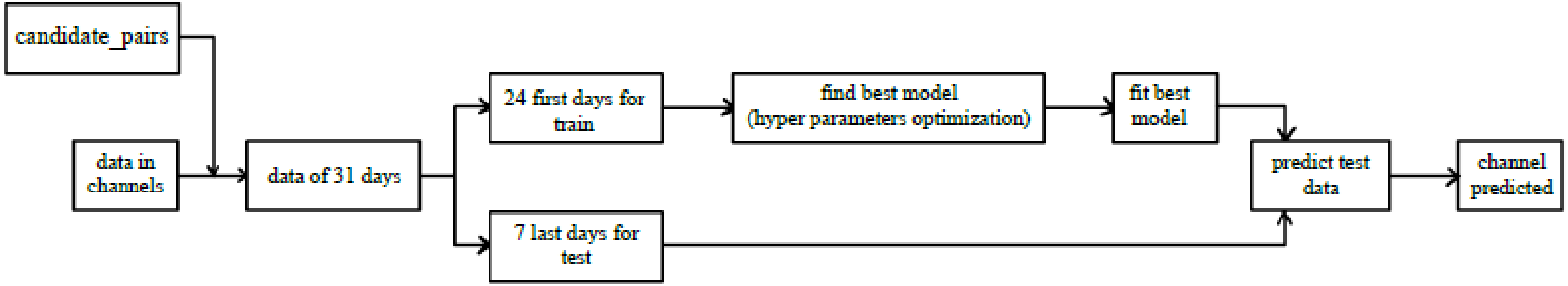
LSTM – Model Fitting

- In order to find the best model, we tried different parameters combination using a grid search
- We used the model that obtained the highest MSE, MAE and R^2 score.

LSTM kb/s Best Model	
gap	[16]
n_epochs	[50]
n_layers	[7]
n_neurons	[[18, 15, 12, 12, 9, 6, 3]]
opti	['adam']
thinkback	[16]

	thinkback	gap	n_layers	n_neurons	n_epochs	opti	mse	mae	r2
0	8	8	3	[10, 7, 3]	20	adam	0.000545	0.008714	0.959872
1	8	8	5	[15, 12, 9, 6, 3]	20	adam	0.000554	0.007934	0.959184
2	8	8	7	[18, 15, 12, 12, 9, 6, 3]	20	adam	0.000584	0.007869	0.956979
3	16	16	3	[10, 7, 3]	20	adam	0.000571	0.007928	0.957953
4	16	16	5	[15, 12, 9, 6, 3]	20	adam	0.000590	0.007875	0.956552
5	16	16	7	[18, 15, 12, 12, 9, 6, 3]	20	adam	0.000538	0.008491	0.960400
6	24	24	3	[10, 7, 3]	20	adam	0.000609	0.008478	0.957774
7	24	24	5	[15, 12, 9, 6, 3]	20	adam	0.000597	0.008184	0.958577
8	24	24	7	[18, 15, 12, 12, 9, 6, 3]	20	adam	0.000643	0.008678	0.955434
9	32	32	3	[10, 7, 3]	20	adam	0.000639	0.008296	0.955730
10	32	32	5	[15, 12, 9, 6, 3]	20	adam	0.000602	0.009790	0.958264
11	32	32	7	[18, 15, 12, 12, 9, 6, 3]	20	adam	0.000598	0.007919	0.958545

LSTM – Channels Prediction

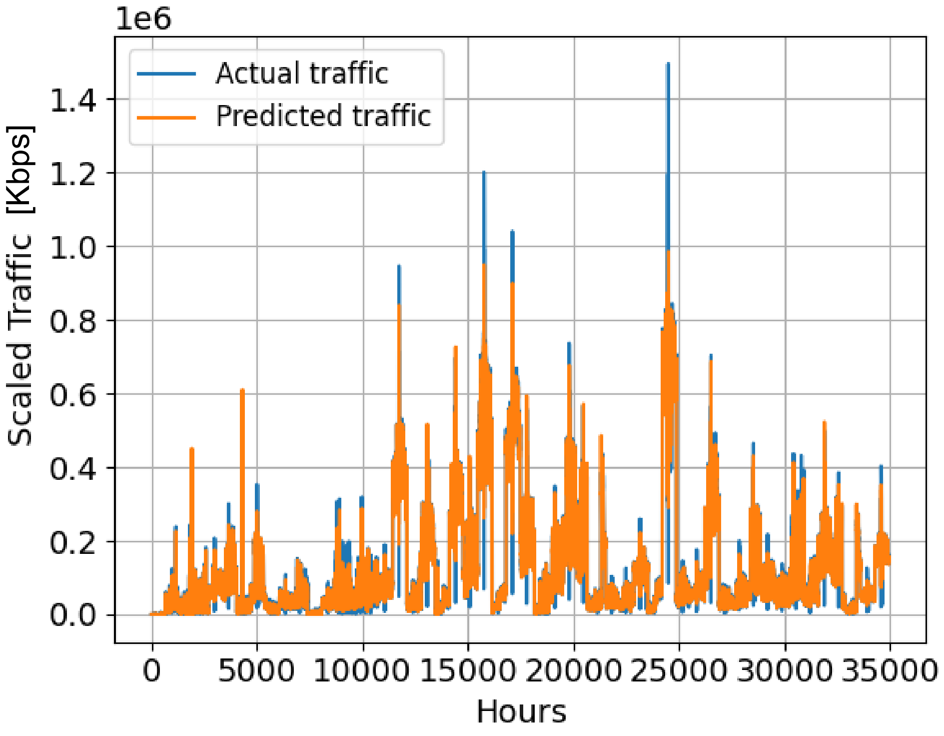


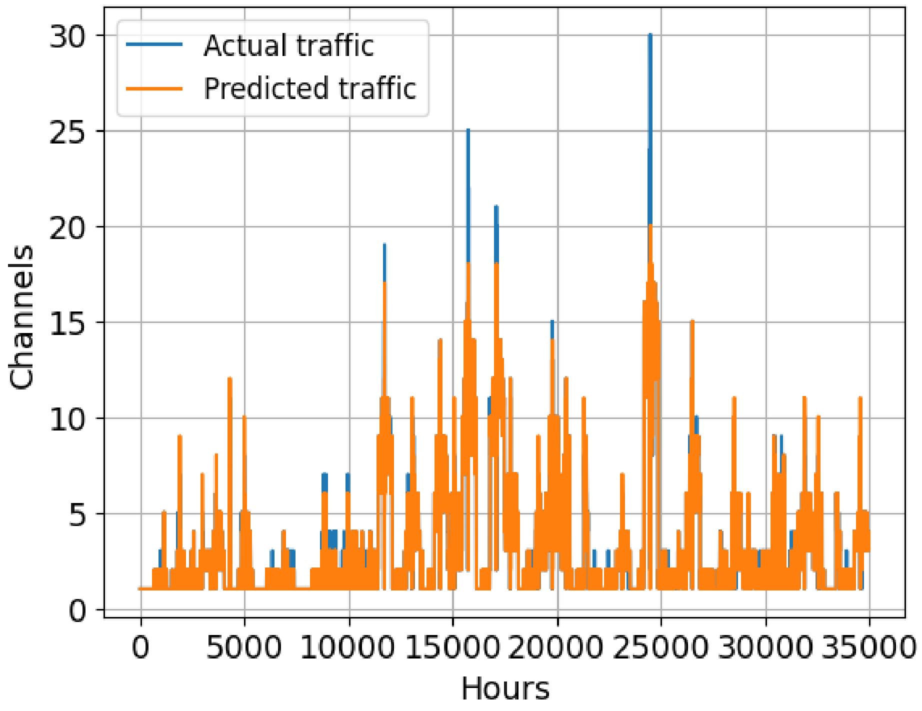
- The incoming data in this case are directly the channels
- Best model is obtained in the same way as before (it comes out they are the same)
- Channels required are the prediction’s results

LSTM – kb/s vs Channel

LSTM kb/s	
over	3757
n_over	3057
under	2712
n_under	2147
incorrect ratio	0,14892
ratio_over	0,08748
ratio_under	0,06144
mse	847213547
mae	9853
r2	0,96278

LSTM Channels	
over	3017
n_over	2475
under	3708
n_under	2811
incorrect ratio	0,15127
ratio_over	0,07083
ratio_under	0,08044
mse	0,49476
mae	0,19245
r2	0,94483





Traffic Prediction – ML algorithms

- **LSTM – Long Short-Term Memory**
- **Feature-based approach: ←**
  - LinReg – Linear Regression
  - ANN – Artificial Neural Network

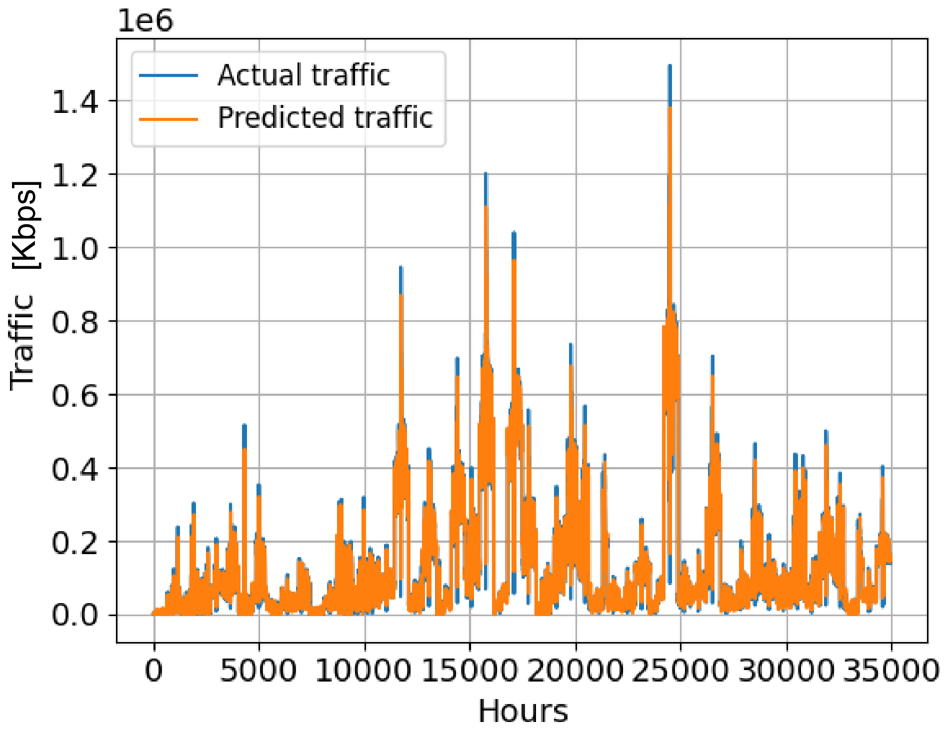
Features

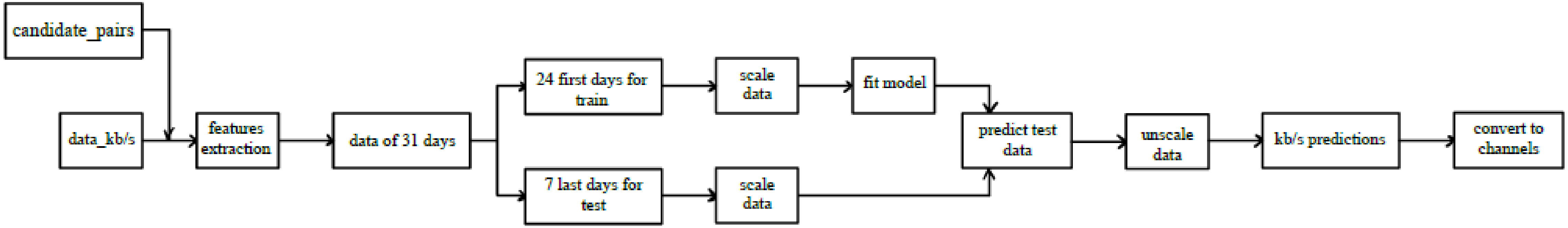
- These are the features extracted, scaled and used to train the LinReg model

Var name	Type	Description
src	int [1,23]	Index of the source node
dst	int [1,23]	Index of the destination node
day	int [1,7]	Day of the week
working	bool	True if it is a working day, else false
hour	Int [0,24]	Hour of day at which traffic has ben measured
prev_hour	int	Traffic sample for the previous hour
prev12_hours	int	Traffic sample for the previous 12 hours
prev_day	int	Traffic sample for the previous day

LinReg – kb/s Prediction

1. From all the data and the pairs we chosen in the initial part we extract the features and obtain the dataset
2. Dataset is split in train (first 24 days) and test (last 7 days) set
3. Scale the train and test set using MinMaxScaler
4. Fit the Linear Regression model with train set
5. Predict the test set
6. Unscale the data
7. Obtain the prediction in terms of kbit and then converted into channels

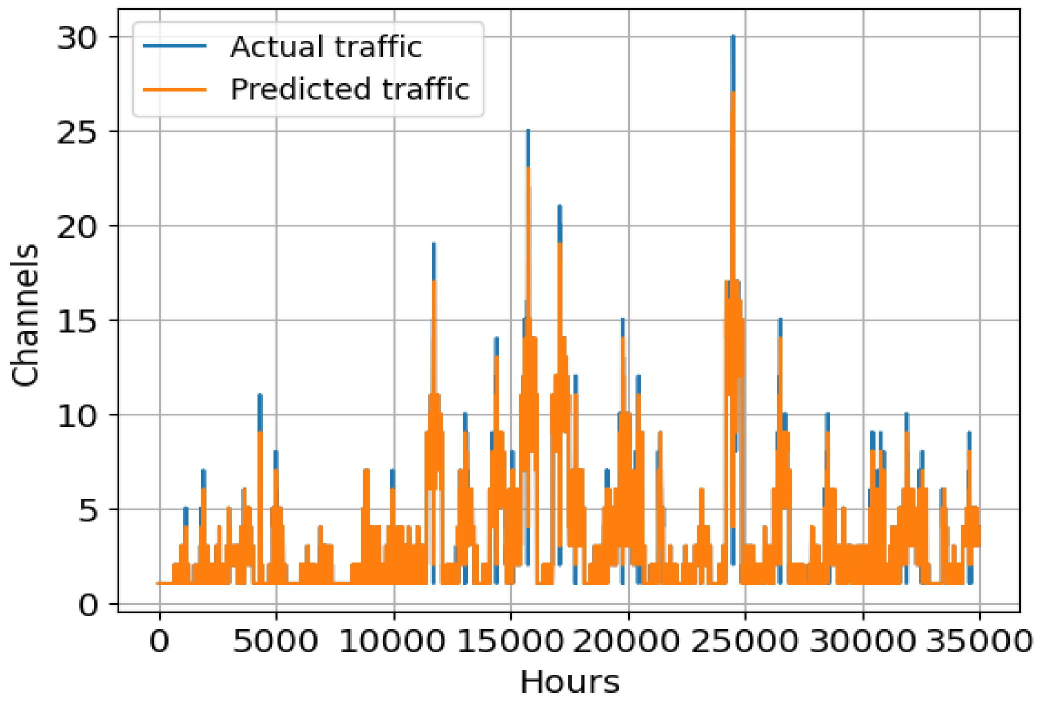


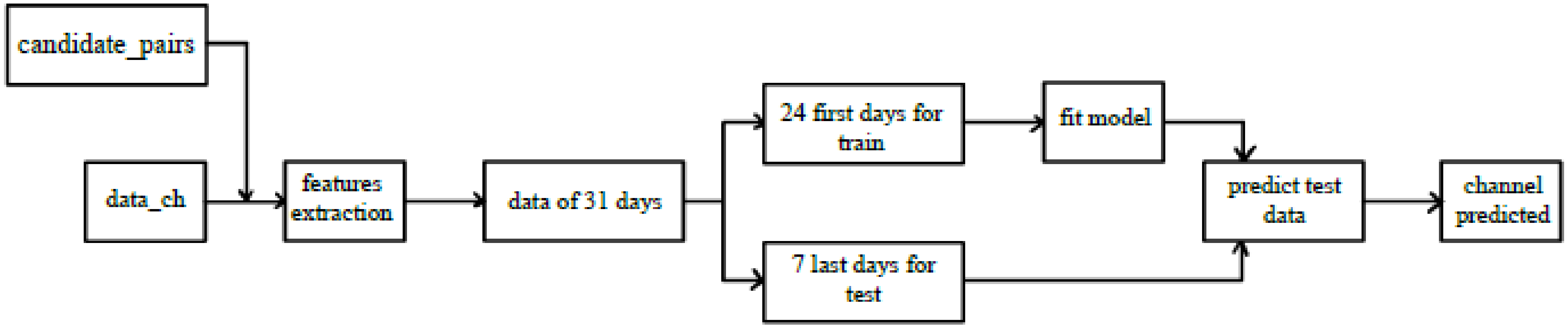


LinReg – Channels Prediction

- Build and fit the model with the data referring to channels.
- Only use unscaled dataset.

- The steps are basically the same as before but here the **model predicts directly the number of channels required to manage the traffic**.

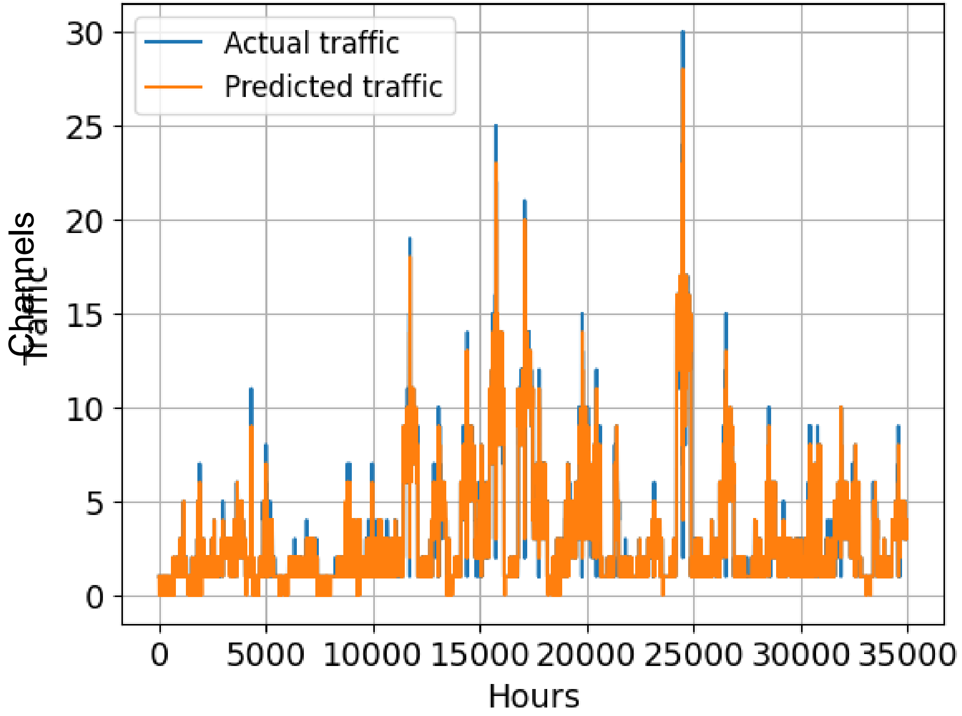


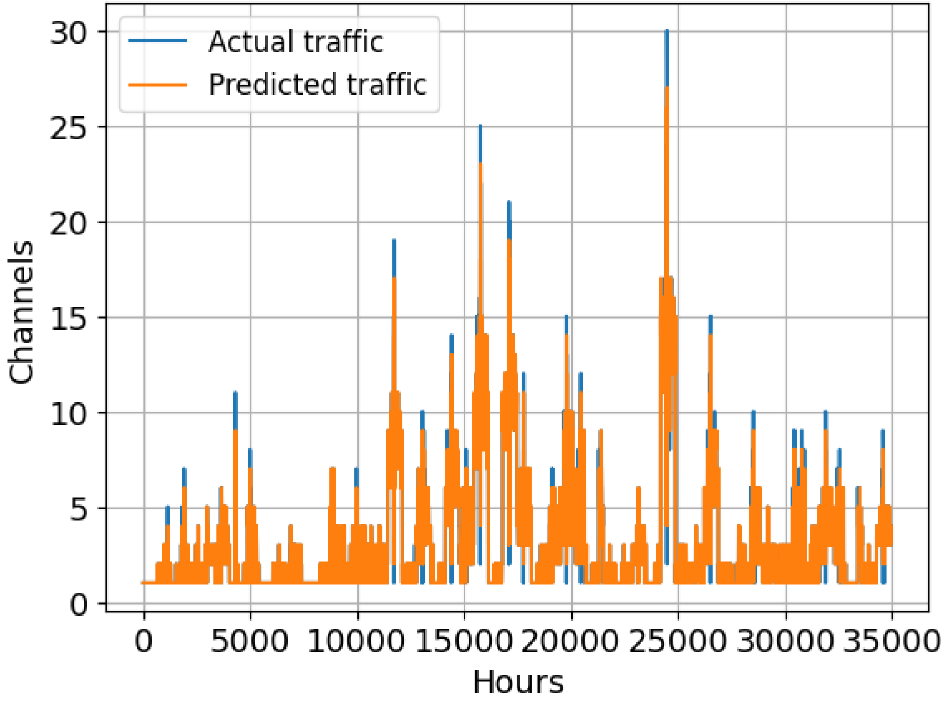


LinReg – kb/s vs Channels

LINREG Kb/s	
over	5939
n_over	4690
under	4940
n_under	3987
incorrect ratio	0,24831
ratio_over	0,13421
ratio_under	0,11410
mse	0,69966
mae	0,31133
r2	0,92199

LINREG Channels	
over	5460
n_over	4129
under	5167
n_under	4161
incorrect ratio	0,23724
ratio_over	0,11816
ratio_under	0,11908
mse	0,69583
mae	0,30412
r2	0,92241





ANN – Hyperparameters Optimization

- ANN hyperparameters are chosen using a grid search over a set of parameters defined in a parameters grid (shown below)
- The best ones in terms of MSE and R2 are chosen in order to predict the traffic of test set
- Finally, the best ANN model is built and fit with 100 epochs in order to obtain better results

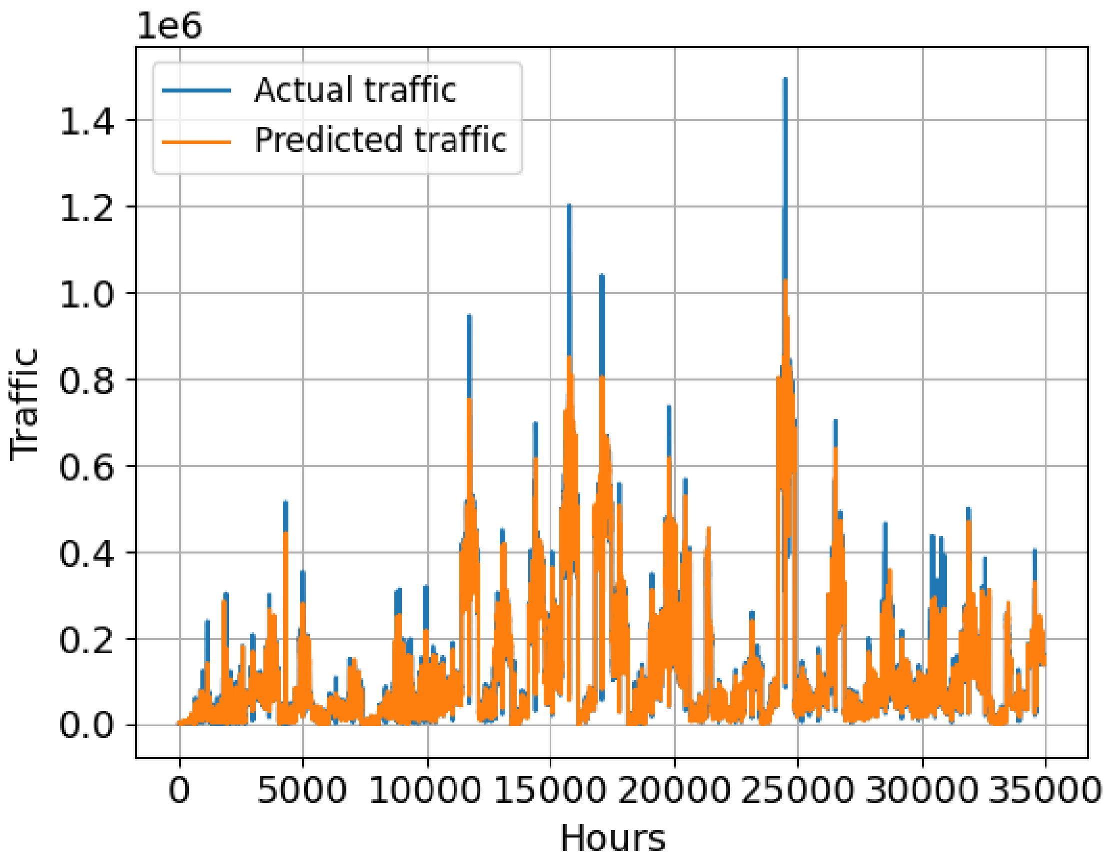
ANN parameters grid	
Layers	3, 5, 7
Neurons	[15,10,5], [21,18,15,12,9], [24,21,18,15,12,9,6]
Activation	‘sigmoid’, ‘relu’
Epochs	20

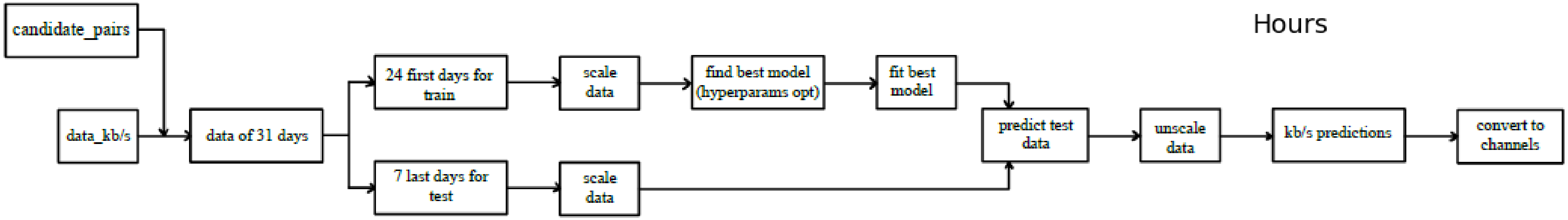
Grid search

ANN best parameters	
Layers	**7**
Neurons	**[24,21,18,15,12,9,6]**
Activation	**‘relu’**
Epochs	**20**

ANN – kb/s Prediction

1. From all data and the pairs we chose initially we extract the features and obtain the dataset
2. Dataset is split in train (first 24 days) and test (last 7 days) set
3. Scale the train and test set using MinMaxScaler
4. Optimize the model
5. Predict the test set
6. Unscale the data
7. Obtain the prediction in terms of kbit and then converted into channels

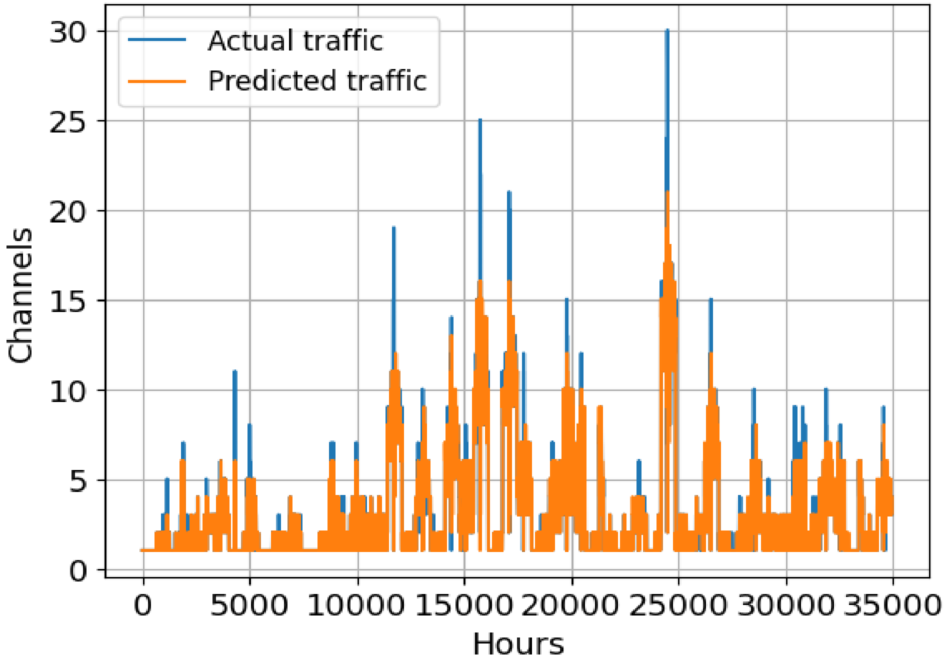


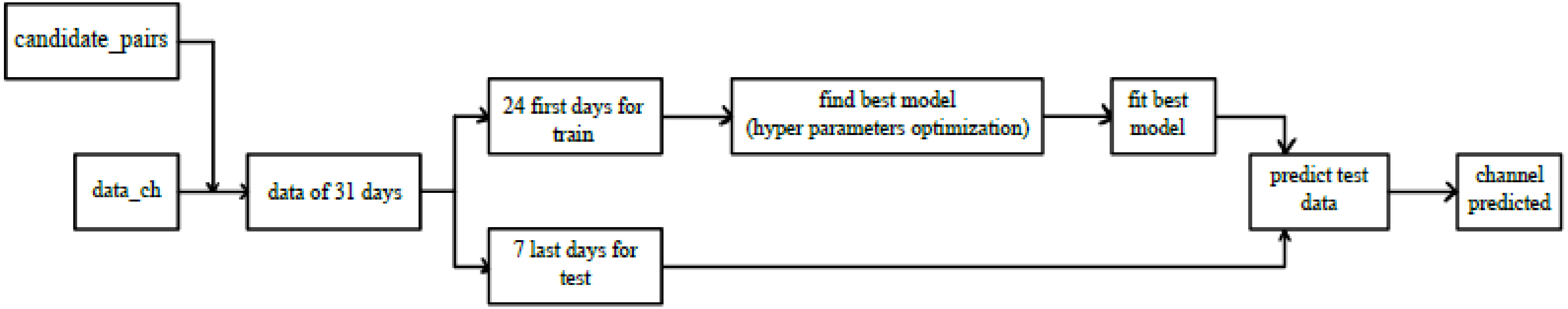


ANN – Channel Prediction

- As we did with linear regression when we build our model with the data referring to channels.
- We only use unscaled dataset.

- The steps are basically the same as before but here the **model predicts directly the number of channels required to manage the traffic**.

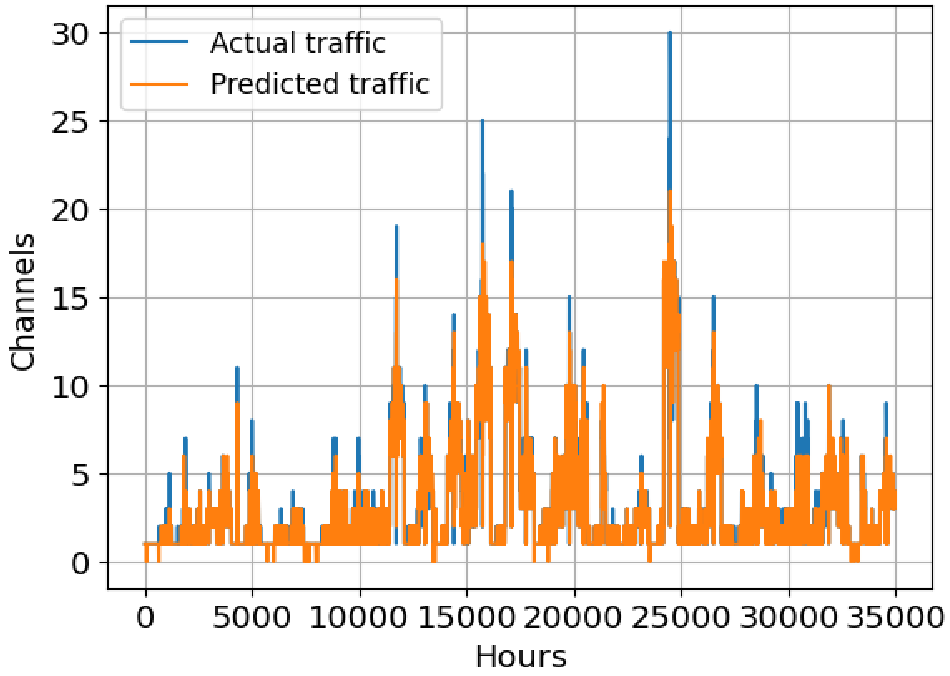


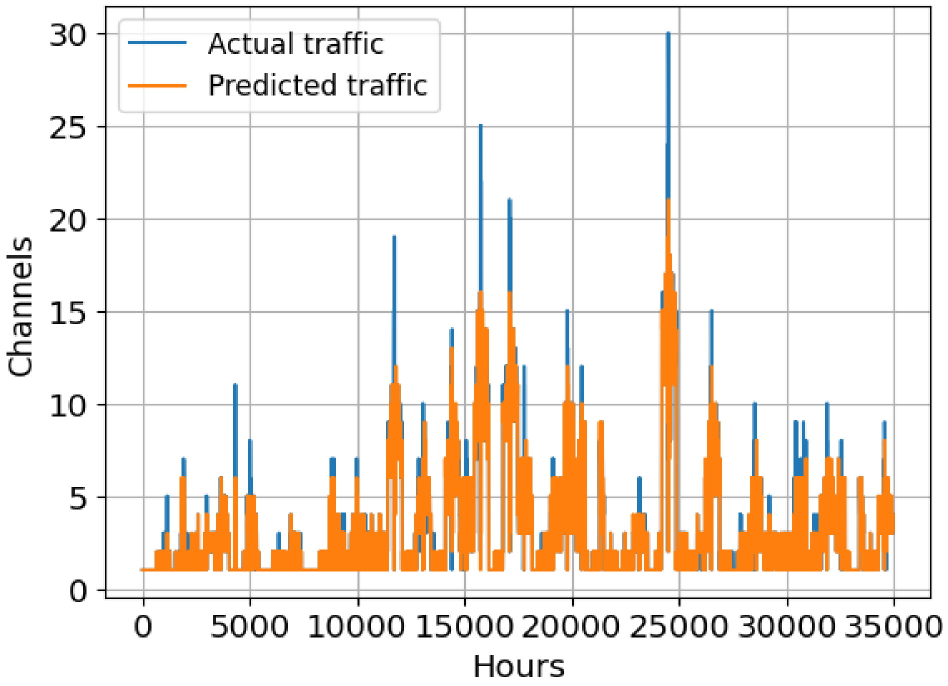


ANN – kb/s vs Channel

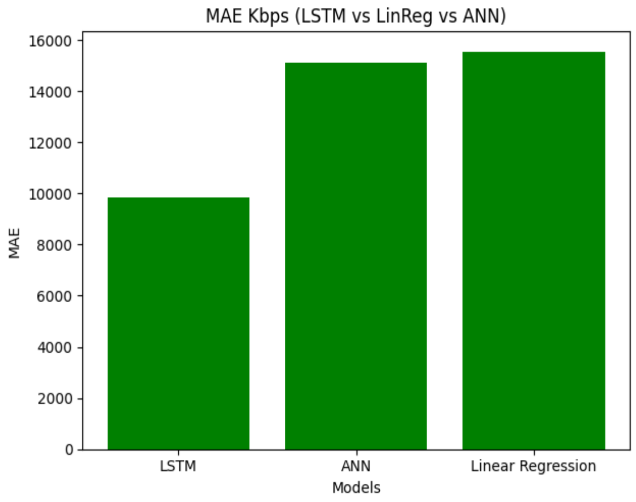
ANN kb/s	
over	5452
n_over	4196
under	4553
n_under	3700
incorrect ratio	0,22596
ratio_over	0,12008
ratio_under	0,10588
mse	0,65714
mae	0,28632
r2	0,92673

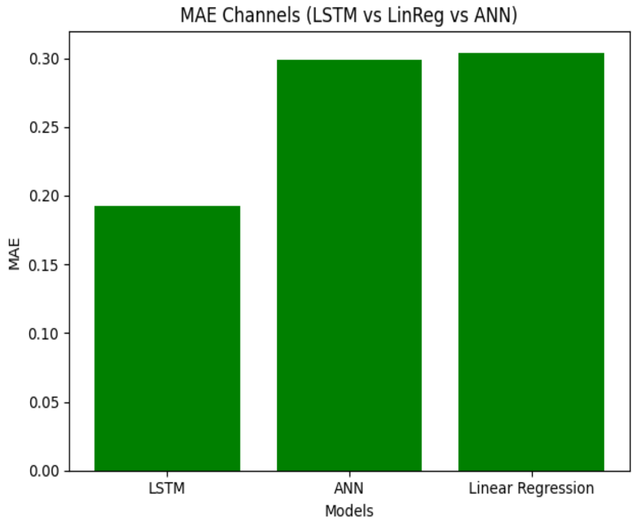
ANN Channels	
over	5366
n_over	4051
under	5062
n_under	4227
incorrect ratio	0,23689
ratio_over	0,11593
ratio_under	0,12096
mse	0,67113
mae	0,29842
r2	0,92517

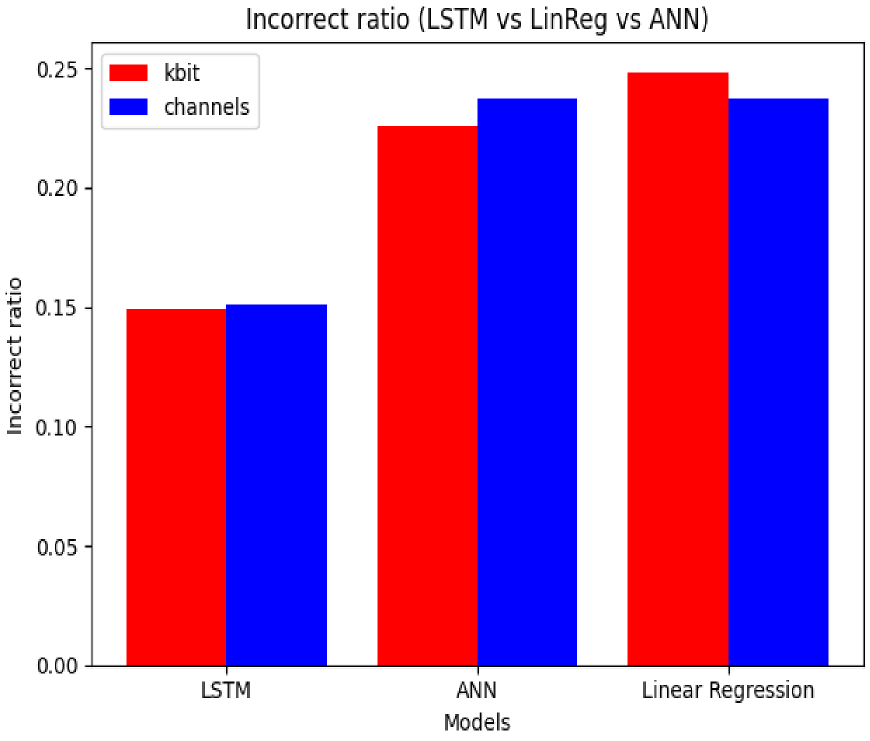




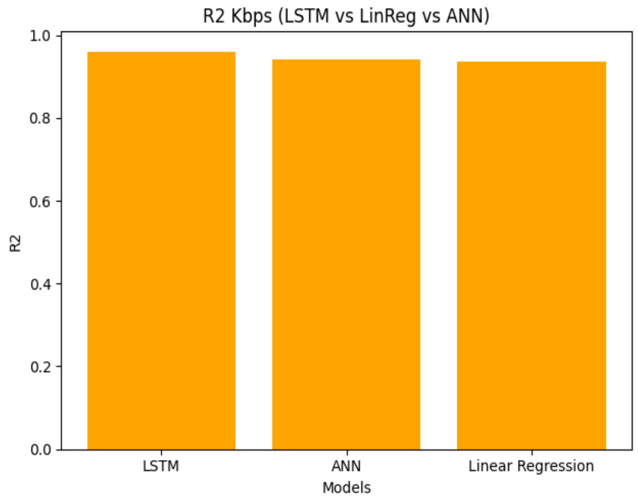
Comparison: MAE - Incorrect Ratio – R2

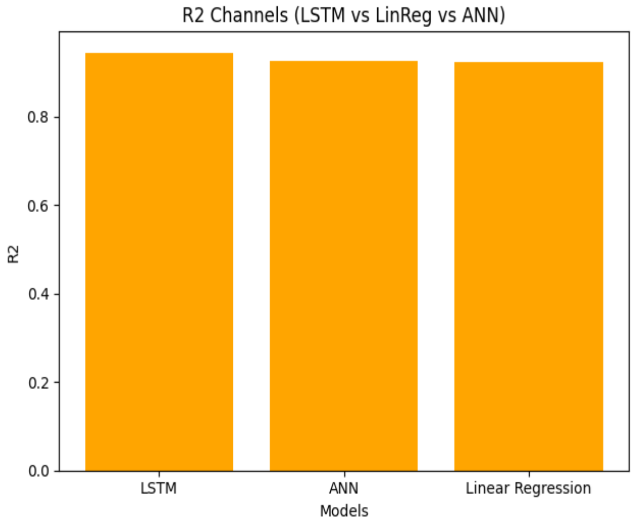






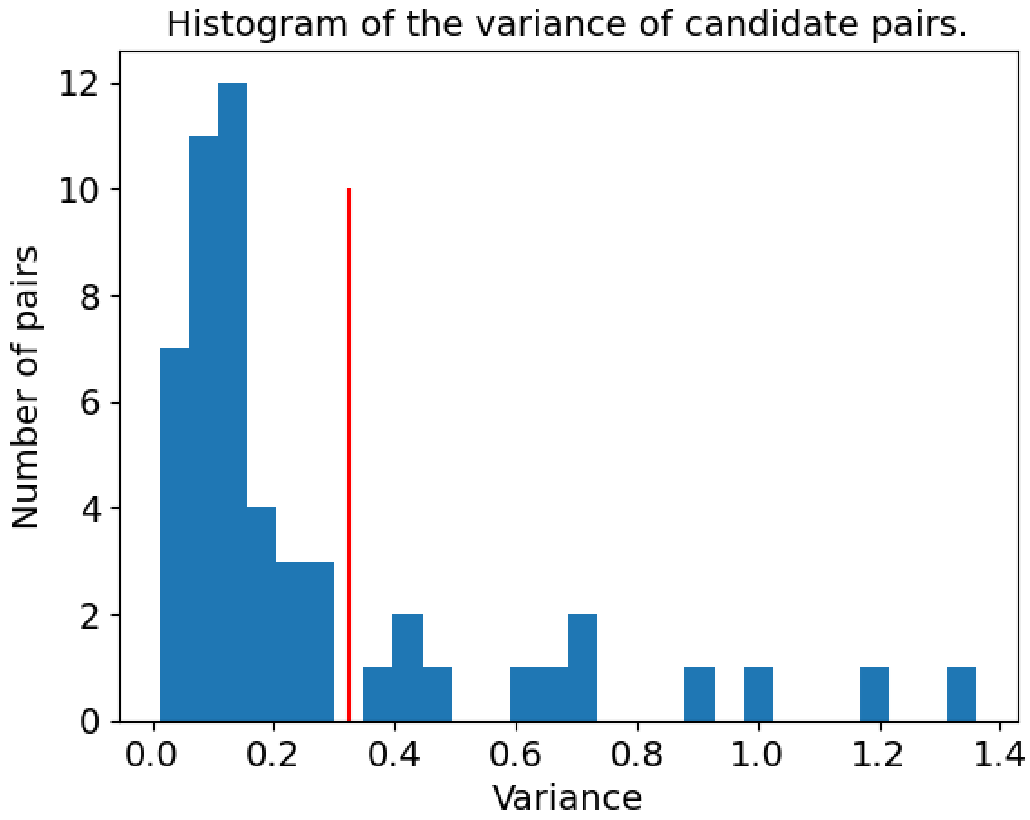
- Results are averaged across 52 src-dst pairs
- LSTM performs way better than ANN and LinReg
- Difference between predicting kb/s and predicting channels smaller than expected
  - For LSTM and ANN it’s better to predict traffic and then convert it to channels
  - For LinReg it is better to predict directly channels



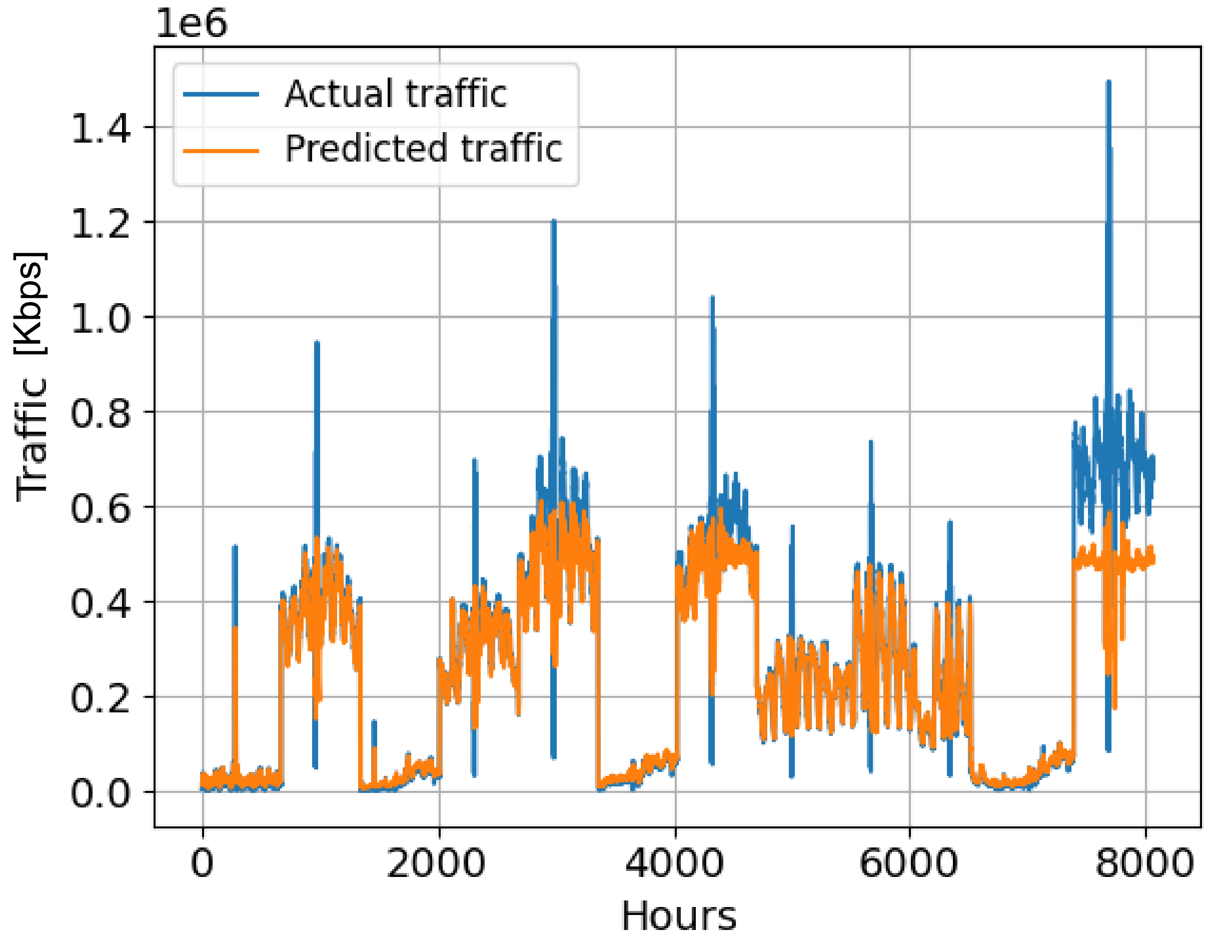


Transfer Learning – Different Approach

- Until now we trained our models using the first 24 days of the time series and used the last 7 to evaluate it (of the whole candidate pairs)
- Now the model will be trained on the entire series of a part of the 52 candidate pairs, and tested on the remaining pairs
- We split the candidate pairs in 2 groups
- Train on low variance couples
- Test on high variance couples
- We do only consider LSTM



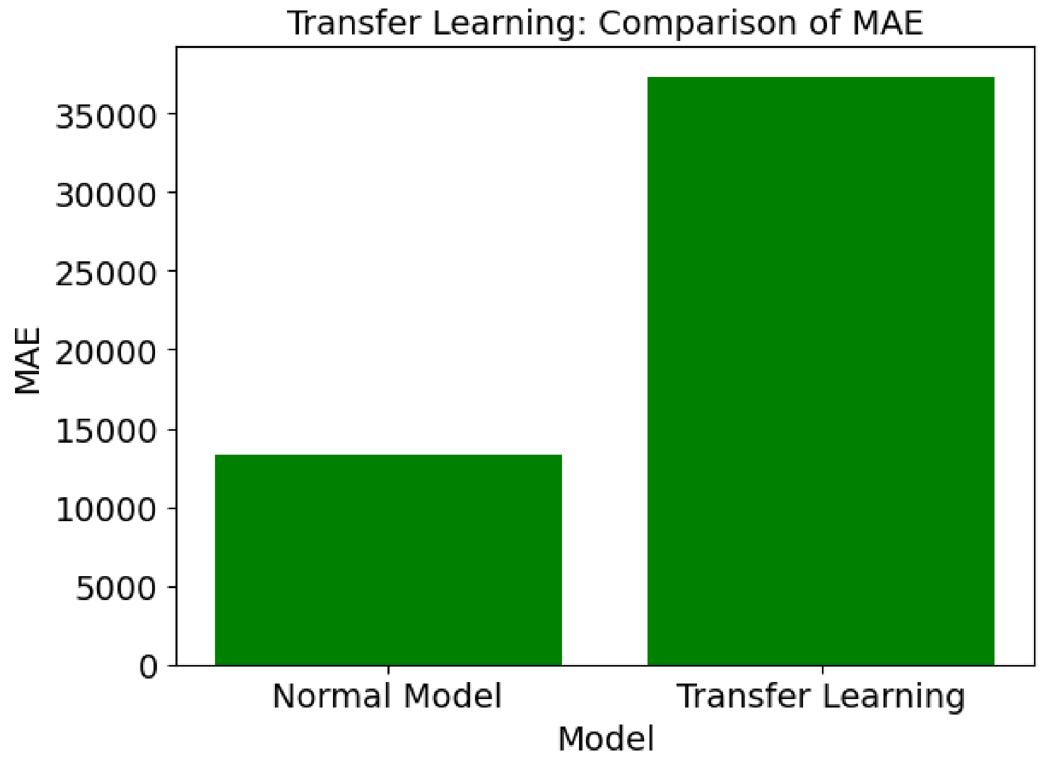
Transfer Learning – LSTM Predictions

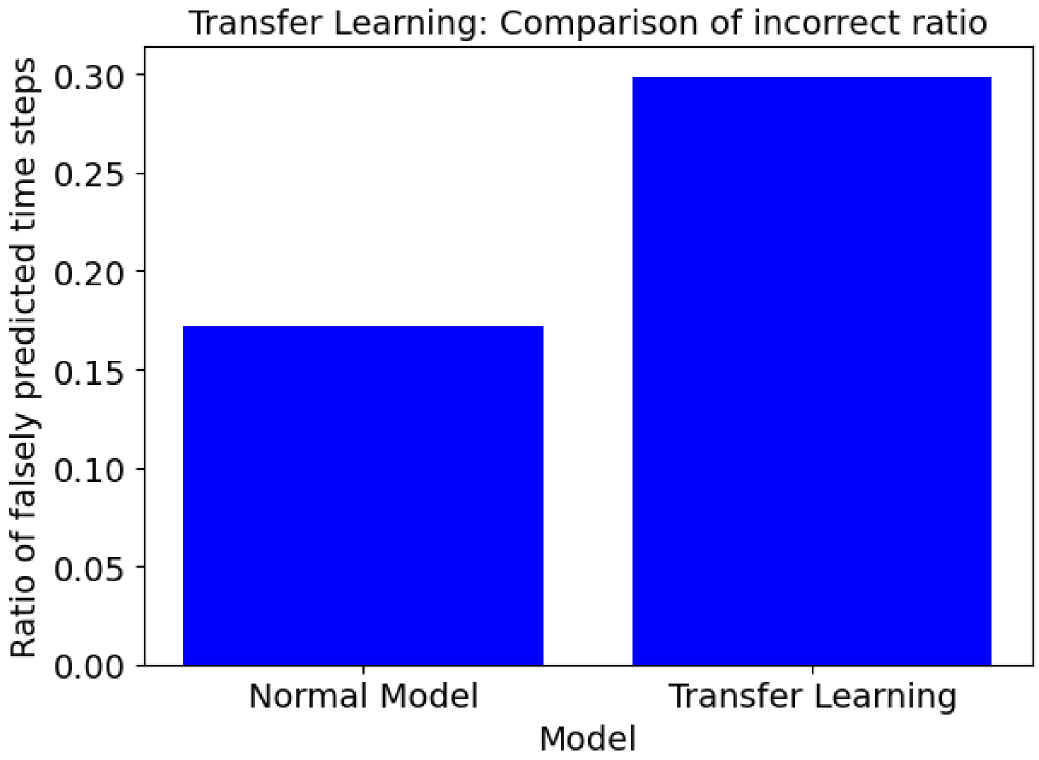


- It follows more or less the same path
- It performs worse
- It does not predict high values

Transfer Learning – Comparison

- Here: Results are averaged across 12 high-variance source-destination pairs
- For transfer learning only channels-predicting models have been used





Conclusions

- LSTM performs way better than ANN and LinReg
- LSTM took about 10 hours to train and fit, ANN 30 minutes. So, in some cases ANN could be a better choice
- LinReg took some seconds to train and it is the only one that has higher results in predicting directly channels

- In some scenarios where data availability is limited, transfer learning could be a good choice and prove to be very useful.
- While transfer learning may not always deliver exceptional performance, the overall results are acceptable and demonstrate its practical value.