# Enhancing Vision-based Vehicle Detection with Real-time Traffic Features

Xuan Li

## 1 Introduction

Given the rapid development in visual analytics, data sensing and transmission, an integration of the rich and diverse information would bring new perspective to urban traffic tracking for Intelligent Transportation Systems (ITS), which aims at improving urban mobility, reducing congestion and boosting travel safety by analyzing key traffic parameters including speed, volume, and density. Traditionally, these parameters are mainly inferred from data captured by loop detectors, which is costly and only sparsely deployed in city scale. As an alternative, vision-based inference have been widely applied due to high precision of convolutional models as well as cost effectiveness of camera sensors. However, in this work we explore the possibility of further enhancing the vision-based approach by fusing the model with features captured by existing transportation sensors. Specifically, by leveraging public bus transit and inrix data with images from a single camera, the authors have designed a convolutional neural network for real-time vehicle density estimation. Compared with pure visual techniques, the authors have validated that the inclusion of domain features can improve the overall performance, and proposed to expand the model for other transportation related tasks such as speed estimation by vehicle tracking.

## 2 Data Collection and Feature Processing

The data used are sampled at: 09/06/2018, from 14:00 - 15:00; 09/13/2018, from 14:00 - 15:00; and 09/14/2018, from 17:00 - 18:10, during rush hour. Each of the image is sampled at every other 5 second, so in total there are 2200 images. For this experiment I use 1100 images for training and the other 1100 for testing. In addition, to make the prediction more robust **data augmentation** is applied to images by adjusting random contrast, brightness, and adding Gaussian noise. In this case this option is named with tag ‘aug’ in this paper.

- **image**: from the camera located in front of CMU garage, as it is showed in Figure 1 marked with red boundary. The data is collected at 1 fps[1], For each frame the labelled variables are the four parameter $x, y, h, w$, the coordinate of the upperleft corner, the height, and width of the box drawn around each vehicle appeared in the region of interest (non-black region). By assigning a 2D Gaussian distribution given the parameter of each bounding box, finally we have the ground truth as the density of vehicle, and the total number of vehicle (sum of density). After cropping and resizing, the raw image is used as the first input feature for the model.

- **bus API**: during the meantime we also collect the information of all buses (route) that pass Forbes Avenue, which are 61A, 61B, 61C, 61D, 67, 69, 58, 28X. The features are retrieved from the API supported by PortAuthority, which contains information like bus route, bus vid, destination, latitude, longitude, speed, etc. In align with the sampling frequency as the image

[1]fps: frame per second

data source, the information are queried and saved at the rate of every 5 second. The selection and configuration of bus features are described as follows:

- the categories of the features are (for each bus): the current latitude, longitude, speed, heading direction, indicator of traffic congestion (0, 1).
- for each direction (inbound, outbound), the upstream regions (where bus comes) are segmented according to the location of bus stop, as the example showed in Fig 2. The locations of the four corners are saved and would be used to justify at each timestamp a bus is located in that region. For each route of bus, 5 regions are selected for each route.
- for each region, for each route of bus, at each timestamp, use the latitude, longitude of the bus to justify whether the bus is in the region. If true, record its corresponding features.
- at each timestamp, we collect the previous one minute (including the current timestamp) features of all the bus route, so the dimension of the feature would be

  $$12 \times 350$$

  where 12 represent the total number of timestamp in the past minute, and $350(14 \times 5 \times 5)$, where 14 the number of direction (two direction for all the routes except 58, 28X), 5 is the number of regions, and the last 5 is the number of features.

- **INRIX**: The third set of available features are speed information on Forbes Avenue retrieved from INRIX. Basically, the features are the traffic flow speed, average speed, reference speed, travel time, which are averaged every other 5 minutes on inbound and outbound direction. Thus, to include these features in the model, at each timestamp during the past 5 minutes we used the current averaged feature. The detail of the implementation is showed in Fig 8. Also, using the features of previous one hour the dimension of this input is

  $$12 \times 8$$

- **Weather**: The fourth set of features are retrieved weather underground API, where is information is updated every hour. The features include: temperature, dewpoint, humidity, ..., pavement condition, icon (clear, cloudy, etc), which are 20 in total considering the converted one-hot vector for the category feature icon. Since the sampling frequency of this set of feature is quite slow, in this case we also incorporate it as temporal features with timestamp as 3, which is the current plus previous two hours' weather condition. Thus, the dimension of the feature would be

  $$3 \times 20$$

The reasons for the setting of the features are based on the following assumption and observations:

1. at each timestamp for each region, for either inbound/outbound direction, for each bus route only one bus would appear.
2. The bus location gathered via GPS does not update regularly, for example, every fifteen second. Thus, by gathering the bus information of previous timestamp, we want the model to learn that at current timestamp whether the retrieved bus has passed the region, is passing or will be passing the region.

## 3 Model and Experiment Setup

Based on the features we construct a baseline model (Fig 7) and proposed model (Fig **??**) to compare whether the inclusion of bus features would improve the estimation of bus density.

- **baseline mode_1**: as showed in Fig 7, the processed image (shape: $224 \times 224 \times 3$) is used as the only input. The density map and density (sum of value in density map) are the two output of the model.
- **baseline mode_2**: only using the temporal part (LSTM layers) of the model, the bus feature is the used as the only input, and density (count) is the only output.
- **proposed model**: under a similar architecture as the baseline model, the image is processed using the same configuration used in the baseline model. On the other hand, the bus, INRIX, and weather features are passed to the stacked LSTM[2] layer, and the final density is the summation of the result of image pipeline and bus feature pipeline. The detail is showed in Fig 8. Also, for a fair comparison different combination of features has been explored as well and the features are indexed using suffix as 'b', 'i' and 'w' respectively.

Given this, the total dataset containing 2200 samples are evenly split as training set and testing set of 350 samples. The optimal setting of model is acquired after 200 epoch, considering the epoch that has the minimum mean absolute error.

## 4 Comparison

### 4.1 Criterion

#### 4.1.1 Definition

Mean Absolute Error (MAE) is used to evaluate the performance of the model. MAE is defined as

$$\text{MAE} = \frac{\sum_{i=1}^{n} |D^i_{\text{pred}} - D^i_{\text{true}}|}{n}$$

where $D^i_{\text{pred}}$ and $D^i_{\text{true}}$ are the predicted and ground truth density of the $i^{\text{th}}$ image, and $n$ is the total number of image. Also, mean square error (MSE) is used as the criterion in the loss function during training.

#### 4.1.2 Result

##### 4.1.2.1 comparison with baseline model

Given this, the result on the testing set in terms of MAE of the two model is listed in the table below:

Table 1: MAE on Count Estimation (vs baseline model)

[2]long short-term memory, time-series model

| MAE/Model | Full | with-Bus | without-Bus |
|---|---|---|---|
| Baseline_1 | 0.3779 | 0.5362 | 0.3646 |
| Baseline_2 | 1.4310 | – | – |
| **Proposed_b** | 0.2993 | 0.4137 | 0.2897 |
| **Proposed_b_i** | 0.3356 | 0.4792 | 0.3235 |
| **Proposed_b_w** | 0.3441 | 0.4947 | 0.3315 |
| **Proposed_b_i_w** | 0.2894 | 0.3955 | 0.2804 |

where 'full' means the full testing set, 'with-Bus' means a subset where there is at least one bus appears in the region, and 'without-Bus' means the opposite. Also, 'baseline' points to the baseline model that only uses image as input, 'proposed_b' is the one with temporal bus features, and finally 'proposed_b_i_w' is the model with all three source of information. However, given the current implementation it is observed that the final model performs better than baseline but can hardly compare with the second model without INRIX feature. A possible explanation for this is that the features are sampled at a relatively slow rate (5 minutes vs 5 second), which might not provide sufficient and accurate information for count (density) prediction.

#### 4.1.2.2 comparison for rush, non-rush hours

Currently the data collected on 09/06/2018 and 09/13/2018 are used as non-rush hours case, and data collected from 09/14/2018 is used as rush hour case. The basic statistics are listed in Table 3. The MAE is listed in Table 2.

Table 2: MAE on Count Estimation on rush, non-rush hour

| MAE/Model | Full | non-rush hour | rush hour |
|---|---|---|---|
| Baseline_1 | 0.3779 | 0.3217 | 0.4761 |
| **Proposed_b** | 0.2993 | 0.2544 | 0.3779 |
| **Proposed_b_i** | 0.3356 | 0.2769 | 0.4383 |
| **Proposed_b_w** | 0.3441 | 0.2912 | 0.4368 |
| **Proposed_b_i_w** | 0.2894 | 0.2385 | 0.3784 |

Table 3: Mean, variance of density

| | mean | variance |
|---|---|---|
| 09/06/2018 | 1.1175 | 1.1111 |
| 09/13/2018 | 1.8748 | 1.4786 |
| 09/14/2018 | 2.9561 | 1.6502 |

#### 4.1.2.3 comparison with Yolo

Yolo [3] is one of the state-of-the-art CNN model for object detection. However, since essentially it is a classification method that mainly predict class, to count the number of vehicle the outcome would only be integer (1,2,3, etc). Thus, in order to make a fair comparison with our proposed method, the ground truth and the prediction are rounded to the nearest integer as well. Given this, the result is showed in the table below:

Table 4: MAE on Count Estimation vs Yolo

[3] https://pjreddie.com/darknet/yolo/

| MAE/Model | Full |
|---|---|
| Yolo | 0.8109 |
| Baseline_1 | 0.3364 |
| **Proposed_b** | 0.2500 |
| **Proposed_b_i** | 0.2927 |
| **Proposed_b_w** | 0.2718 |
| **Proposed_b_i_w** | 0.2736 |

Thus, in terms of counting our model also outperforms the Yolo model.

#### 4.1.3 Short Conclusion

For all the cases it is showed that the proposed model for all the cases, and the improvement is not quite significant, compared to the result in my previous implementation.

## 4.2 Training/Testing Loss Trend

The trend of baseline model and proposed model is showed in Fig 4 and Fig **??** respectively. It is observed that the proposed model is more stable and the loss is asymptotically decreasing compared to the baseline model.

## 4.3 Analysis

#### 4.3.1 performance: with/without bus occurrence

From the result in Section 3.1 it is observed that the proposed model significantly outperforms the baseline model. To investigate the detail of the behavior of the proposed model for 'with-Bus' and 'without-Bus' cases we compare the number of occurrence (images) that the proposed model beats or is beaten by the baseline model[4]. The result is listed in the Table below:

Table 5: Comparison: number of supporting samples , Proposed vs baseline

| # samples | with-Bus | without-Bus |
|---|---|---|
| Baseline_1 | 30 | 386 |
| Proposed_b | 55 | 629 |

Table 6: Comparison: number of supporting samples , Proposed_v2 vs baseline

| # samples | with-Bus | without-Bus |
|---|---|---|
| Baseline_1 | 31 | 370 |
| Proposed_b_i_w | 54 | 645 |

Table 7: Comparison: number of supporting samples , Proposed_b vs Proposed_b_i_w

| # samples | with-Bus | without-Bus |
|---|---|---|
| Proposed | 41 | 506 |
| Proposed_b_i_w | 44 | 509 |

[4]by comparing the absolute difference of the predicted density with ground truth

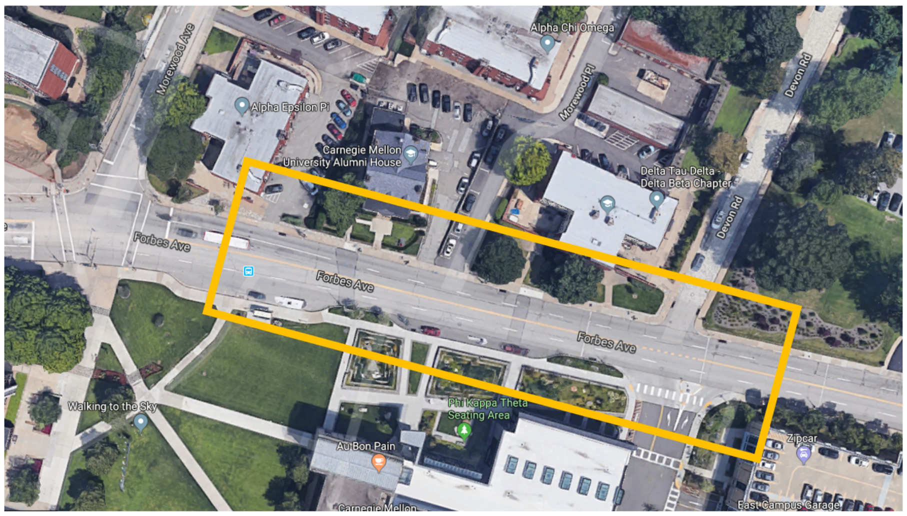
### 4.3.2 correlation analysis

- **correlation between density and bus occurrence**: at each time stamp, we analyze how the occurrence of bus (number of bus in the region) is correlated with traffic density. Thus, the coefficient correlation factor is 0.1255.
- **correlation between bus feature and bus occurrence**: at each time stamp, we analyze whether the designed bus features (extracted from BUS API) could predict the occurrence of bus (0 or 1) well. Based on this, we used **baseline_2** (temporal model with bus features only) to predict the indicator. Notice that here we are using the classification format, and the performance if evaluated using classification accuracy. Thus, the classification accuracy we got is 0.8592[5], which indicates that the bus temporal features can predict occurrence of bus well.

### 4.3.3 Potential

In order to explore how much the additional features could improve the performance of density estimation using the vision part, we first trained the network only with image input for 50 epoch. After this, we froze the weight of the convolutional part, added the bus features with LSTM layers, and fine-tuning the model using both image and bus features. The result indicated that the addition of bus features improve MAE from 0.8898 to 0.6793.

Figure 1: vision scope of the camera



[5]since the number of two condition is not balanced, a greater weight is assigned to the bus-occurrence case when doing classification.

Figure 2: Example of region segmentation: 61A, 61B, 61C, 61D, outbound

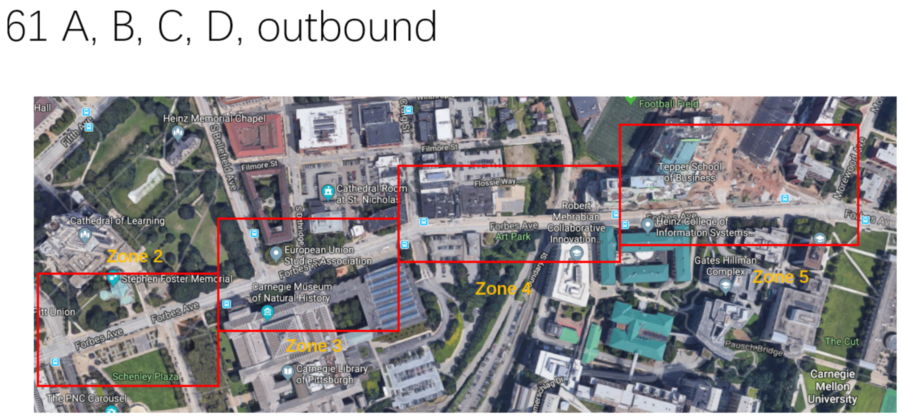

61 A, B, C, D, outbound
Zone 2
Zone 3
Zone 4
Zone 5
Forbes Ave
Cathedral of Learning
Carnegie Museum of Natural History
Carnegie Library of Pittsburgh
Tepper School of Business
Heinz College of Information Systems
Gates Hillman Complex
Carnegie Mellon University
Schenley Plaza

Figure 3: pipeline of input image processing

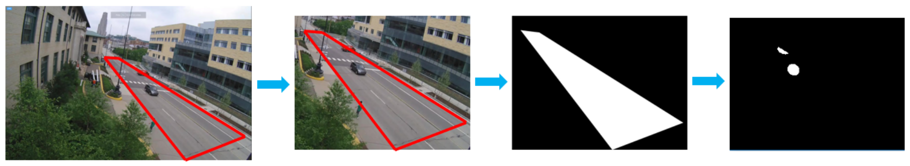


Figure 4: Loss Trend (baseline model)

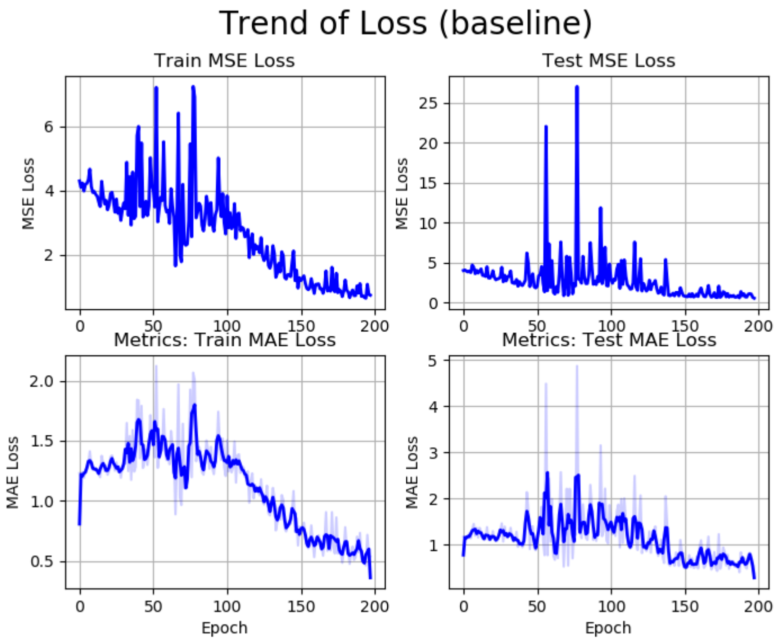


Figure 5: Loss Trend (proposed model_b)

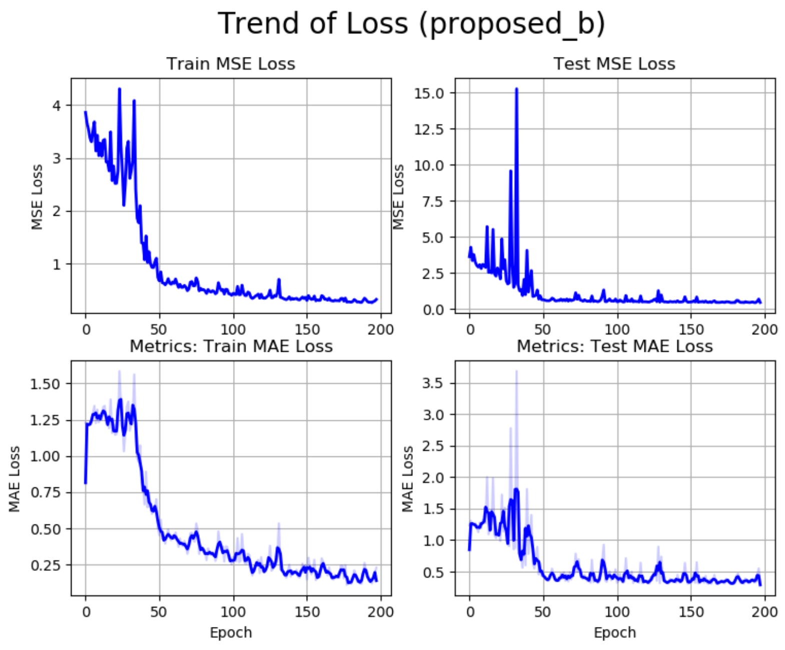
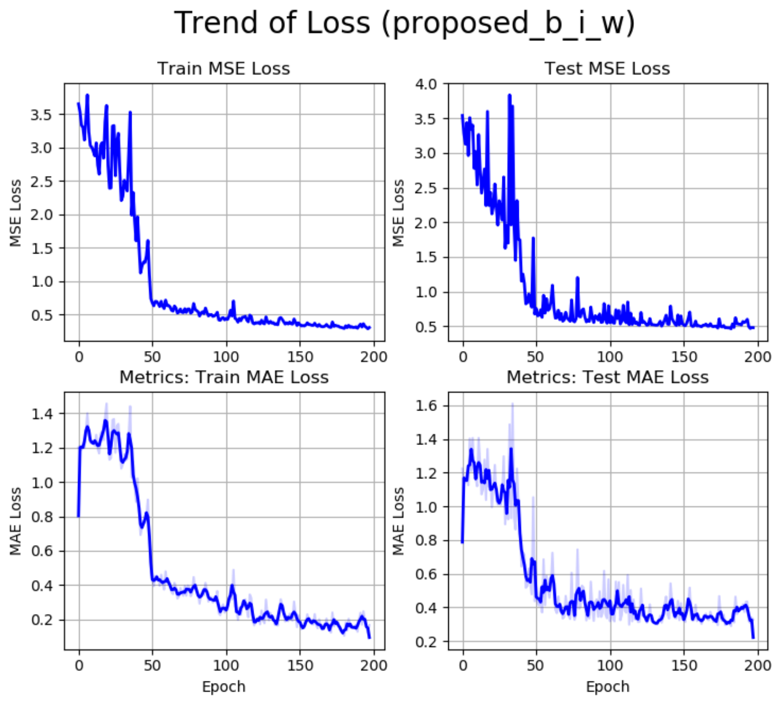


Figure 6: Loss Trend (proposed model_b_i_w)



## Appendix

Figure 7: Baseline Model (single input)

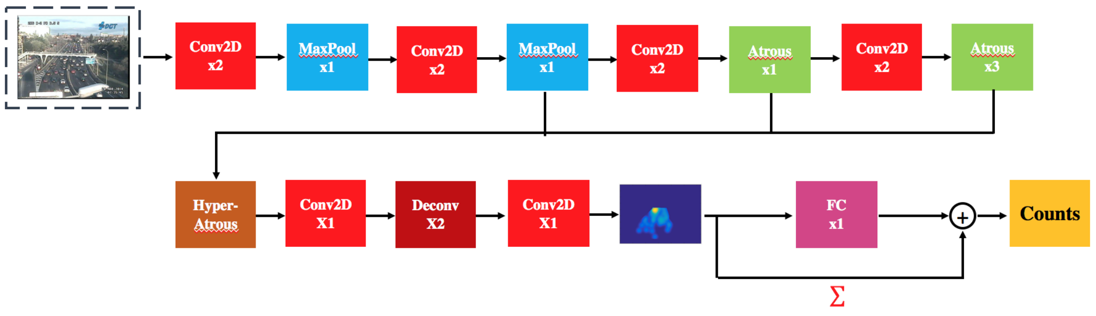


Figure 8: Proposed Model v2 (Multiple input)

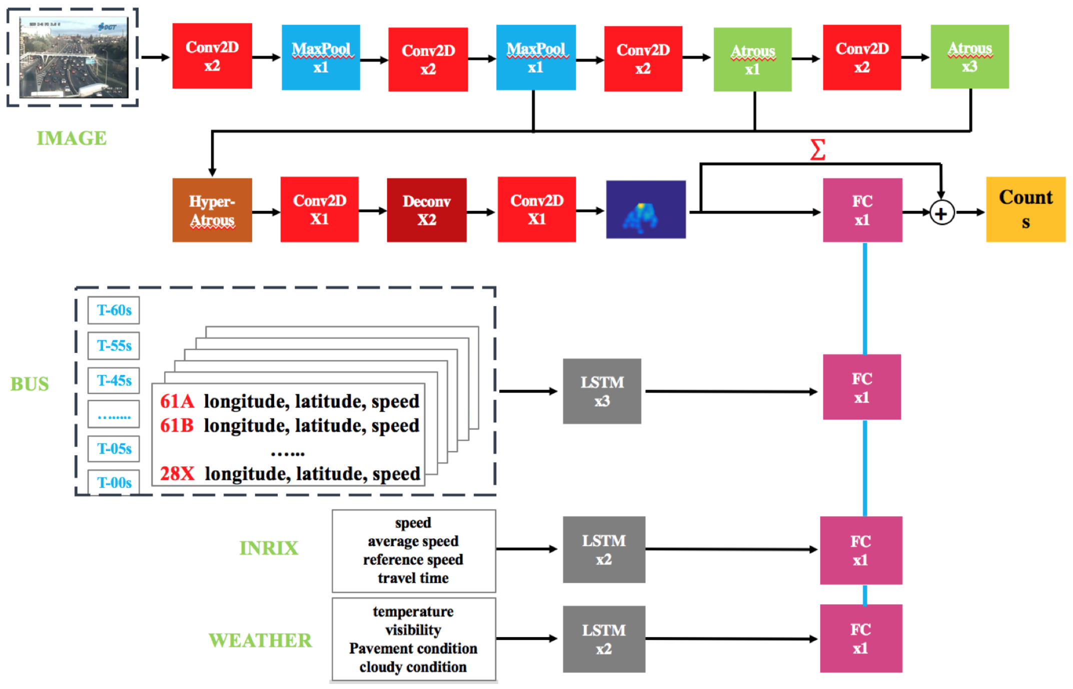


IMAGE
Conv2D x2
MaxPool x1
Conv2D x2
MaxPool x1
Conv2D x2
Atrous x1
Conv2D x2
Atrous x3
Hyper-Atrous
Conv2D X1
Deconv X2
Conv2D X1
Σ
FC x1
Count s
BUS
T-60s
T-55s
T-45s
........
T-05s
T-00s
61A longitude, latitude, speed
61B longitude, latitude, speed
......
28X longitude, latitude, speed
LSTM x3
FC x1
INRIX
speed
average speed
reference speed
travel time
LSTM x2
FC x1
WEATHER
temperature
visibility
Pavement condition
cloudy condition
LSTM x2
FC x1