Applying UNet for the traffic map prediction across different time and space : Traffic4cast challenge 2021

Sungbin Choi

sungbin.choi.1@gmail.com

Abstract. This paper describes our experiments applying UNet for the traffic map prediction on the Traffic4cast challenge 2021. Traffic map contains information regarding traffic volume and speed aggregated over one hundred square meter region. Challenge comprises two tasks. Core task is about making traffic map predictions over different time periods. Extended task is about making predictions over different time periods and space. We trained various UNet based deep convolutional neural networks having worldwide eight cities traffic map data as training set. Our method showed competitive performance in this year's challenge dataset.

1. Introduction

In Traffic4cast challenge, we need to predict future traffic map having traffic volume and speed information per each pixel, representing one hundred square meter local area. Traffic4cast challenge 2021 [1,2] is composed of two separate but related tasks. In the core competition task, we need to make a prediction over four big cities worldwide, given traffic map data of the same cities in previous year. In the extended competition task, we make predictions on two cities, but without having traffic map train data from corresponding test set cities.

We need to make predictions on each pixel of a high resolution traffic map image. UNet [3] produces output prediction having equal image dimension as input image, and is a very popular and effective method for these kinds of problem settings. We used it in our previous participation [4,5] also.

Given eight worldwide cities as train data, we trained various UNet models having differences in model structure or train set composition. Then we combined them by averaging predictions from a single model. Our method showed competitive performance in this year's challenge.

2. Methods

2.1 Basic task

All traffic images have 495 x 436 image dimensions (height and width). Each pixel represents a one hundred square meter area. It is captured per 5 minute time interval. Given a 12 timeframe traffic map image which represents one hour long traffic map of cities, we need to predict the future 6 timeframe, which represents the next 5, 10, 15, 30, 45, 60 minute timeframe traffic maps respectively.

2.2 Input preprocessing

Challenge's input data is composed of dynamic input data which varies over time, and static input data which is invariant to the time.

2.2.1 Dynamic input data

Dynamic input data is (12, 495, 436, 8) shaped tensor. First axis represents the time bin and the last axis represents the feature channel. The eight feature channel contains traffic volume and speed for four headings (northeast, northwest, southeast and southwest) respectively. Each channel is normalized and discretized to range from minimum 0 to maximum 255.

In this study, we combined the time bin axis with the feature channel, so input data is converted to (495, 436, 96) shaped tensor.

2.2.2 Static input data

Static input data is (495, 436, 9) shaped tensor. This data represents properties of the road maps, which is invariant to time change. We appended this static info to dynamic input data, so now we have (495, 436, 105) shaped input tensors combined together via the feature channel.

High resolution static data is newly provided for this year's challenge, but we haven't used it.

2.2.3 Number of train set instances per city

Each traffic image contains 5 minutes's traffic data. So there are 288 traffic images per day. Each traffic map instance has a two hour time length (1 hour for input, and 1 hour for target output to predict). By sliding a two hour (which translates to 24 continuous time frame) time window, we can get 265 train instances per day.

There are eight cities train data provided for this challenge. Four of them have 361 days traffic data and other four of them have 180 days traffic data. So cities like Berlin (having 180 days) have 47,700 training instances. Cities like Moscow (having 361 days) have 95,665 training instances.

2.3 Output sizes

Prediction output is expected to be (6, 495, 436, 8) shaped tensor data. First axis represents the six time bin (future 5, 10, 15, 30, 45, 60 minute) and the last axis represents traffic volume and speed for four headings. In the same fashion as its input, the time axis is combined with the feature channel so (495, 436, 48) tensor data is

initially produced as output from our method, then reshaped back to (6, 495, 436, 8) shape for submission.

2.4 UNet based approaches

We implemented a UNet [3] based model on Pytorch [6], having encoder and decoder with skip connections as described in Figure 1. Our experiment code is publicly available at https://www.github.com/sungbinchoi/traffic4cast2021. UNet has been widely used in various tasks including image classification and segmentation [7,8,9,10,11]. Mean squared error [15] is used as a loss function and evaluation measure. Adam optimizer [12] is used with the learning rate set to 3e-4.

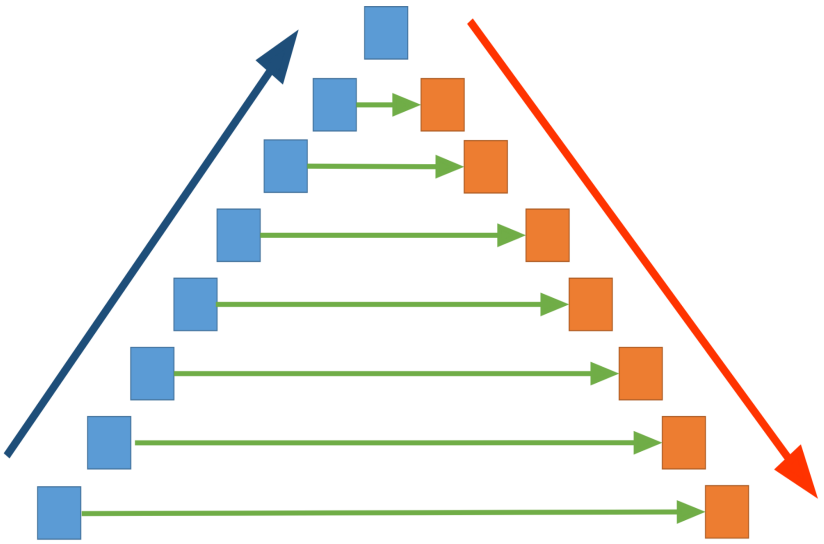

Figure 1. Overall model structure

Each blue box represents dense convolution layers block with average pooling layer. Each orange box represents deconvolution layers. Green arrow represents skip connection between downsampling path and upsamping path.

We experimented with various model structures as follows.

2.4.1 Model_A

Each downsampling convolutional block comprises two convolutional layers densely connected with each other in a feed forward fashion [13]. Between each downsampling blocks, average pooling is used to reduce image size in half.

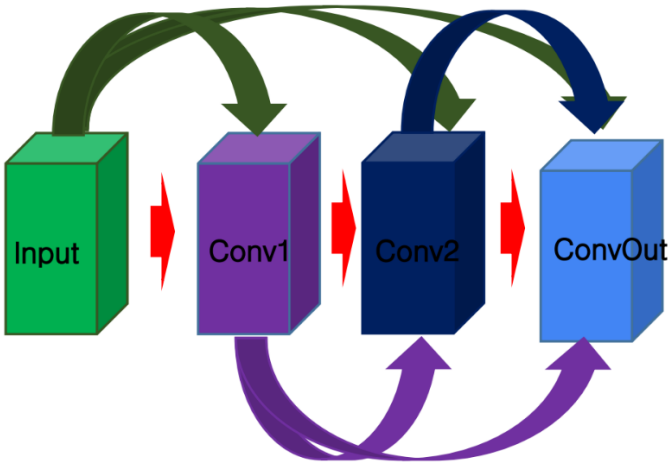

Figure 2. Model_A downblock structure

Inside the downblock of Model_A, two convolutional layers are densely connected with each other.

2.4.2 Model_B

Model_B is similar to Model_A, but linear interpolation with different scale factors is used instead of average pooling. In Model_A, image size is halved so it is similar to setting scale factor as 0.5. In Model_B, we set it to 0.7, so practically image size is reduced more slowly while going through each downsampling block.

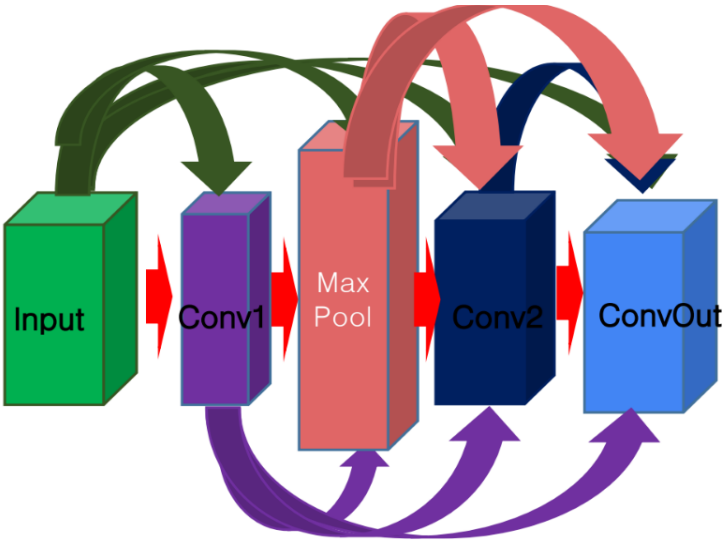

Figure 3. Model_C downblock structure

Inside the downblock of Model_C, two convolutional layers and one max pooling layer are densely connected with each other.

2.4.3 Model_C

Model_C is similar to Model_A, but a max pooling layer is added in the middle of the two convolutional layers. In Model_A, no max pooling layer was used.

2.4.4 Model_D

In Model_D, inspired from residual network [14], convolutional layers and max pooling layers having various kernel sizes are applied and outputs are added to the input.

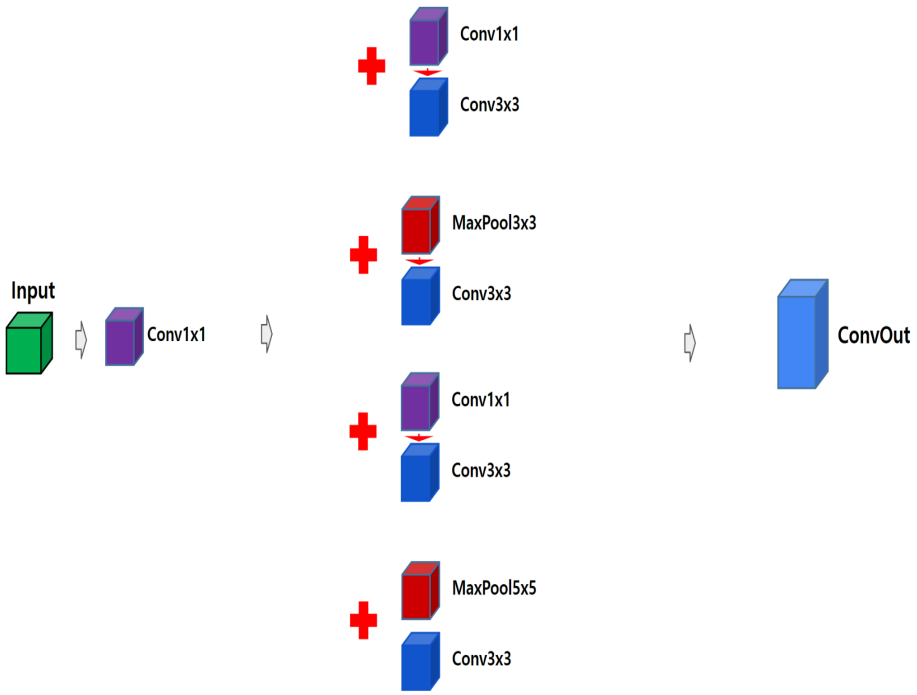

Figure 4. Model_D downblock structure

In Model_D, outputs from convolutional layers and one max pooling layer are added to the input, similar to the residual networks.

2.5 Core competition task

For the core competition task, we combined 7 individual predictions by averaging. Each of 7 predictions (named as t1m1 ~ t1m7) are made using the following method as summarized in Table 1.

Table 1. Core competition task base model runs

	Description
t1m1	Model_B used. Model is trained on only the target city train set (e.g., the model for Berlin is trained on a train set from Berlin only).
t1m2	Same as t1m1
t1m3	Model_C used Model is firstly trained in an arbitrary other city (Moscow), then later trained on a target train set.
t1m4	Model_A used Model is trained on train set having all eight cities combined
t1m5	Same as t1m4
t1m6	Model_B used. Model is trained on train set having all eight cities combined
t1m7	Model_D used Model is trained on train set having all eight cities combined

2.6 Extended competition task

For the extended competition task, we combined 4 individual predictions by averaging. Each of 4 predictions (named as t2m1 ~ t2m4) are made using the following method as summarized in Table 2.

Table 2. Extended competition task base model runs

	Description
t2m1	Model_A used Model is trained on train set having all eight cities combined
t2m2	Model_A used Model is trained on train set having all eight cities combined Train data is augmented with horizontal/vertical image flipping
t2m3	Model_C used Model is trained on train set having all eight cities combined
t2m4	Model_C used

	Model is trained on train set having all eight cities combined Different loss weight is applied per city while training

With regard to the t2m4 run, the average loss value from each train city has a different scale. For example, the average loss value from Berlin or Moscow was higher than Antwerp or Melbourne. When training models on train set different cities combined, we thought that it might be beneficial to scale the loss value from different cities. Loss weight per city were Antwerp: 0.612, Barcelona:1.000, Bangkok: 0.707, Berlin:0.311, Chicago:0.707, Istanbul:0.280, Melbourne:0.866, Moscow:0.252. These weights were set arbitrarily just observing loss values from the train set.

3. Results

Our UNet based model takes input from height 495 width 436 sized input image and incrementally downsizes it using average pooling or convolution pooling up to height 4 width 4 image at the top of the UNet (Table 3). Then it regenerates height 495 width 436 sized output image back with prediction.

Table 3. Model A output shape per each block

	Output shape
DenseBlock-1	(495, 436, 64)
AveragePooling	(248, 218, 64)
DenseBlock-2	(248, 218, 96)
AveragePooling	(124, 109, 96)
DenseBlock-3	(124, 109, 128)
…	
DenseBlock-7	(8, 7, 128)
AveragePooling	(4, 4, 128)
DenseBlock-8	(4, 4, 128)
Convolution Layer	(4, 4, 128)
DeconvolutionBlock-1	(8, 7, 128)
DeconvolutionBlock-2	(16, 14, 128)
…	
DeconvolutionBlock-7	(495, 436, 128)
Convolution Layer	(495, 436, 48)

In this year's challenge, we tried various model structure types with different train set compositions. There were no significant differences enough to say which model is better than the other.

In a core competition task, single model performance (measured by mean squared error from ground truth) ranged from 48.9 to 49.4 according to the test set evaluation. Regarding train set composition, training from a train set having other cities combined seems comparable performance with training from only target city's traffic data. Our best score was 48.49 when seven individual model predictions combined with averaging.

Table 4. Core competition task evaluation results

	MSE
t1m1	49.11
t1m2	49.16
t1m3	49.33
t1m4	48.96
t1m5	48.98
t1m6	49.36
t1m7	49.33
Combined by averaging	**48.49**

t1m1 and t1m2 were trained from a train set acquired from only the target city. Other models utilized train data acquired from other cities. In previous competitions (traffic4cast 2019 and traffic4cast 2020), we had also tried to utilize train data from other cities, but it hurt performance compared to not using them. This year, models trained on other cities data combined showed comparable performance. One possible reason for this could be the number of cities the training set has. In previous competitions, three cities' training data was provided. This year training data increased to eight cities in total.

In an extended competition task, target city's train data is not provided. So all models trained were based on a train set having other cities combined. Single model performance ranged from 59.9 to 60.2 according to the test set evaluation. Our best score was 59.55 when four individual model predictions combined with averaging.

Table 5. Extended competition task evaluation results

	MSE
t2m1	60.19
t2m2	59.94
t2m3	60.21
t2m4	59.92
Combined by averaging	**59.55**

4. Conclusion

This is our third participation in the Traffic4cast challenge. Last year three cities traffic map data was used for competition train and evaluation. This year it is increased to 8 cities in core task and 2 more cities used for extended task evaluation, which can make more robust evaluation possible. UNet models seem to be effective in both tasks. This year we tried various UNet models and combined them to achieve more accurate predictions.

5. References

1. https://www.iarai.ac.at/traffic4cast/
2. Kreil, David P., et al. "The surprising efficiency of framing geo-spatial time series forecasting as a video prediction task–Insights from the IARAI Traffic4cast Competition at NeurIPS 2019." NeurIPS 2019 Competition and Demonstration Track. PMLR, 2020.
3. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.
4. Choi, Sungbin. "Traffic map prediction using UNet based deep convolutional neural network." arXiv preprint arXiv:1912.05288 (2019).
5. Choi, Sungbin. "Utilizing UNet for the future traffic map prediction task Traffic4cast challenge 2020." arXiv preprint arXiv:2012.00125 (2020).
6. Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." Advances in neural information processing systems 32 (2019): 8026-8037.
7. Li, Xiaomeng, et al. "H-DenseUNet: hybrid densely connected UNet for liver and tumor segmentation from CT volumes." IEEE transactions on medical imaging 37.12 (2018): 2663-2674.
8. Weng, Yu, et al. "Nas-unet: Neural architecture search for medical image segmentation." IEEE Access 7 (2019): 44247-44257.
9. Chen, Wei, et al. "S3D-UNet: separable 3D U-Net for brain tumor segmentation." International MICCAI Brainlesion Workshop. Springer, Cham, 2018.
10. Zhou, Yongjin, et al. "D-UNet: a dimension-fusion U shape network for chronic stroke lesion segmentation." IEEE/ACM transactions on computational biology and bioinformatics (2019).
11. Qamar, Saqib, et al. "A variant form of 3D-UNet for infant brain segmentation." Future Generation Computer Systems 108 (2020): 613-623.
12. Kingma, D. P., & Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014.
13. Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
14. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
15. Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1, no. 10. New York: Springer series in statistics, 2001.