Traffic States Map Prediction

Zhijie Wang
Department of Electrical and Computer Engineering
University of Waterloo
zhijie.wang@uwaterloo.ca

Abstract

In this project, I will give my solutions to Traffic4cast 2020, which contains a task of predicting traffic states of several cities given previous states information. The task is basically a time-series prediction problem, however, different from traditional ones, the traffic states were described as 2D images, which means that the outputs and inputs are both images sequence. From the result of Traffic4cast 2019, some Computer Vision methods have been proved useful toward this task. Since this year's competition has new datasets and requirements, we proposed a new network structure based on U-Net to combine and fuse static and dynamic spatio-temporal information.

1 Introduction

Traffic4cast 2020 is a competition track of NeurIPS 2020 hosted by Institute of Advanced Research in Artificial Intelligence (IARAI). The competition's task is to predict short-term large-scale states, which include volume and speed of specific city at a time (See Fig. 1). Real-time traffic states prediction is an essential task for urban road management and control. A better understanding of current traffic states and prediction of future ones will significantly help the control of road systems, reduce the happenings of traffic congestion, and improve road safety. The traffic states prediction problem is a time-series prediction problem, and representative solutions include Recurrent Neural Networks (RNNs), Long-short Term Network (LSTM). However, in this competition, we're going to predict traffic states for a whole city instead of a specific place. Therefore, the problem is more complicated since we also need to consider the spatial information in the map, that religions (points on the map) which close to each other might share similar road and traffic conditions. And these features make it possible of using Convolution Neural Networks (CNNs) which are designed for computer vision of image processing problems.

From the top solutions of Traffic4cast 2019 in the last year, U-Net based methods[1, 2] show great performance on such problems, whereas this year's competition requires more complicated 8 channels' prediction targets (4 directions' speed and volume) instead of 3 channels (heading, volume and speed), and the target time stamps vary from 5 minutes to 60 minutes in the future instead of 5 to 15. Moreover, this year's dataset also provides static traffic map, which includes features like junction cardinalities and facilities.

Therefore, the main challenge of this competition is that requiring the model having the ability to extract features from both space domain and time domain from past, and also from some static information, then to predict the future traffic states of a whole map.

In the following, we first review literature and solutions in Traffic4cast 2019. Next, we present details about our designed network architecture, followed by experiments and evaluation results.

Submitted to Introduction to Machine Learning (CS480/680, 2020 Spring). Do not distribute.

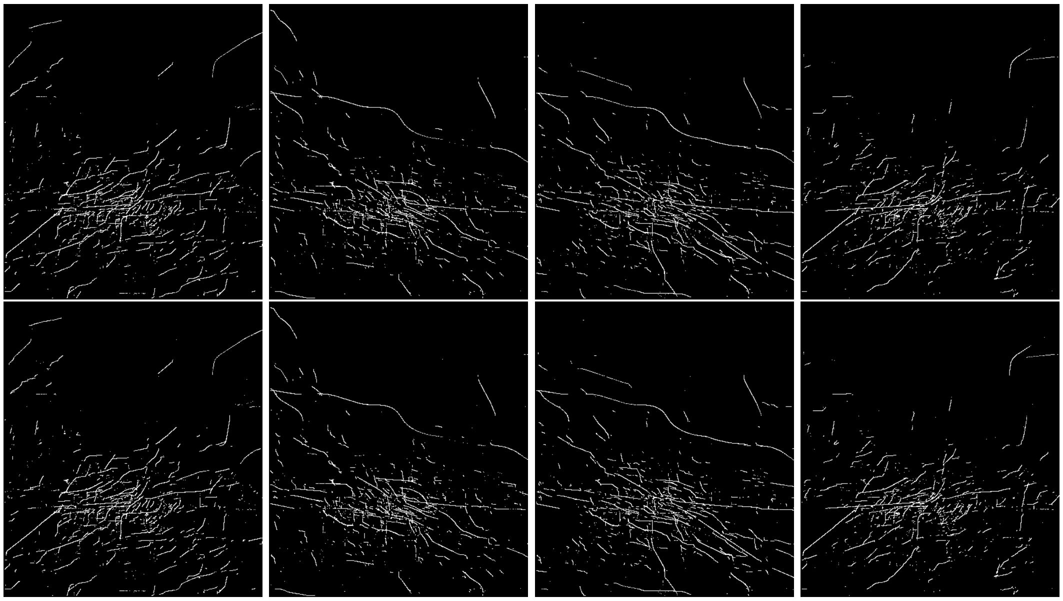


Figure 1: Sample datasets of Berlin from Traffic4cast 2020. The first row are speeds of 4 directions: NE, NW, SE, SW. The second row are volumes of 4 directions, respectively.

2 Related Works

From 6 best solutions[1–6] of Traffic4cast 2019, we can find that mainly two kinds of solutions have the best performance. In [1–4], the authors all used U-Net[7] based CNNs' structure, and in [5, 6], the authors used RNN/LSTM based methods.

LSTM[8] is a typical using of RNNs, which has better performance on time-series prediction with its feedback connections. However, LSTM focus on modeling long-term dependency in time-series, and might fail to capture temporal pattern during short-term, hence, TPA-LSTM[9] (Temporal Pattern Attention LSTM) was proposed to improve this. Different from the traditional time-series problem, this competition requires predicting image sequences, which is similar to movie prediction problem, hence, methods[10, 11] combining CNNs and LSTM were designed for such problem. In [5], the author used a 2D convolutional layer to process the input before feeding into LSTM, which is designed to extract the temporal and spatial features. In one of the best solutions of Traffic4cast 2019, [6] used an RNN to predict the future states. To reduce the memory usage and computational cost, the author used a two-way autoencoder to encode and decode the input and output.

The input data of Traffic4cast competition has a similar structure as images (height * wight * channel), where channel here represents volume, speed or heading. Hence, methods[12] used in real-movie predictions might be useful in such problem. However, though traffic states images have both spatial and temporal coherence, they have less temporal correlations which make it more difficult to use generate methods, whereas methods like U-Net in [1–4], which might still achieve a good performance in this competition although they didn't pay much attention to temporal relationships. U-Net is a network structure containing an equal number of down-sampling and up-sampling blocks, which is designed for medical image segmentation. In this competition, the output image only has values on roads, which looks similar to a segmentation map. In[1, 2], the authors both used the U-Net structure directly. The only difference is that [2] used a fully convolutional network instead of the normal 2D convolution. In[4], the authors cropped the original input and output into 4 sub-regions, therefore, trained 4 models instead of 1 for each city. Besides, the author also designed a new loss function, which calculated cross-entropy loss on heading and mean absolute error on volume and speed channel. This modification on loss function is reasonable since only 4 values of headings existed. (However, it might not be useful in this year's task). [3] also designed a new loss function, which contains a domain-transformed $L2$ loss to represent loss on temporal and spatial.

Recently, some implementations[12, 13] based on U-Net structure which designed for special tasks like road segmentation were also meaningful and might be useful in this competition's tasks. In [13],

Table 1: Datasets format and introduction

	Dynamic data	Static Data
Size	$(288, 495, 436, 9)$	$(495, 436, 7)$
Channel 0	Speed NE	Junction cardinality
Channel 1	Heading NE	Road classes
Channel 2	Speed NW	eat, drink and entertainment
Channel 3	Heading NW	hospital
Channel 4	Speed SE	parking
Channel 5	Heading SE	shopping
Channel 6	Speed SW	transport
Channel 7	Heading SW	
Channel 8	Incident level	

the authors add residual units to U-Net to facilitate training and address the degradation problem, while in [12], the authors add a res-block between convolutions layers of each U-Net's layer. Both these two methods showed that they have better performances than direct usage of U-Net on road segmentation problems.

However, all the methods above still have some remained problems, for CNN based methods, how to extract the temporal features better still need to be concerned, and for RNN based methods, the hardness of training RNN is always a big issue. Besides, the new task requirements and new data in this year's competition also propose some new challenges. The input and output images were increased from 3 channels into 8 channels, and the predictions require from 15 minutes in future into 1 hour in the future. Both the new tasks and he usage of newly provided static information lead to a demand for a new network structure.

3 Fusion U-Net

In this project We designed a new network structure based on U-Net structure, which can fuse the spatial-temporal dynamic information and the spatial static information. Instead of stacking the static map into the input of U-Net, we used a two channels' network, which processes the dynamic and static information individually, then fusing them into the output images. In the following subsections we will introduce the input representation and network architecture.

3.1 Input Representation

As we can see from Table 1, the datasets contain two types - dynamic and static. The dynamic datasets have a tensor size of (t, w, h, c), t represents the timestamps, $(w * h)$ is the size of the map, and $c = 9$ represents 9 different channels of different kinds of values, respectively. The static datasets is a tensor of size (w, h, c), and $c = 7$ represents 7 different channels of static information. Since the static information won't change with time, the static datasets only have 3 dimensions.

The competition task requires using the past hour's data to predict 6 specific timestamps' traffic states (8 channels) in the next hour. Therefore, to maintain each input sample as a 3D tensor, we flattened the input tensor on t and c, and stack the past hour's dynamic data (12 timestamps) and static data, therefore, each input will be $(115, 495, 436)$, and the output will be $(48, 495, 436)$.

3.2 Network Architecture

Fig. 2 shows my proposed network architecture. The input has a size of $(115, 495, 436)$ as we have discussed in the above subsection, to avoid non-integer size in down-sampling, the input size was padded into $(115, 496, 448)$.

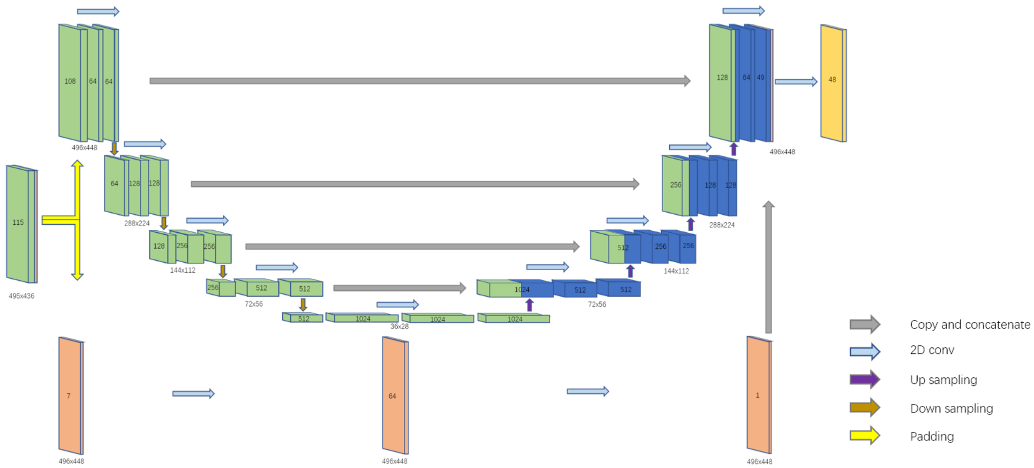


Figure 2: Proposed network architecture. The number on each tensor represents the number of channels, the number below the tensor represents its image size (width $\times$ height)

3.2.1 U-Net for processing dynamic information

After padding, the network can be split into two parts. The upper one is a U-Net that has 5 layers on depth, each layer has two 2D convolutional connections. The input was first down-sampled into $(36, 28)$, then up-sampled back into $(496, 448)$. One important setting of U-Net is the skip connection between down-sampling and up-sampling blocks, which can keep the features extracted from different scale, high-resolution images can help extract local features whereas low-resolution images can help extract global features. In this project, we want to predict traffic states according to a continuous sequence of past timestamps, global features mainly refer to temporal features and local ones refer to spatial features, respectively.

Fig. 3 shows the procedure of down-sampling and up-sampling, as we can see, the down-sampling could keep the roads have heaviest traffic, and the up-sampling procedure will add more details on it.

3.2.2 CNN for processing static information

The lower part is a 3-layer CNN, which is used for processing static information. Here we use this simple CNN structure to extract features from these static information, in particular, we would like to reduce the 7 channels' input into a "mask". These static information don't have obvious connections with the traffic states, however, some hidden connections might exist. Therefore, these hidden connections could refine the results obtained from U-Net. Fig. 4 shows an example of this static information mask, brighter the area, more possible the area would have a heavy traffic given the information like junction cardinality, road classes, facilities, etc. (See Table 1 for detail instruction).

3.2.3 2D Conv for fusing dynamic and static information

The final output is 6 timestamp's traffic states, which is already obtained from U-Net. Since we also have obtained the static information mask from CNN, we simply fusing these two kinds of information by a 2D convolutional layer.

4 Experiments

Since the competition is still ongoing, here we evaluated our method on the validation datasets provided by Traffic4cast 2020. In the following subsections, we first introduce the implementation details and training scheme, then conduct and discuss about the evaluation.

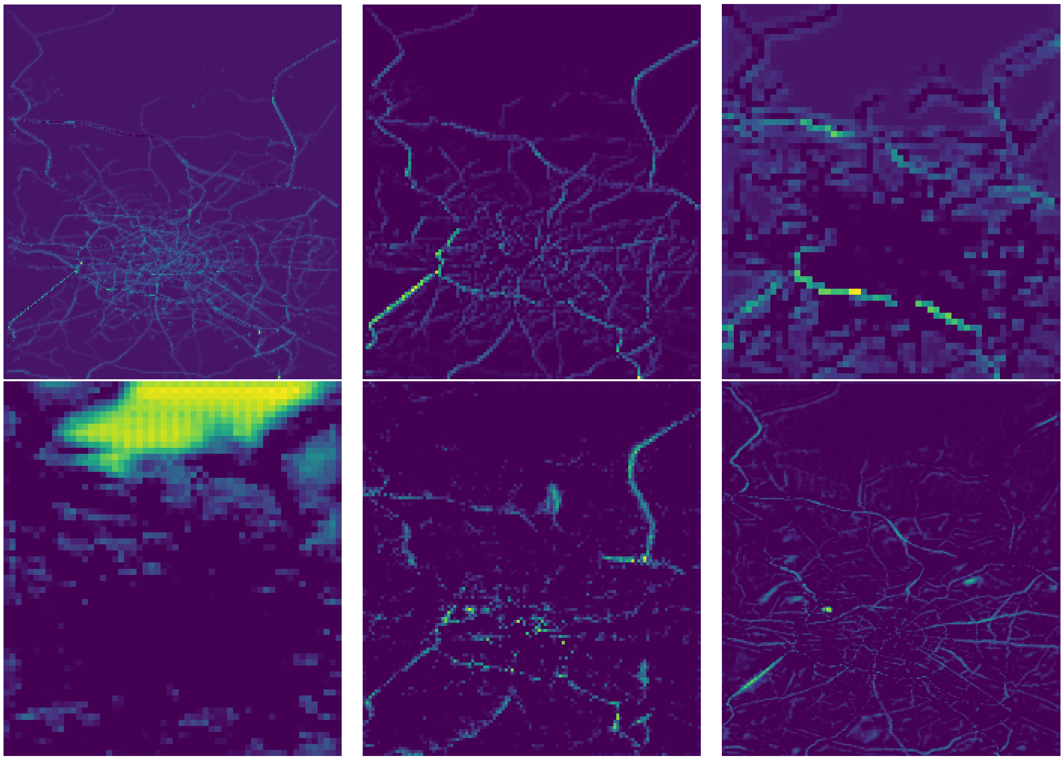


Figure 3: From left to right, the first row shows the procedure of down-sampling; the second row shows the procedure of up-sampling and skip connections

Figure 4: Static Information Mask

4.1 Implementation details

The network used in this project was implemented with PyTorch 1.6. The provided training data includes 181 days (half year)'s traffic states, each day has 288 records. We first split the datasets through a "sliding window" method (See Fig. 5) into features and targets. Therefore, for each day we have 265 samples, 47965 training samples in total. To handle the computational cost on a normal GPU (NVIDIA GTX 1660 Super), here we randomly select 10000 samples for training.

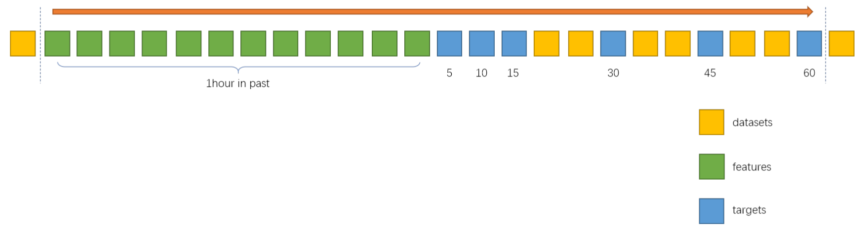


Figure 5: Creating datasets

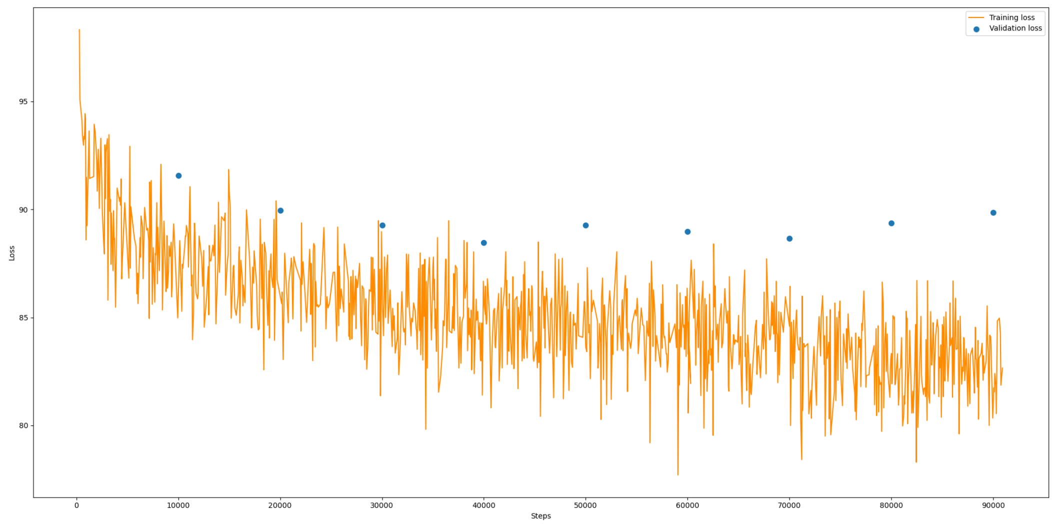


Figure 6: Training and validation loss vs steps

4.2 Training scheme

Since the evaluation metric of Traffic4cast 2020 is MSE (Mean Squared Erorr), we also selected MSE as our loss function, and utilize Adam optimizer for training. Fig. 6 shows the training progress, after 50000 steps (5 epochs), the training loss became stable, and after 7 epochs, the validation loss started increasing. Here we also used momentum in the training scheme, after 5 epochs, the learning rate decreased into $10^{-}3$.

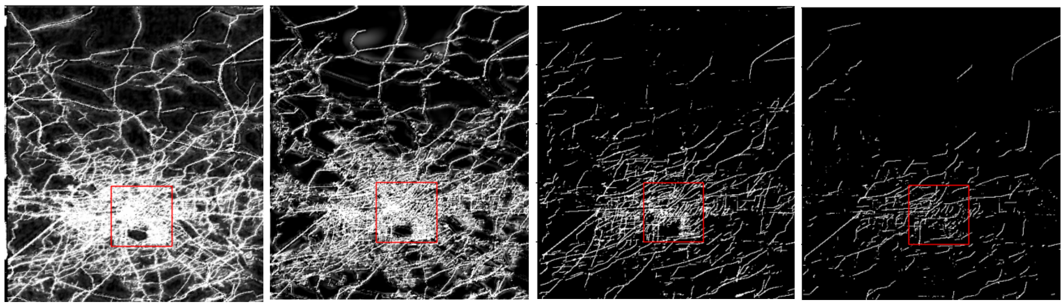
4.3 Evaluation

To better showing our network have a good performance on this task, we conducted comparison experiments based on validation results comparing to U-Net[2], Res U-Net[13]. The validation set contain 18 days data, and each day has 265 samples like training set. We used MSE as the evaluation metric in comparison. Table 2 shows the results of three methods, and our method have the lowest MSE among 3 methods. Fig. 7 shows prediction results on NE Speed of 3 methods on one validation sample, we can see that our method have s clear segmentation of roads comparing to the other two methods, which look blur especially in red rectangle area, these were due to the fact that the other two methods might have wrong predictions on some areas without roads, and which should actually be 0 value. This also gave us some guidelines on future work, if we can make an accurate segmentation on roads before predicting, we can significantly avoid many invalid predictions and reduce the usages of computation resources.

Fig. 8 shows the average MSEs according to different timestamps, as we can see, the nearest 5 minutes have the lowest MSE, because traffic states might only have slight change during this period. Fig. 8 also shows that our network's prediction results were still reliable even in predicting traffic states in 1-hour later, which didn't have a sharp increase on MSE.

Table 2: Comparison results

	U-Net	Res U-Net	Ours
MSE	93.48	91.42	**88.66**



U-Net Res U-Net Ours ground-truth

Figure 7: From left to right, predictions on NE Speed of 1 validation sample of 3 methods and ground-truth are shown.

4.4 Run-time issue

In our computing environments, training 1 epoch (10k sequences) took about 1.5 hours, which is almost the same as directly using U-Net (without fusion parts), it showed that our network will not increase the computation load. In testing, making one prediction took about 0.91 seconds on average, which is acceptable on real-time performance since it's less than 1 minute.

5 Conclusion

In this project, we proposed our solutions to Traffic4cast 2020. Different from direct usage of U-Net as the top solutions in Traffic4cast 2019, we utilized the static information which was newly

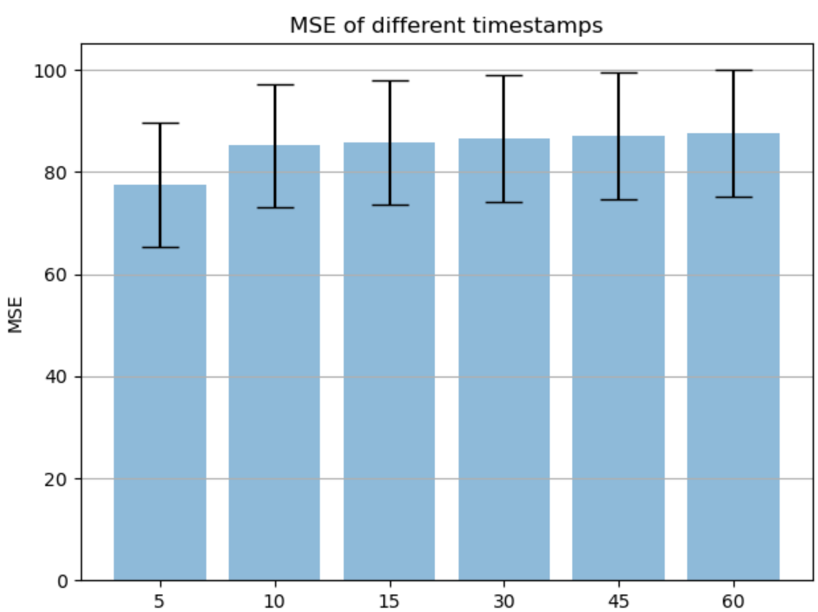


Figure 8: MSE of predictions of different timestamps.

provided in this year's competition, and designed a new structure, which processes dynamic and static information individually before fusing to fit new challenges that the training set size was reduced to half and need to predict a longer time's traffic states. From the experiments' results, we showed that our new network structure has a better performance than normal U-Net (best solutions in Traffic4cast 2019), and was also reliable when predicting a longer time's traffic states. For future work, we believe a more accurate segmentation on roads before prediction will improve the results.

References

[1] Dominik Bucher Christian Rupprecht Rene Buffat Henry Martin, Ye Hong. Traffic4cast-traffic map movie forecasting, 2019.

[2] Sungbin Choi. Traffic map prediction using unet based deep convolutional neural network, 2019.

[3] Pedro Herruzo and Josep L. Larriba-Pey. Recurrent autoencoder with skip connections and exogenous variables for traffic forecasting, 2019.

[4] Yang Liu, Fanyou Wu, Baosheng Yu, Zhiyuan Liu, and Jieping Ye. Building effective large-scale traffic state prediction system: Traffic4cast challenge solution, 2019.

[5] Tu Nguyen. Spatiotemporal tile-based attention-guided lstms for traffic video prediction, 2019.

[6] Wei Yu, Yichao Lu, Steve Easterbrook, and Sanja Fidler. Crevnet: Conditionally reversible video prediction, 2019.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[8] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[9] Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8-9):1421–1441, 2019.

[10] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[11] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104, 2018.

[12] Aidan Clark, Jeff Donahue, and Karen Simonyan. Efficient video generation on complex datasets. *arXiv preprint arXiv:1907.06571*, 2019.

[13] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

[14] Noel Cressie. *Statistics for spatial data*. John Wiley & Sons, 2015.

[15] Foivos I Diakogiannis, François Waldner, Peter Caccetta, and Chen Wu. Resunet-a: a deep learning framework for semantic segmentation of remotely sensed data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 162:94–114, 2020.