Proceedings of Machine Learning Research 123:47–55, 2020 NeurIPS 2019 Competition and Demonstration Track

Recurrent Autoencoder with Skip Connections and Exogenous Variables for Traffic Forecasting

Pedro Herruzo[*] PHERRUZO@AC.UPC.EDU
Josep L. Larriba-Pey LARRI@AC.UPC.EDU
Dept. of Computer Architecture, Polytechnic University of Catalonia
North Campus, C6 building, 08034 Barcelona, Spain

Editors: Hugo Jair Escalante and Raia Hadsel

Abstract

The increasing complexity of mobility plus the growing population in cities, together with the importance of privacy when sharing data from vehicles or any device, makes traffic forecasting that uses data from infrastructure and citizens an open and challenging task. In this paper, we introduce a novel approach to deal with predictions of volume, speed and main traffic direction, in a new aggregated way of traffic data presented as videos. Our approach leverages the continuity in a sequence of frames, learning to embed them into a low dimensional space with an encoder and making predictions there using recurrent layers, ensuring good performance through an embedded loss, and then, recovering back spatial dimensions with a decoder using a second loss at a pixel level. Exogenous variables like weather, time and calendar are also added in the model. Furthermore, we introduce a novel sampling approach for sequences that ensures diversity when creating batches, running in parallel to the optimization process.

Keywords: traffic forecasting, video prediction, recurrent autoencoder, exogenous variables, sequences' sampling

1. Introduction

Traffic forecasting deals with both, the problem of regression and classification of road states taking into account spatio-temporal features. Road states refers to traffic variables (Respati et al., 2018) *volume* (number of cars in this location), *speed* (average speed of cars at a specific location or area), and *direction* (the angle from 0 to 359 along which vehicles move). This problem, consisting of assigning future road states at each location, is particularly difficult mainly due to (i) the complexity and dynamic property of the traffic environment in cities (Liao et al., 2018), (ii) high resolution of the data (iii) lack of up-to-date road maps and data in some locations, (iv) external impacts of unknown agents to traffic like weather or soccer games, and (v) errors in data collection from sensors.

The current complexity of mobility, the growing population in cities and the increasing traffic-data collection (Respati et al., 2018) (Loop detectors, Bluetooth Mac Scanners, Mobile Phones or Connected Cars), calls for more powerful models that allow better city planning and calculating more precisely travel times. In this regard, Traffic4cast Challenge 2019 at NeurIPS (Dr Sepp Hochreiter, 2019) proposes a new representation of traffic

[*] Work performed at SEAT, S.A.

© 2020 P. Herruzo & J.L. Larriba-Pey.

data that deliberately ignores the underlying road network, mapping GPS trajectories to spatio-temporal cells that do not depend on the lack of up-to-date road maps, which empowers the development of methods that can produce high-resolution traffic states even for fast-evolving cities, like emerging economies.

Traffic forecasting is traditionally (Hong, 2011) treated as a time series problem faced with methodologies like Arima, Sarima or Support Vector Regression while more recent works use also Neural networks (Huang et al., 2014). However, these methods take into account the state of streets independently while there is a strong correlation between streets in real life. After the great success of Convolutional Neural Networks (CNN) in Computer Vision with images, or with tasks related to Signal Processing or Natural Language, the work in (Sun et al., 2017) translates traffic data into images and uses for the first time a CNN to tackle this problem. We take inspiration from the work Hierarchical Long-term Video Prediction without Supervision (Wichers et al., 2018), where the prediction of a future frame is done using an autoencoder, with predictions occurring in the embedded space and then gradually recovering back the original dimension with the decoder, all minimizing both the output of the next frame and the prediction in the embedding space.

Our work presents a novel method to tackle the problem of traffic forecasting as a scene completion task along time using the data provided by the Institute of Advanced Research in Artificial Intelligence (IARAI) (Dr Sepp Hochreiter, 2019) together with Here Technologies in the NeurIPS Traffic4cast challenge aforementioned. The proposed method is based on (Wichers et al., 2018), using their proposed loss function but reshaping the network as a sequence to sequence problem and using a U-Net (Respati et al., 2018) like architecture together with exogenous data like time of the day, day of the week and weather, benefiting from both the prediction in a lower dimensional space and the output of several future frames within a single inference. We also present a data sampling technique that allows for faster training and batch diversity, when the input is a continuous sequence that we need to break in batches for training. Furthermore, we propose a multitask loss function to minimize, taking into account that the regression for the heading channel for these data is indeed a classification problem.

The rest of the paper is organized as follows. In Section 2, we describe the problem to be solved and the data. Then, in section 3 we present the proposed methods for data sampling and frames prediction. Section 4 discusses the experimental results and section 5 concludes the manuscript with remarks and future work.

2. Problem definition

Traffic4cast presents the problem of traffic forecasting as a scene completion task along time for one year in three different cities: Berlin, Istanbul and Moscow. In particular, trajectories of raw GPS positions for each city are projected to an image containing the city with shape height = 495, width = 436, and channels = 3 (*volume*, *speed* and *heading* in this order). Each pixel in the image contains the aggregated information for a square region of $100m \times 100m$ over a time bin of 5 minutes. As a day is composed of 288 time bins of 5 minutes, for each city and day we can represent the data as a tensor $T_{city}^{day}[t, h, w, c]$ with shape $(288, 495, 436, 3)$, with channel domains $volume, speed \in \{0, 1, 2, ..., 255\} \subset \mathbb{N}$, and $heading \in \{0, 1, 85, 170, 255\}$. The aggregation of *volume* is the counting of vehicles in the interval and region (t, h, w), capped at a minimum and maximum level to remove noisy readings. Then, capped numbers are mapped proportionally to the interval $[1, 255]$ and rounded to the nearest integer, where the value 0 means no data available at this time bin. *speed* is similarly calculated, only differing in the aggregation being the average instead of counting, and only capping the maximum speed and normalizing to the interval $[0, 255]$. Here $speed = 0$ means the vehicles are not moving, if *volume* in this location is greater than zero. The computation of *heading* is different, each probe point records the heading direction in degrees (from 0 to 359), which is binned in four heading directions; North-East (from 0 to 90, represented as $heading = 85$), South-East (from 90 to 180, as $heading = 255$), South-West (from 180 to 270, as $heading = 170$), and North-West (from 270 to 359, as $heading = 1$). The selected value is the bin with the highest number of points, with the possibility to assign 0 when all directions in this region have the same number of points, being impossible to determine which is maximum. Note that there is no data if and only if $volume = 0$.

For each city Traffic4cast dataset provides 285 days for training, 7 for validation and 72 for testing. The first two have information for each time bin, but the test set only contains information in 5 blocks of 12 bins (1 hour of information each block). The goal of the challenge is to predict 3 time bins after each of these 5 blocks. In other words, we need to predict the traffic for 5, 10 and 15 minutes ahead, given the information about the previous hour, a total of 5 times per day. In particular, the 5 blocks of 15 minutes to predict start in Istanbul and Moscow at time bins $(57, 114, 174, 222, 258)$, which correspond in hours to (04:45h, 09:30h, 14:30h, 18:30h, 21:30h) respectively. In Berlin, time bins are $(30, 69, 126, 186, 234)$ that correspond to (02:30h, 05:45h, 10:30h, 15:30h, 19:30h).

Then, the problem can be formulated as follows. Given a *city*, find a function f such that:

$$f = \min_{\widetilde{f} \in \Theta} L\left(\widetilde{f}\left(T^{day}_{city}[s-q : s, h, w, c]\right),\ T^{day}_{city}[s : s+3, h, w, c]\right), \forall\ day, s, h, w, c \tag{1}$$

where $s-q \geq 0$, $s+3 \leq 288$, $q \in \{1, 2, ..., 12\}$ is the length of the input sequence, $L(\cdot, \cdot)$ is a *loss function* that measures the error between the ground truth and the prediction, and Θ is the parameter space. Note that time intervals $s-q : s$, and $s : s+3$, in eq. 1 are left-closed and right-opened, meaning that the value in the right side of the interval is never taken. Note also that if we aim to learn to predict other value than 3 future frames, we only need to change this in the formulation.

3. Methodology

In this section, we present our twofold contribution; the sampling strategies when framing a problem as a sequence to sequence, and our proposed models.

3.1. Sampling strategies for sequences

We define three sampling strategies that depend on the desired sequence input length $q \in \{1, 2, ..., 12\}$, while the output length is fixed to 3 frames. The first strategy (*non overlapping*), consists in divide each day into $T_q = ceil(288/(q + 3))$ number of sequences without overlapping, where *ceil* returns the smallest integer value greater than or equal to its input. For instance, with $q = 3$, we can divide a day into $T_3 = 48$ sequences of length 6, 3 frames for training and 3 for testing. The second strategy (*sliding window*), makes use of every possible sequence of length q+3 starting from frame 0, then starting from frame 1, and so on until starting the last sequence from frame 288−(q+3−1). Note that with the example of $q = 3$, method *non overlapping* produces 285 days × 48 sequences = 13680 sequences to train per city, whereas method *sliding window* produces $285 \times (288 - (3 + 3 - 1)) = 80655$ sequences, which is almost 6 times more sequences to train. The third strategy (*like-test*), only trains sequences with the output time bins being the same as in the test set. Hence, in this procedure, we only have $285 \times 5 = 1425$ sequences to train.

Each different strategy defines a set of time bins to train for each day in the traffic movie. Note that selecting a day and a time bin we obtain a frame, hence, the input sequence is created slicing the video of a day from the initial time bin specified up to 3 consecutive frames. In order to impose diversity when creating each training batch, our dataset is defined as a list of pairs (*day*, *time bin*), that are shuffled at the beginning of each epoch, and allow us to create batches that contain sequences of different days and time bins together. This should lead to faster convergence since batches are always different after each epoch, as explained by Yoshua Bengio in (Bengio, 2012). Furthermore, we perform all batch preparation and preprocessing in parallel to the optimization, which makes our training diverse and efficient. In preprocessing, we cast data to float numbers and normalize all values into the interval [0, 1].

3.2. Proposed models

Our model builds upon the architecture proposed by Nevan Wichers et al. in (Wichers et al., 2018), where the authors presented a neural network that can predict the next frame (F_{t+1}) in a movie given the previous one (F_t). Given F_t, an encoder followed by a recurrent layer predicts the embedding of the future frame, which is compared with F_{t+1} applied to the same encoder using an L2 loss. Then, the embedding is upsampled to the original space by a decoder, which is compared to F_{t+1} also with an L2 loss, using skip connections between different layers in the encoder-decoder to refine the output.

Given that the nature of our problem is a sequence to sequence prediction, we adapted the aforementioned architecture to take profit of the entire input sequence of length q X_q, when predicting the three future frames Y_3. To do so, we accept a sequence of any size as input by iteratively using the encoder and concatenating its outputs. Then a recurrent encoder accumulates the temporal information of the input sequence into a single representation, and a recurrent decoder gives us three embedded predictions $\widetilde{e_3}$. Afterward, these predictions are upsampled to the original space ($\widetilde{Y_3}$) by a decoder that uses skip connections from each layer of the encoder, but only from the last frame of the sequence including the input video. As in the original model, we train both the predictions in the embedding and in the original space using an L2 loss with weights $\alpha, \beta \in [0, 1]$:

$$L = \alpha L2(Y_3, \widetilde{Y_3}) + \beta L2(e_3, \widetilde{e_3}) \tag{2}$$

Figure 1 illustrates our architecture. As can be seen, we further introduce exogenous data to our model. In particular, for each frame in the sequence, we use its time of day, day of the week, current weather and its prediction in the next 3 time bins, using data downloaded from the website *World Weather Online* (Weather, 2019). In our implementation, the encoder is composed of 6 blocks of i) two sub-blocks of (convolution, Batch normalization, and ReLu), ii) Max pooling, and iii) Dropout=0.5, with number of convolutions [16, 16x2, 16x4, 16x8, 16x8, 16x2] in this order for each block, and downsampling the image from 512 to 16 by powers of 2. The decoder is composed of 6 blocks of i) Transposed Convolutions (Dumoulin and Visin, 2016), ii) Concatenation from its sibling layer in the encoder iii) Dropout=0.5, and iv) same two sub-blocks than the encoder, with number of convolutions [16x8, 16x8, 16x8, 16x4, 16x2, 16x1] consecutively. The recurrent encoder-decoder uses layers of GRU with (2048, 256, 128) and (128, 256, 2048) units respectively. Frames are upsampled using bilinear interpolation in the first place in the model to 512×512, and cropping plus a convolution of size 3 with ReLu are used at the end of the model to match the original size 495×436.

In the rest of the paper, we will call the proposed model: Recurrent Autoencoder All (*RAE_all*), which includes skip connections (including the input), weather and time information. We also tried different versions of this model: i) one without using the input of the last sequence in the skip connections for the decoder (*RAE_not_In*), ii) another one that does not use exogenous variables (*RAE_not_Exo*), and iii) a third one, were we explore a variant model (*RAE_Clf*) that outputs two regression channels for *volume* and *speed*, and five other outputs for the classification of the *heading* channel, minimizing the first two with an L2 loss and the classification with a softmax crossentropy. Then, we use all 7 outputs to finally output the 3 original channels and minimizing again with an L2 loss. This way we believe that outputs in the *heading* channel will be good for mse but more accurate in the only 5 possible outputs thanks to the classification task.

We also define two baseline models. The first is called *ConvLSTM* and consists of three layers of Convolutional LSTM (SHI et al., 2015) with 32, 64, 64 units followed by the *tanh* activation, and a final ConvLSTM layer with 3 units and a ReLu activation, all minimized with an L2 loss. Note that a final ReLu allows the model to predict the value of 1, which is in the domain of the *heading* channel (see section 2), whereas *tanh* would not. The second variant is called *ConvLSTM+Clf*, which adds an extra classification branch for the *heading* channel making the model multitask, with a second loss being a softmax crossentropy.

4. Results

The results of our experiments use the validation set provided by the challenge (see section 2). Note that this set is actually used as a test, since the models never saw these data before. Our own validation set was built with another 7 random days per city from the given training set, which were used to select the snapshots of the epoch with the best performance. In the challenge, the test set does not have the ground truth, so we can not compute locally the performance. At the end of this section, we report the score provided by the cloud system for the best of our models over.

Table 1 shows the results for baseline models in the three cities. We show the mean square error (mse), and the accuracy (acc) for the *heading* channel, computed comparing the only five possible values in the ground truth (see section 2) with the output of the regression task. We can see that adding the classification task (column *ConvLSTM+Clf* in Table 1) improves mse in Moscow significantly, whereas in Berlin gets only slightly better, and in Istanbul gets worst. On the other hand, accuracy for the *heading* channel improves

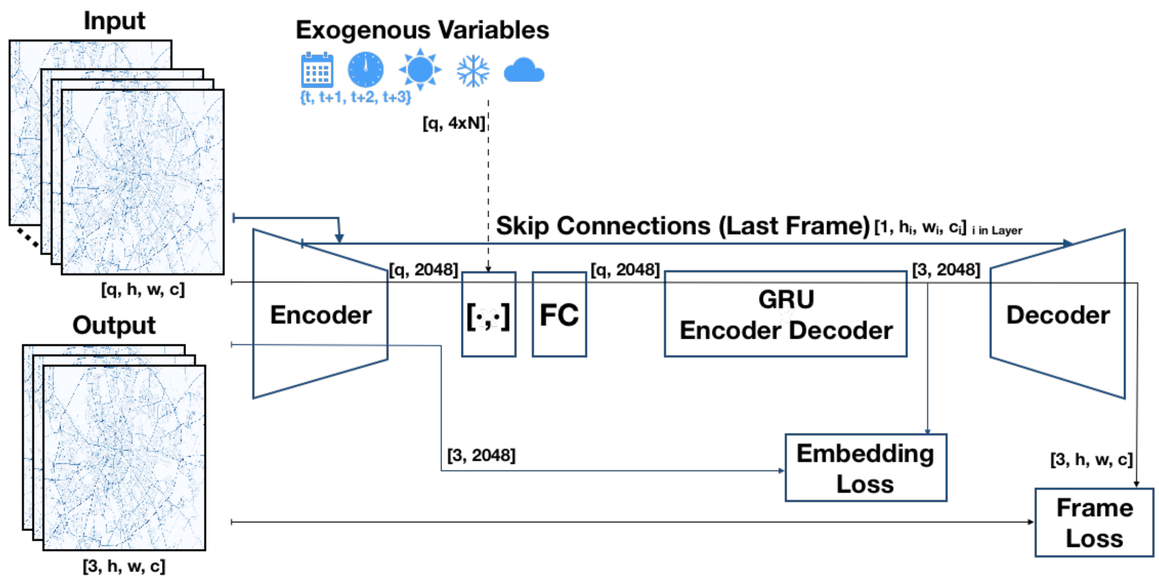

Figure 1: Recurrent Autoencoder with Skip Connections and Exogenous Variables. The Embedding Loss allows the recurrent layer GRU together with the encoder to produce better predictions in a low dimensional space. The Frames Loss and skip connections from sibling layers in the encoder empowers the decoder to produce outputs with high definition. Finally, the exogenous variables are concatenated with the encoder output before a fully connected layer followed by recurrent layers. Best seen in electronic form. Our method code is available at https://github.com/pherrusa7/Traffic4cast_NeurIPS_2019.

in two of the three cities, showing that the regression in this channel seems to benefit from the multitask approach.

We also tried using the output of the classification branch in the model *ConvLSTM+Clf*, as the output for the *heading* channel, by taking its *argmax* values. By doing so, acc increases from 0.455 to 0.803, but mse also increases from 0.012037826 to 0.023907198, making the important metric for the challenge worst. Our intuition is that the difference between the wrong prediction *heading* values now is bigger than using the default prediction minimized with mse, because softmax crossentropy does not lead to closer values when not guessing the correct one (it does not take into account inter-class similarities), so we discard this strategy for the challenge. We left the study of a classification loss with a distance for similar classes for future work, which could help the model to at least predict the most similar class when having a wrong prediction.

Baselines were only trained with the *non overlapping* sampling strategy, and input sequence with length $q = 3$, since training time for 1 epoch took more than 7 hours at an NVIDIA Titan RTX of 24GB. In comparison, 1 epoch of our proposed method needs only 35 minutes, mainly because recurrent layers act in a much lower dimensional space. Due to the time constraints of the challenge, the rest of our experiments focus only on improving the performance for the city of Moscow, since baselines show the worst results there. Future work will include comparisons also for the other cities.

Table 1: Baseline comparison for all three cities. We show global mse and accuracy of the heading channel together with epochs ('+' means that *ConvLSTM+Clf* started from last epoch in *ConvLSTM* with random weights in the classification branch).

mse (acc *heading*), #epochs	*ConvLSTM*	*ConvLSTM+Clf*
Moscow	0.012643729 (0.265), 3	0.**012037826** (0.455), +3
Istanbul	0.**009066001** (0.657), 3	0.0096280435 (0.686), +3
Berlin	0.0071446532 (0.536), 5	0.**007143738** (0.418), +1
mean	0.0096181277 (0.4860)	0.**0096032025** (0.**5196**)

In Table 2 we show the performance of the proposed method and its variants for Moscow. Note that the weights for the two terms in the loss function of eq. 2 were found through a small grid search, due to to time constraints, in the set $\{0, 0.5, 0.9, 1\}$, with best results achieved for $\alpha = 1$ and $\beta = 0.9$. The best model, in bold, is *RAE_all* (see Table 3 for more details), which was fine-tuned from *RAE_not_In* with a new skip connection from the last frame in the input sequence to the decoder, possibly allowing the model to further refine static regions. It is worth to mention that model *RAE_Clf* is really learning to be more accurate with the *heading* channel while preserving a mse of ~ 0.0144, which is better than $\sim$ 0.0239, the score achieved by the similar approach in the baseline. Future work will focus more on this model since we desire that *heading* outputs belong only to one of the five possible values in the domain, taking into account a metric for the inter-class similarity (Hou et al., 2016).

Table 2: Comparison of results for our proposed method and its variations against the baseline in Moscow city. We show global mse and the accuracy of the *heading* channel. Epochs are also shown with the particularity that if they are shown with format 10+5, it means that it is a fine-tuned model trained 5 epochs with loaded weights of the model at its left, that was trained 10 epochs already.

	ConvLSTM+Clf	*RAE_not_Exo*	*RAE_not_In*	*RAE_all*	*RAE_Clf*
mse	0.012037826	0.011873306	0.011875369	**0.011816756**	0.014442413
heading acc	0.455	0.469	0.453	0.437	**0.508**
epochs	4	10	10+5	15+3	44

Our proposed architecture for the city of Moscow together with the best baseline for Istanbul and Berlin achieved a mse score of 0.0098134960517874 in the challenge. We believe that using our method well trained for all cities will lead to much more better performance, which could not be tried due to time constraints and hardware limitations.

Table 3: Complete mse results in Moscow city for our best model *RAE_all*. As expected, the closer the time bin ahead to predict, the better the result is.

Moscow (mse: 0.011816756)			
	speed	*volume*	*heading*
5 minutes	0.000095909214	0.005128963	0.029793534
10 minutes	0.000102340266	0.0051734922	0.030223647
15 minutes	0.00010444787	0.0052143666	0.03051411

5. Conclusions

In this paper, a Recurrent Autoencoder with Skip Connections and Exogenous Variables has been proposed for Traffic Forecasting under a novel representation of traffic data, introduced by the Traffic4cast Challenge at NeurIPS 2019, that aggregated traffic data for cities into images, generating videos of traffic behavior. Our proposed method leverages the fact that the input is a sequence of frames, aggregating the spatio-temporal information in a lower-dimensional space and generating the output sequence in a single inference. The model uses two loss functions ensuring good predictions in the embedding space and high definition images in the original space. Exogenous variables like the day of the week, time of the day and weather have been also included. Furthermore, we propose a sampling method for sequences that run in parallel to the optimization process, producing rich batches in terms of diversity at each epoch. The result reported on the Traffic4cast Challenge is a mse of 0.0098134960517874 in the test set.

In future work, we will apply our method to Berlin and Istanbul and focus on the multitask method that uses the results of the classification of the *heading* channel taking into account the inter-class similarity as an input for the real regressive prediction. We will also work on the understanding of the intrinsic properties of geospatial models, in order to determine which deep learning architectures work better given the underlying rules characterizing the data (Jonietz and Kopp, 2019).

Acknowledgments

This work is supported by SEAT, S.A., and the Secretariat of Universities and Research of the Department of Economy and Knowledge of the Generalitat de Catalunya, under the Industrial Doctorate Grant 2017 DI 52. This research is also supported by the grant TIN2017-89244-R from MINECO (Ministerio de Economia, Industria y Competitividad) and the recognition 2017SGR-856 (MACDA) from AGAUR (Generalitat de Catalunya).

References

Yoshua Bengio. *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pages 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-

3-642-35289-8. doi: 10.1007/978-3-642-35289-8_26. URL https://doi.org/10.1007/978-3-642-35289-8_26.

Dr Michael Kopp Dr Sepp Hochreiter, Dr David Kreil. Institute of advanced research in artificial intelligence (iarai), 2019. URL https://www.iarai.ac.at/traffic4cast/#the-challenge. Accessed: 2019-10-24.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2016.

Wei-Chiang Hong. Traffic flow forecasting by seasonal svr with chaotic simulated annealing algorithm. *Neurocomputing*, 74:2096–2107, 06 2011. doi: 10.1016/j.neucom.2010.12.032.

Le Hou, Chen-Ping Yu, and Dimitris Samaras. Squared earth mover's distance-based loss for training deep neural networks, 2016.

W. Huang, G. Song, H. Hong, and K. Xie. Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2191–2201, 10 2014. ISSN 1524-9050. doi: 10.1109/TITS.2014.2311123.

David Jonietz and Michael Kopp. Towards modeling geographical processes with generative adversarial networks (gans) (short paper). In *COSIT*, 2019.

Binbing Liao, Siliang Tang, Shengwen Yang, Wenwu Zhu, and Fei Wu. Multi-modal sequence to sequence learning with content attention for hotspot traffic speed prediction. In *PCM*, 2018.

Sara Respati, Ashish Bhaskar, and Edward Chung. Traffic data characterisation: Review and challenges. *Transportation Research Procedia*, 34:131 – 138, 2018. ISSN 2352-1465. doi: https://doi.org/10.1016/j.trpro.2018.11.024. URL http://www.sciencedirect.com/science/article/pii/S2352146518303132. International Symposium of Transport Simulation (ISTS'18) and the International Workshop on Traffic Data Collection and its Standardization (IWTDCS'18)Emerging Transport Technologies for Next Generation Mobility.

Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wangchun WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 802–810. Curran Associates, Inc., 2015.

F. Sun, A. Dubey, and J. White. Dxnat-deep neural networks for explaining non-recurring traffic congestion. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2141–2150, 12 2017. doi: 10.1109/BigData.2017.8258162.

Online World Weather. Weather data and api, 2019. URL https://www.worldweatheronline.com/developer/. Accessed: 2019-07-30.

Nevan Wichers, Ruben Villegas, Dumitru Erhan, and Honglak Lee. Hierarchical long-term video prediction without supervision. In *ICML*, 2018.