# An Evaluation of the Performance of Convolutional LSTM and GRU on Forecasting Tasks.

Qihang Dai, 10578812

Final year project

For the BSc degree in Environmental Sciences

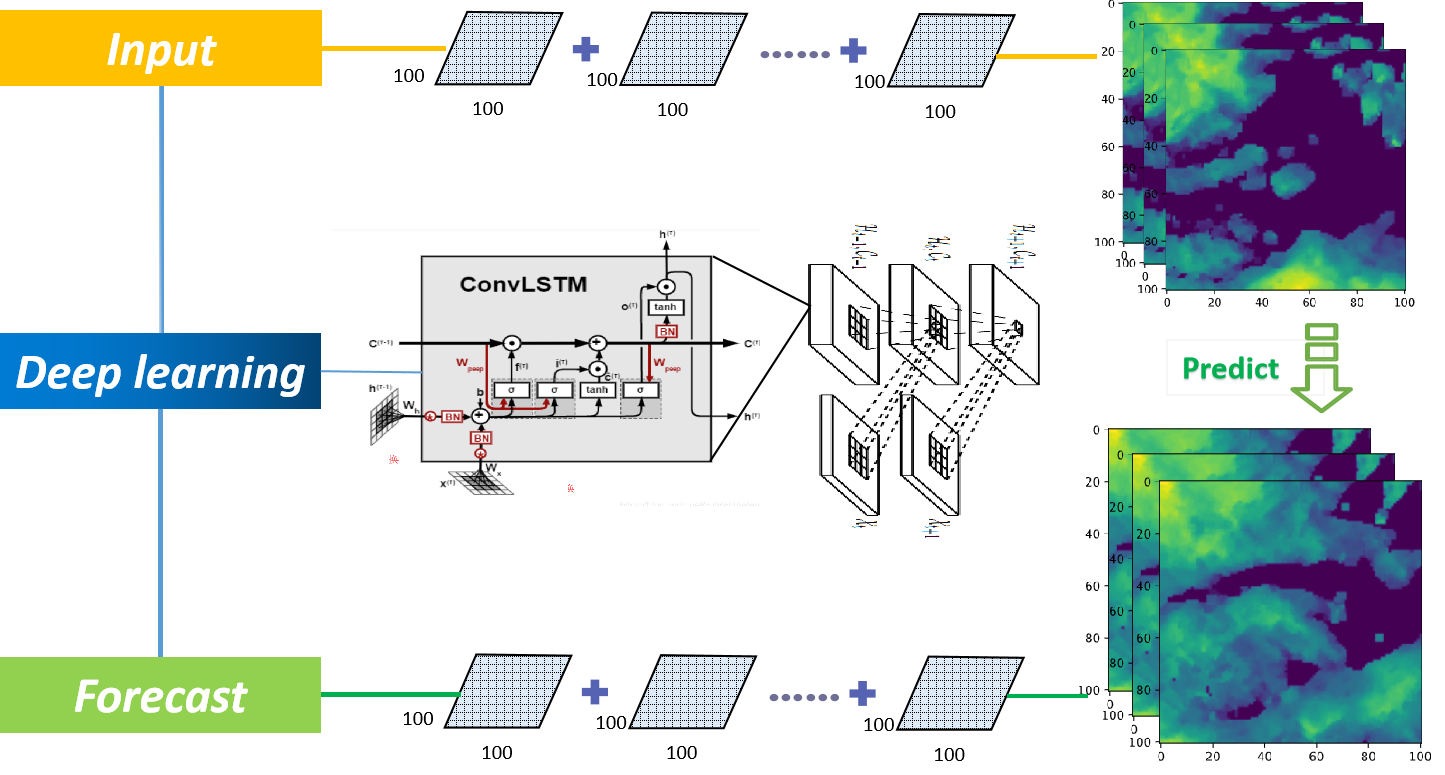
University of Manchester

Supervised by

David Schultz

Word Count:4107

Figure No.: 10



## Highlights:

* ConvLSTM performs better than ConvGRU on forecasting tasks.
* ConvGRU is less computing power and thus faster than ConvLSTM.
* Data preprocess and training tricks greatly improve the training speed in deep learning.

## Abstract

Recurrent neural network (RNN) and its variants such as long short-term memory (LSTM) and gate recurrent unit (GRU) have been widely used in two-dimensional time-sequence forecasting tasks. But, convolutional LSTM (ConvLSTM) and GRU (ConvGRU) that cope with three-dimensional data like sequences of radar images are still fresh and need a lot of testing about the practical value. This paper evaluates the performance of ConvLSTM and ConvGRU on Moving MNIST dataset and Shenzhen radar dataset. Experiments shows that ConvLSTM produce better forecasting results but ConvGRU is less computational demanding. Whether ConvGRU presents smaller loss or not, more white noisy points caused by overfitting appears in the prediction of ConvGRU. This paper also includes an elaborated explanation of data preprocess, model training and the mechanisms of ConvLSTM and ConvGRU.

**Keywords:** Deep learning; Radar images; Spatiotemporal forecasting.

## 1. Introduction

Precipitation nowcasting has been an important fields of weather forecasting for long. Apart from traditional forecasting methods like numerical weather prediction system, deep learning tools have been a popular method for precipitation nowcasting recently. Deep learning uses sequences of radar images as inputs and output predicted image sequences, treating the precipitation forecasting as video prediction tasks. Images are actually stored as a matrix of numbers in computers. Digits in each grid of a radar echo image represent the local precipitation rate. A consecutive series of radar echo thus represent the variation and movement of rainfall area. Therefore, computers can observe some patterns from historical precipitation data for a better prediction of the variations of future precipitation.

Deep learning models, such as RNN, LSTM and GRU can make prediction via learning temporal relationships in historical two-dimensional data. However, in precipitation nowcasting, spatial relationships exist in the different digits of radar echo in one image. Normal RNN, LSTM and GRU for processing two-dimensional time sequence data cannot learn the spatial features. Convolutional LSTM (ConvLSTM) was invented to solve this problem (Shi et al., 2015). Convolution neural networks (CNN) have been widely used for image recognition tasks, showing strong representative learning ability of spatial information. Embedding convolution structure with time sequence prediction models, deep learning can successfully produce liable forecasting results of precipitation in the next few hours (Ayzel et al., 2019). Other combinations like convolutional RNN and convolutional GRU are also adopted in spatiotemporal prediction tasks like precipitation nowcasting.

Many studies compared these spatiotemporal prediction models on different datasets (Trebing and Mehrkanoon, 2020; Zhang et al., 2019). By evaluating the difference of design and performance of these spatiotemporal prediction models, scientists have the opportunity to discover the adaptability and shortages of current models. Based on the evaluation of previous models, more advanced deep learning model for spatiotemporal tasks could be designed. At the same time, the evaluation of the old models can help the forecasting department to use the deep learning model more effectively in real precipitation nowcasting. Similarly, this study aims to conduct a performance evaluation of ConvLSTM and ConvGRU on different datasets. At first this paper explains the basic concepts that help with understanding the mechanisms of ConvLSTM and ConvGRU. Then this paper introduces the datasets and describe experiments. At last, this paper evaluates the forecasting performance of the two model. This study trains and tests the forecasting ability of the two models on Moving MNIST dataset and Shenzhen radar dataset.

## 2. Deep learning concepts

### 2.1 Backpropagation

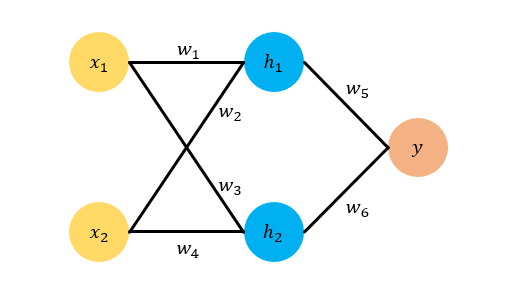


Fig. 1 A simplified example of neural networks with *xi, wi, hi* and *y* representing inputs, weights, hidden layer nodes and output respectively (Ma, 2018).

Firstly, this paper describes the mechanism of deep learning training. Fig. 1 is a simple example of neural networks. The *xi, wi, hi* and *y* represent inputs, weights, hidden layer nodes and output respectively. Neural networks can be seen as a mathematic function with lots of weights. Training the neural network is to adjust the inside weights *wi* to approach the correct output *y* via backpropagation. Backpropagation includes two key element, loss and optimizer that is the function to update weight.

In the above function represents the input data. is the target value or ground truth. Learning rate decides the extent of each step adjustment and is manually set before the training. For minimizing the error (loss) between the output of the neural network with the ground truth, at the end of a training epoch the weights of next epoch would be updated based on the weights and loss of previous epoch. Therefore, after many epochs of training, the neural network grows into a model with a set of stable weights of which the output is very close to the ground truth, namely a well-trained deep learning model.

### 2.2 Convolution, RNN, LSTM and GRU.

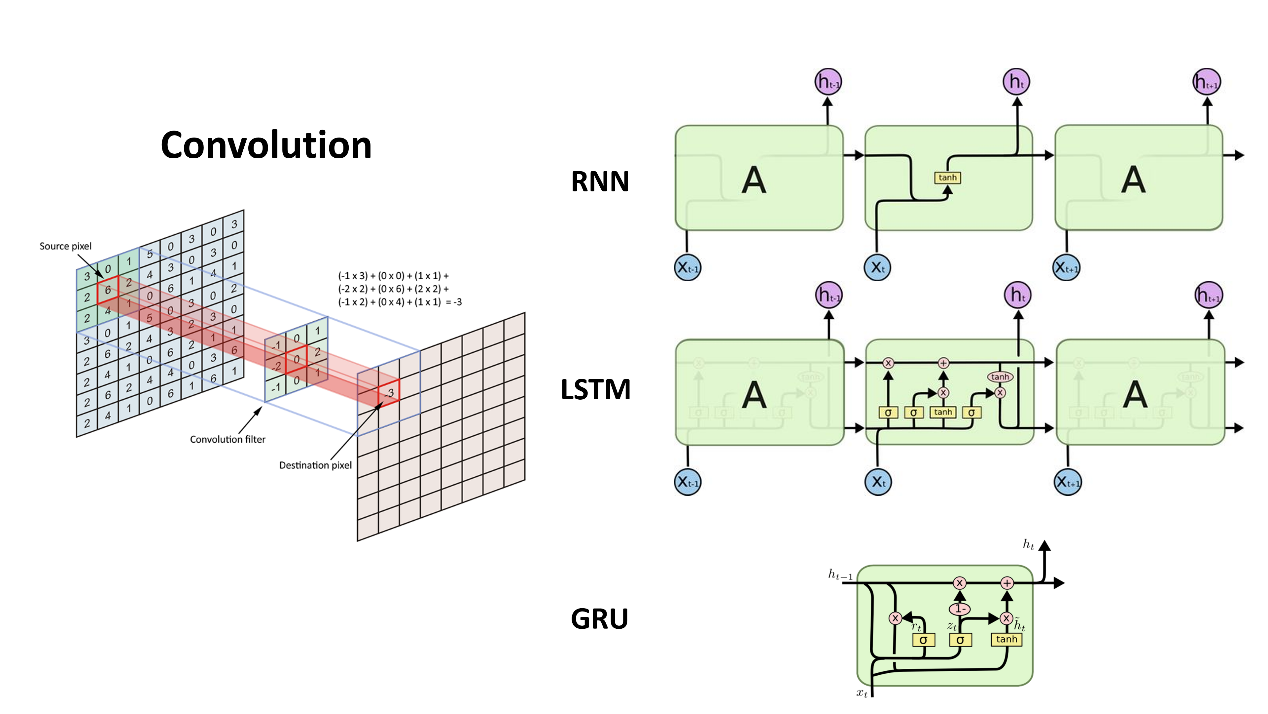


Fig. 2 Structures of convolution, RNN, LSTM and GRU (Jiang, 2016; Palagi et al., 2018).

Next this paper introduces how deep learning learn spatial and temporal information from data. As the inputs of deep learning on precipitation nowcasting tasks, sequences of radar echo images hold both spatial and temporal information. Fig 2 is the visualizations of convolution, RNN LSTM and GRU. In the field of deep learning, convolution is used to extract spatial features from deluge of data. A colorful image holds 103 pixels in height and width, and 3 parameters to record RGB information. A total of parameters is used to present the information of the images. The huge number of parameters would significantly lower the processing speed since one neural network usually need to process hundreds and thousands of pictures. As shown in Fig. 2, convolution uses a matrix kernel to scan over the original image matrix by a specific stride, calculates out one value of each step of scan, and then gets an image of smaller size. Conducting another convolution on the generated smaller image, the image size could be furtherly reduced. Conducting the reverse calculation on the smaller image, the original-size image is reproduced without losing important features. Hence the parameter amount of a picture is largely reduced while the information in the picture is effectively restored.

As for the temporal relationship in image sequences, RNN can extract temporal information from data by treating the output of last time step as part of the input of the current step. In this way RNN can learn the difference between each step prediction and the input of next step, and gradually adjust the weights to reduce the difference. After learning enough samples from known dataset on train dataset, RNN could eventually output predictions of future time steps on test dataset. However, as the time sequence becomes longer, the time step long before current sequence may have less and less influence on the output of current step due to the gradient vanishing and explosion problem in back propagation algorithm. Because the gradient is multiplied steps by steps in the process of backpropagation, if all gradients are lower or bigger than one, the final gradient would be infinitely large or small, i.e., gradient vanishing and explosion. LSTM and GRU are two variations of RNN that solve the gradient problems. A single cell (timestep) of LSTM and GRU consists gate units which can calculate the weight of last step data and store the weight in a cell gate. Then the cell gate would accumulate the value and pass it to next step, while also generate part of the inputs of the next cell (timestep). In this way the influence of previous timesteps is stored via accumulation rather than multiplying. Thus, the influence of important information from far previous steps could be successfully transmitted to the current step, and the gradient in the whole neural network would not vanish or explode. The main difference between GRU and LSTM is that GRU adopted fewer gates in the cell, which make GRU computational cheaper than LSTM, i.e., less parameters and faster processing speed. Nevertheless, GRU does not always have a higher accuracy compared to LSTM (Fu et al., 2016; Shewalkar, 2019).

### 2.3 ConvLSTM

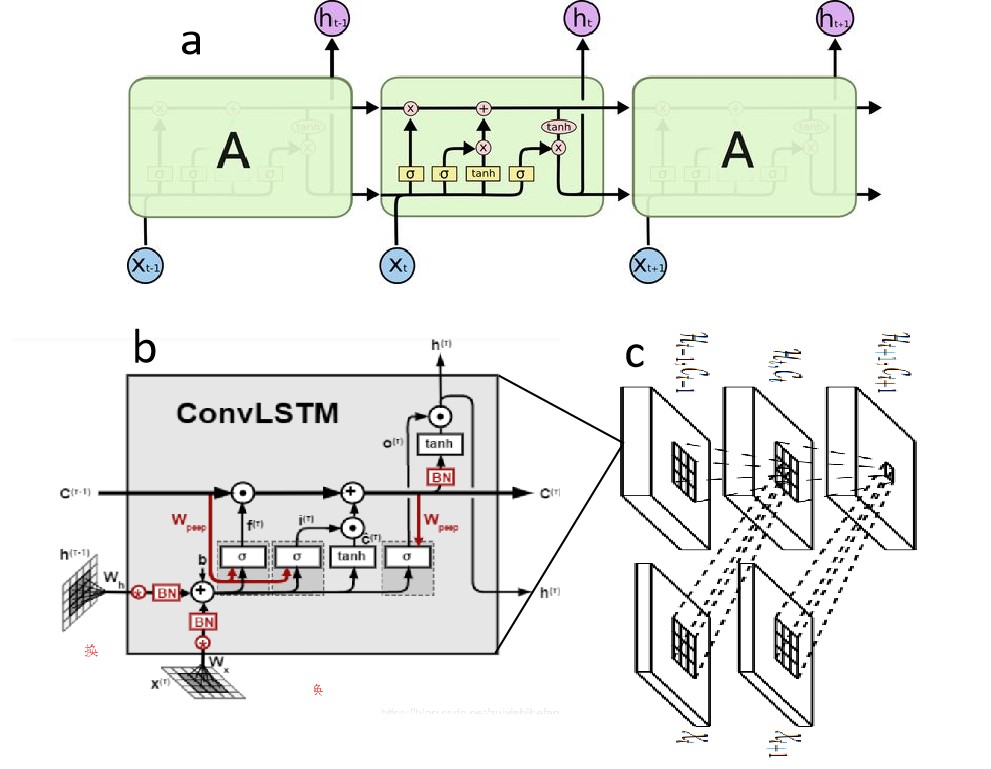


Fig. 3 Structures of (a) LSTM, (b) ConvLSTM cell, and (c) ConvLSTM network (Shi et al., 2015).

For two-dimensional data, traditional RNN, LSTM and GRU calculate the inputs into one value and transfer it to next unit. However, for three-dimensional input like images, spatial relationship within different part of one image is lost via converting the whole image into one value. To solve this problem, ConvLSTM that adopts convolution operation in data transmission is invented (Shi et al., 2015). The computing of inputs, outputs, and inner gates is transformed from previous simple operation into convolution operation, and matrixes with spatial information rather than single value is passed through the cell (Fig. 3b). Transmitting output matrix from one ConvLSTM cell as the input of the next cell, a ConvLSTM network (Fig. 3c) that has similar structure with LSTM could extract both spatial and temporal information from a consecutive sequence of images, namely videos. ConvLSTM networks (Fig. 3c) can also have multiple layers. The outputs of one layer of ConvLSTM network are fed as the inputs of another layer of ConvLSTM where the ConvLSTM cell would future extract more abstract and regular features of the video. Shi et al., 2015 used two layers of ConvLSTM on radar images of good quality. Considering the quality of dataset in this study is not as good as, this study adopted three layers of ConvLSTM based on the codes in <https://github.com/jhhuang96/ConvLSTM-PyTorch>.

### 2.4 Encoder-Decoder structure

In Shi et al., 2015, Encoder-Decoder structure was adopted for the whole neural networks. While RNN, LSTM and GRU have the same amount of output as input (Fig. 2), Encoder-Decoder structure can have different numbers of input and output. The encoder is the first half part of the entire neural network that compressed the input into a fixed length vector. Then the fixed length vector is fed as input to the decoder, i.e., the other half of the neural network. In this way encoder and decoder are not tightly interconnected and can have different customized inner structure. The input and output could have different lengths. Classic Encoder-Decoder structure is widely used in text translation to solve this problem (Cho et al., 2014). For example, an English sentence could have different numbers of words compared to the Chinese sentence with the same meaning. Traditional RNN, LSTM or GRU can only translate *n* words long English to *n* words long Chinese. With Encoder-Decoder structure, RNN, LSTM and GRU can translate *n* words long English to *m* words long Chinese.

In this paper, ConvLSTM and ConvGRU also have encoder-decoder structures. Both encoder and decoder have three layers of ConvLSTM networks to learn the movement of precipitation efficiently. Using Encoder-Decoder structure, we can use random numbers of input frames to predict random numbers of output frames. For example, Shi et al., 2015 use five frames input to predict 20 future frames, which is very computational demanding. As shown in Fig. 4, at first this study omitted the last time step of the 15 timesteps in preprocessed radar dataset, and then use the first seven frames to forecast the later seven frames.

## 3. Data

### 3.1 Moving MNIST dataset

MNIST dataset is a classic machine learning dataset used for image processing tasks (Srivastava et al., 2016). It contains ten handwriting digits from 0 to 9, half written by American Census Bureau employees and half by American high school students. And the test set of MNIST contains 10000 testing images of 28 pixel. Moving MNIST is a variant dataset of MNIST test dataset and designed for testing sequence prediction ability of machine learning models. Moving MNIST contains 10000 sequence of video 20 frames long videos. In each video digits are moving in a 6464 frames. The digits are randomly picked from the MNIST and assigned a random initial location and constant velocity inside the 6464 pixels frames. Moving MNIST dataset with two moving digits can be directly downloaded from University of Toronto (<http://www.cs.toronto.edu/~nitish/unsupervised_video>), or be generated from MNIST with custom number of digits. Moving MNIST with three digits is adopted for this research.

### 3.2 Radar echo dataset

International Conference on Data Mining 2018 Global A.I. Challenge on Meteorology (ICDM 2018) provides standard radar dataset that contains 320 thousands of six-hour radar images at an interval of 6 minutes. The task of the competition is to use the first three-hour data to predict the after three-hour data. The dataset of ICDM 2018 perfectly suits the topic of our paper. However, the download portal of the dataset is permanently removed due to confidentiality issues. Therefore, we turn to use the radar echo datasets from the Conference on Information and Knowledge Management Analytic Cup 2017 (CIKM 2017). The datasets are jointly provided by SZMB and Alibaba. Nevertheless, since the aim of CIKM 2018 is to forecast the precipitation rate at the center of the radar map, the datasets are not series of radar pictures like that in ICDM 2018, but txt file. Data in the txt files is a series of number split by space in the format of id and radar map. Radar maps are in the format of time (T), height (H), length (Y) and width (X), i.e., THYX. Each radar map has 15 time-steps at the interval of six minutes, four level of heights (0.5 km, 1.5 km, 2.5 km and 3.5 km) and 101101 km grids. 154101101 values exit in one radar map. Thus, the whole datasets are five dimensions in the format of ITHYX (Fig. 4). For legal reasons, SZMB transformed the datasets linearly due to the confidentiality issue, making it impossible to retrieve real precipitation from the data. Thus in this study we mainly focus on the model performance of forecasting ability rather than investigating further into precipitation. The train dataset is collected from three-year historical data with 10000 radar maps. The test dataset is collected from another two-year historical data and have 2000 radar maps respectively.

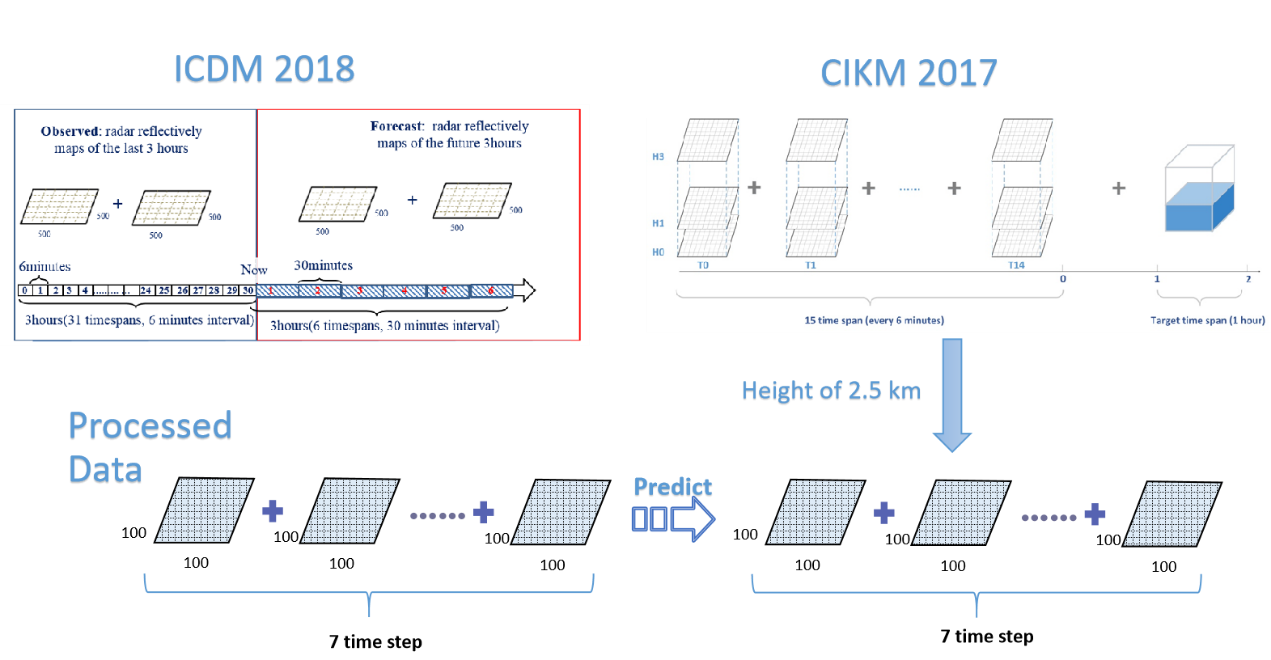


Fig. 4 Visualizations of datasets of ICDM 2018, CIKM 2017 and processed data for this study (Tianchi: Data Sets).

## 4. Methods

### 4.1 Data preprocess.

The original datasets occupy 15.8 GB for train set (1000015 images) and 3.2 GB (200015 images) for one test set, which is a huge storage that would significantly slow down the processing of files. In order to reduce the file size, data format is transformed from utf-8 (occupying 8 to 32 bits per character) into ubyte (8 bits integral data type). The storage space of train dataset is significantly reduced to 5.7 GB, and that of test dataset is reduced to 1.1 GB.

Fig. 4 is visualizations of datasets for ICDM 2018, CIKM 2017 and the preprocessed data of this study. Since CIKM 2017 datasets are not a series of images, there is a need to transform CIKM 2017 dataset to similar format of ICDM 2018 dataset. As shown in Fig. 4, the radar maps of 2.5 km height are extracted from the five-dimension datasets and stored in format of npz that is used to save fast reading numpy matrixes. To suit the input format of convolution layers, the 101 matrix is transformed to 100100 with the first line and row omitted. Among the 15 images for each sequence, we choose the first seven images as the input and produce seven predictions. All source codes for this study can be found on my GitHub repo (<https://github.com/553269487/ConvLSTM-on-TIANCHI-CIKM-2017>).

### 4.2 Training

Pytorch is used for deploying models in this study. Pytorch is an open source deep learning framework designed by Facebook (Ketkar, 2017). Compared to other frameworks like Tensorflow and Keras, Pytorch has a good balance between simplicity and customization. Pytorch provides complete and stable dependency for graphics processing units (GPU) which can train deep learning models much faster than central processing units (CPU).

Training and testing are conducted on a single laptop version GTX 2060 GPU. Despite moving the training from CPU to GPU which sharply increases the training speed, per epoch training still took 35 minutes, which is not acceptable because this study expected to have 20 epoch training for each model. Several tricks were used to speed up the process. First, pushing data from CPU to GPU frequently would significantly slow down the model processing because the difference in accessing speed between CPU and GPU. To reduce the frequency as lowest as possible, we set the batch number (amounts of the sets of data that the model process per iterative) and number workers (amounts of batch loaded from storage to GPU each time) to the maximum level that the laptop CPU and RAM can afford. Besides, setting ‘cudnn.benchmark’ as true enabled the Pytorch to find the most optimized algorithm in data processing. At last mixed precision training is invoked. Rather than treating all numbers as 32 bits, mixed precision training automatically treats part of parameters as 16 bits, which would dramatically reduce the training time cost. Using all the tricks above, the time cost per epoch is reduced from around 30 minutes to 10 minutes for Moving MNIST dataset.

As mentioned in Section 2.2, the two important factors that influence the training of a model is loss and optimizer. Mean squared error (MSE) is a simple loss function that require less computation than other loss function. Models with Adaptive Moment Estimation (Adam) usually outputs more reliable results than other optimizers (Kingma and Ba, 2017). Here we choose MSE as loss and Adam as optimizer.

## 5. Results and Discussions

### 5.1 Moving MNIST train and test

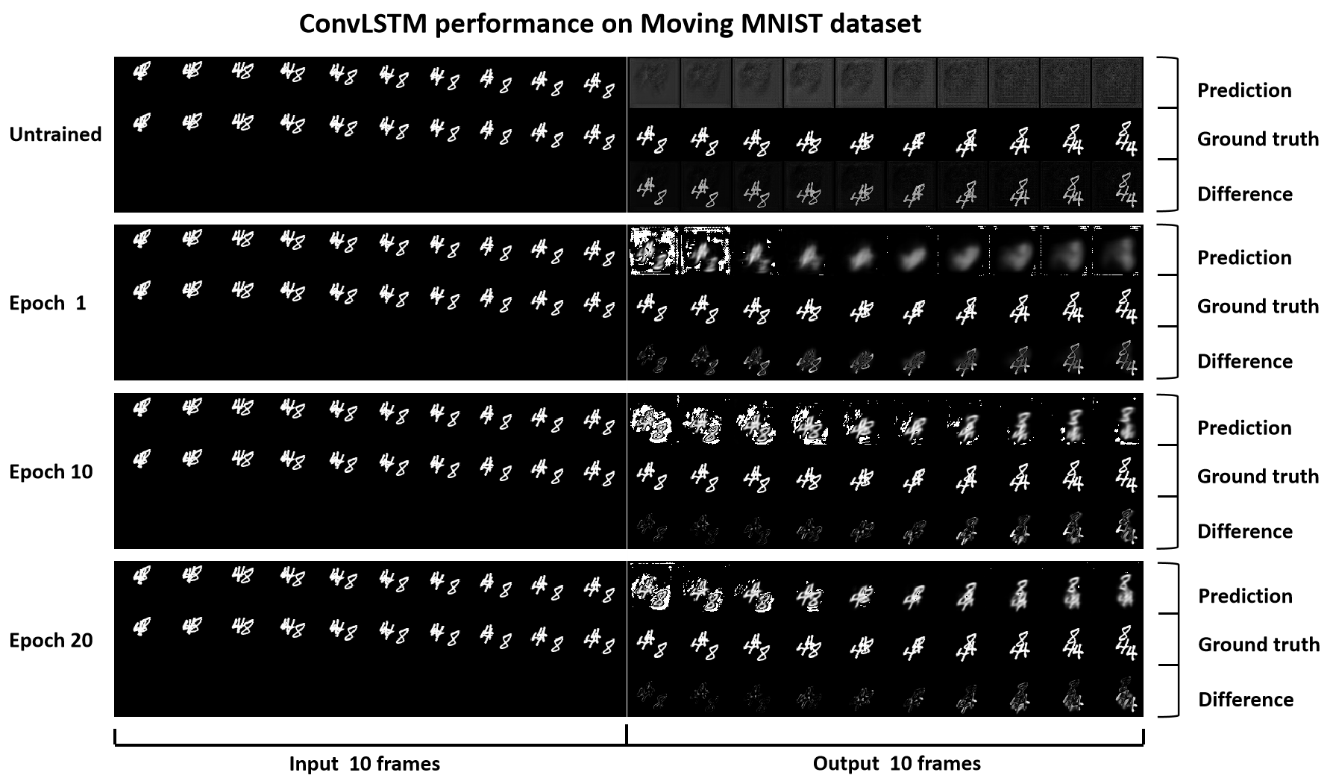


Fig. 5 ConvLSTM predictions on test dataset using untrained model, one epoch trained model, 10 epoch trained model and 20 epoch trained model.

Ten thousand sequences of Moving MNIST are used for training and 2000 sequences for testing. Twenty epochs of training are conducted, and the parameters at the end of each epoch are saved as models. Then we run the models of epoch one, 10 and 20 on test dataset. Training took around 15 minutes per for ConvLSTM and 10 minutes for ConvGRU. Besides the batch number of ConvGRU can be up to eight while that of ConvLSTM is four. Thus, it is validated that ConvGRU is more computational efficient than ConvLSTM, not only demanding less RAM but also having faster training speed.

Fig. 5 presents the forecasting results of ConvLSTM. The predicted frames becom less blurry as the epoch increases. When using an untrained model for testing, the prediction of moving digits presents a whole image of fuzzy area. However, after merely one epoch of training, ConvLSTM generates visible digits. Despite the white noisy points in the first two frames and vague shapes in the afterwards eight frames, the digits’ movements are approaching the ground truth. In epoch 10, the shape of digits become clearer despite some noisy points. In epoch 20, the first 5 predicted frames have less white point and are the most similar to the ground truth. The shapes of last 5 frames still show some extent of vague but no noisy point. The last 2 frames are distorted and present recognizable difference with ground truth. Shown in the row of difference in Fig. 6, the predicted shapes of earlier frames are more closed to the ground truth although there are more noisy points than later frames.

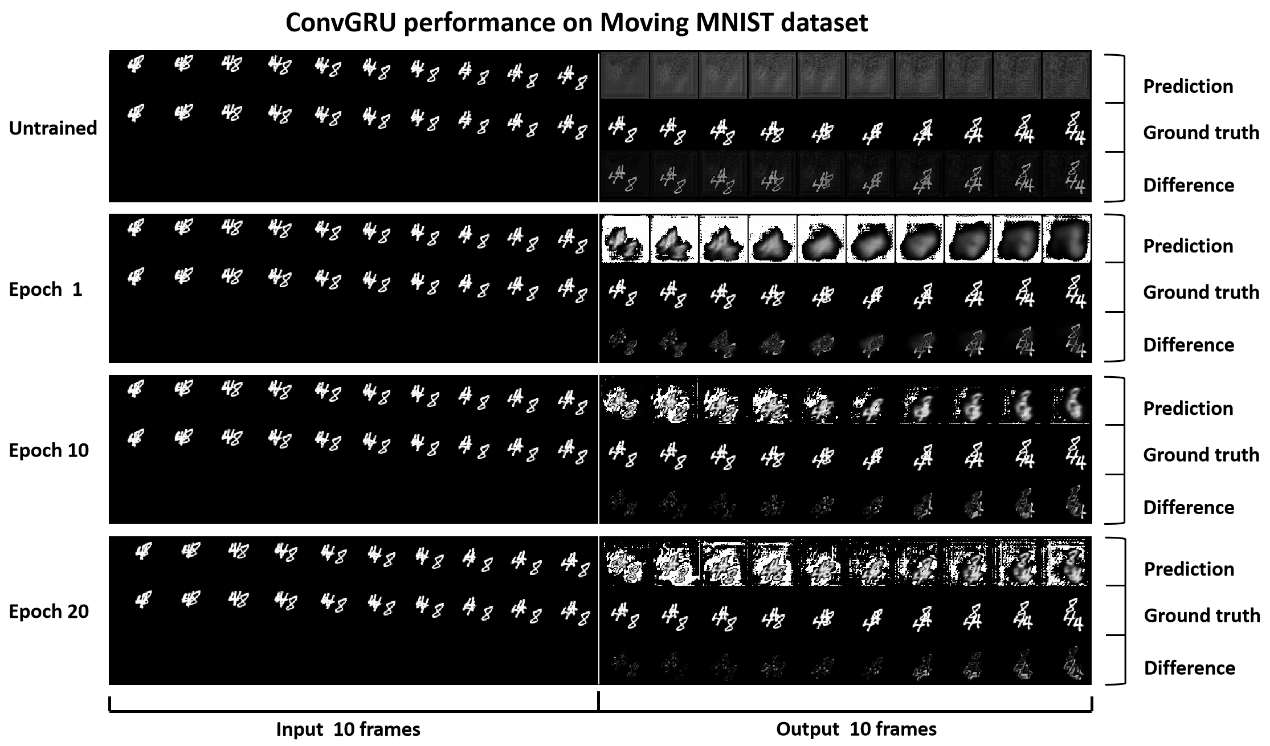


Fig. 6 ConvGRU predictions on test dataset using untrained model, one epoch trained model, 10 epoch trained model and 20 epoch trained model.

As shown in Fig. 6, the predictions of ConvGRU are different from those of ConvLSTM. In epoch one, ConvGRU generates large area of noisy points. Later frames showed less noise than earlier frames. From the prediction row and difference row we can observe that ConvGRU captures the shape of digits efficiently. In epoch 10, white noise area largely decreases and the shapes of digits proceeds to approaching ground truth. In epoch 20, the digit shapes of later frames are more similar to the ground truth while that in the earlier frames does not show obvious improvement compared to epoch 10. White noise area grew larger as a suggestion of overfitting.

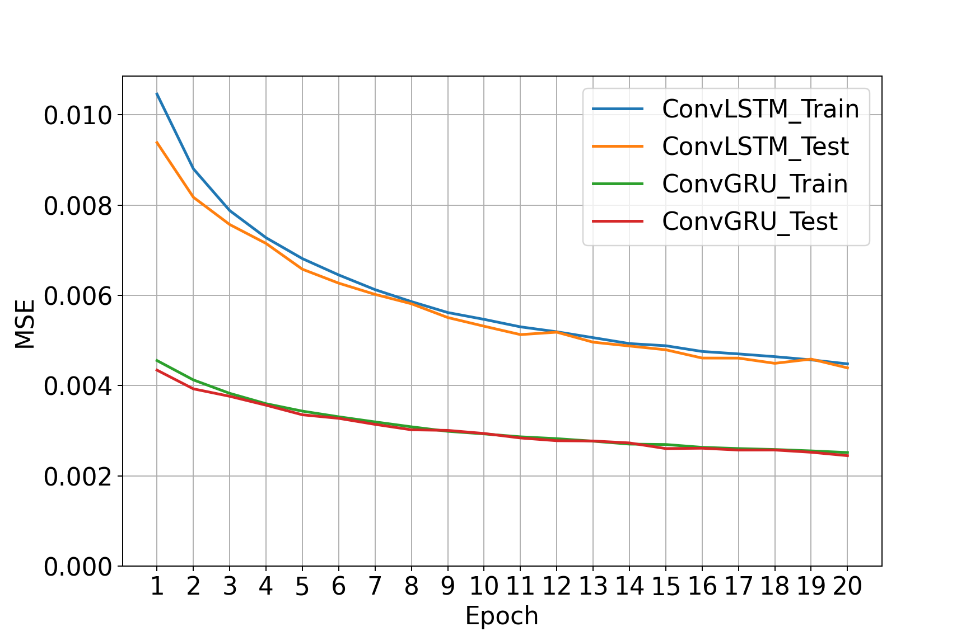


Fig. 7 The variations of MSE of ConvLSTM and ConvGRU.

Fig. 7 presents the MSE variations of ConvLSTM and ConvGRU on Moving MNIST dataset.

As shown in Fig. 5-6, the improvement of forecasting accuracy is more obvious before epoch 10. The gradient of MSE remains at a stable range form epoch 10 to epoch 20. This phenomenon is then validated by Fig. 7 where the decreasing speed of loss slows down apparently after epoch 10. Meanwhile, MSE for ConvGRU is around 50% lower than MSE for ConvLSTM in both train and test. Despite ConvGRU performs better in reducing the loss value, ConvLSTM produces more clear prediction with less noise point. The low loss for ConvGRU indicates that overfitting is the reason for more noisy points in the predictions of ConvGRU. Overall, ConvLSTM performed better than ConvGRU on Moving MNIST dataset.

### 5.2 Radar echo train and test.

Compared to highly-standardized Moving MNIST, the radar echo dataset has less frames and larger matrixes that increase the computational difficulty. Average train time per epoch is 15 minutes for ConvGRU and 23 minutes for ConvLSTM. The chaos nature of atmospheric movement also increases the randomness of radar dataset. The 2000-sequence test dataset could have precipitation patterns never occurred in the 10000-sequence train dataset. Unlike the steady loss decrease for both test and train in Moving MNIST, loss in radar echo test decreases slowly and rebounds frequently. Nevertheless, the average MSE on radar datasets is an order of magnitude smaller than that on Moving MNIST.

Early stopping is set for the training on radar echo dataset. For every epoch of training a testing is conducted. The average loss of test set is recorded. If the new test loss is not smaller than last 5 epochs, the training process would be automatically stopped. As shown in Fig. 8, the test loss of ConvGRU reaches lowest at epoch 11 and training of ConvGRU stopped at epoch 16. The test loss of ConvLSTM reaches lowest at epoch 15 and training of ConvLSTM is manually stopped at epoch 17. On train dataset, the loss decreases at a nearly constant rate, and the loss of ConvLSTM is smaller.

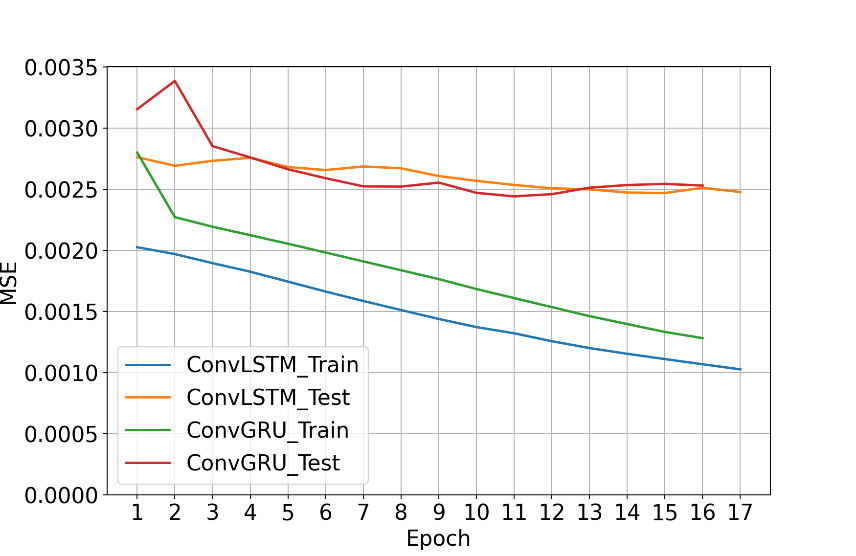


Fig. 8 The variations of MSE loss for radar echo test and train of ConvLSTM and ConvGRU.

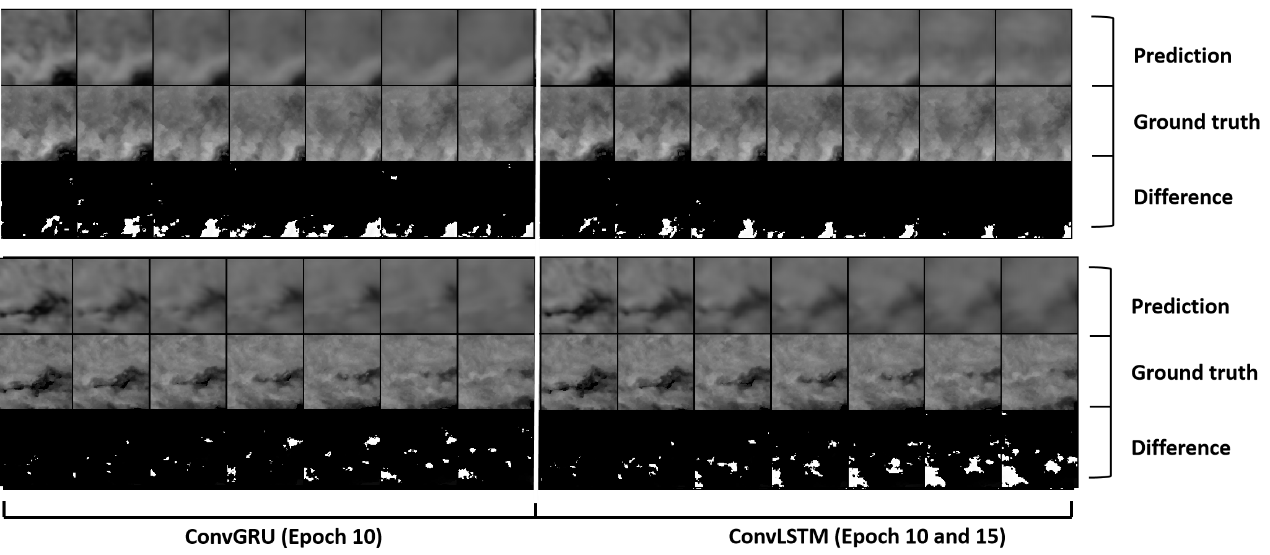


Fig. 9 Predictions on radar train dataset. All predictions of ConvGRU are based on epoch 10 model (smallest test loss). For ConvLSTM, the upper prediction is based on epoch 10 model, and the bottom prediction is on epoch 17 model (smallest train loss).

Since there is obvious difference on MSE variations between train and test dataset, observing the prediction on train dataset may be meaningful. The ConvGRU of epoch 10 (output lowest test loss), ConvLSTM of epoch 10 (for comparsion) and ConvLSTM of epoch 17 (output lowest train loss) are chosen to produce prediction images on train dataset. As shown in Fig. 9, for both models, predictions of first two frames are close to the ground truth, but predictions of later frames tend to be blurry as the forecasting steps grow longer. From the rows of difference we can see, at epoch 10, the forecasting results of ConvGRU and ConvLSTM are very similar. But at epoch 17, ConvLSTM produces overfitting results with more noisy points than ConvGRU.

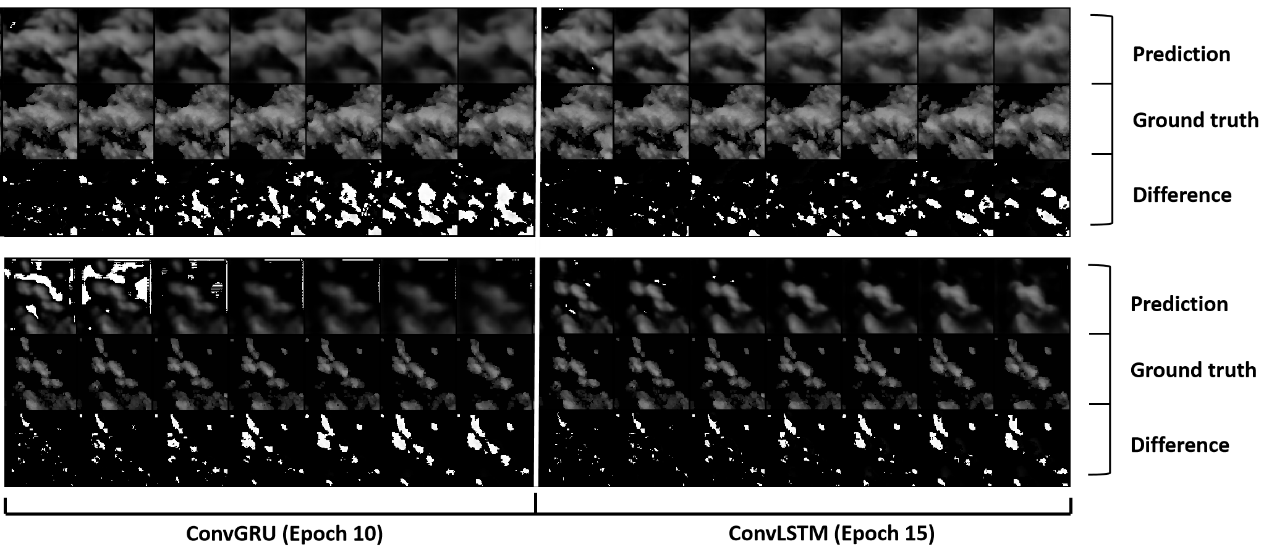


Fig. 10 Predictions on radar test set.

On test set, ConvLSTM has better performance than ConvGRU (Fig. 11). Forecasting frames of ConvLSTM are more distinct as well as more correct. In the upper predictions, ConvLSTM successfully generates the growth of the small cluster of echoes (clouds) at the bottom of matrixes. In contrast the bottom of matrixes remains almost the same among the seven frames in ConvGRU. The row of difference also reveals that the forecast of ConvGRU has larger bias to ground truth. Although the MSE loss of ConvGRU (0.00244) is around the same as ConvLSTM (0.00247), ConvLSTM still have better forecasting performance than ConvGRU.

Because the majority matrixes in train set (extracted at height of 2.5 km) are covered by large area of radar echo (Fig. 9), deep learning models could produce overfitting results on sequence with small clusters of radar echoes on test dataset. Large area of white noise points appears in first two frames of the second row ConvGRU predictions but not in the upper row predictions. But ConvLSTM does not exhibited this deficit.

## 6. Conclusion and Future Work

Both ConvLSTM and ConvGRU can successfully capture spatiotemporal information from historical data and output forecast of future images. ConvLSTM has better performance on forecasting tasks than ConvGRU, producing clearer images with less noise points and capturing movement patterns that ConvGRU failed to learn. ConvGRU is less computing demanding and thus faster than ConvLSTM. The training time of ConvGRU is around two thirds of that of ConvLSTM. Nevertheless, ConvGRU suffers from overfitting problem and produce extra noise points.

Despite the two models can output recognizable forecast, they still produce blurry images in forecast of later frames. One possible solution is using more input frames to predict fewer output frames via Encoder-Decoder structure, which reduce the complexity of prediction. Changing loss function from simple MSE to other loss like cross entropy, trying different kernel sizes may also improve the forecasting performance. New deep learning models like TrajGRU, predRNN and Attention Unet have been proven to have stronger ability in radar image based video prediction tasks (Shi et al., 2017; Trebing and Mehrkanoon, 2020; Yu et al., 2019).

## References

Ayzel, G., Heistermann, M., Sorokin, A., Nikitin, O., Lukyanova, O., 2019. All convolutional neural networks for radar-based precipitation nowcasting. Procedia Computer Science 150, 186–192.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078 [cs, stat].

Fu, R., Zhang, Z., Li, L., 2016. Using LSTM and GRU neural network methods for traffic flow prediction, in: 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC). Presented at the 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), IEEE, Wuhan, Hubei Province, China, pp. 324–328. https://doi.org/10.1109/YAC.2016.7804912

Jiang Z., 2016. What is Convolution? [WWW Document]. Zhihu. URL https://zhuanlan.zhihu.com/p/30994790 (accessed 4.18.21).

Ketkar, N., 2017. Introduction to PyTorch, in: Ketkar, N. (Ed.), Deep Learning with Python: A Hands-on Introduction. Apress, Berkeley, CA, pp. 195–208. https://doi.org/10.1007/978-1-4842-2766-4\_12

Kingma, D.P., Ba, J., 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs].

Ma S., 2018. Back Propagation Explanation with Examples [WWW Document]. Zhihu. URL https://zhuanlan.zhihu.com/p/40378224 (accessed 4.18.21).

Palagi, L., Pesyridis, A., Enrico, S., Tocci, L., 2018. Machine Learning for the prediction of the dynamic behavior of a small scale ORC system. Energy 166. https://doi.org/10.1016/j.energy.2018.10.059

Shewalkar, A., 2019. Performance evaluation of deep neural networks applied to speech recognition : RNN, LSTM and GRU. Journal of Artificial Intelligence and Soft Computing Research 235--245. https://doi.org/10.2478/jaiscr-2019-0006

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W., Woo, W., 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. arXiv:1506.04214 [cs].

Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.-Y., Wong, W., WOO, W., 2017. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp. 5617–5627.

Srivastava, N., Mansimov, E., Salakhutdinov, R., 2016. Unsupervised Learning of Video Representations using LSTMs. arXiv:1502.04681 [cs].

Tianchi: Data Sets [WWW Document], n.d. URL https://tianchi.aliyun.com/dataset/dataDetail?dataId=1085 (accessed 3.15.21).

Trebing, K., Mehrkanoon, S., 2020. SmaAt-UNet: Precipitation Nowcasting using a Small Attention-UNet Architecture. arXiv:2007.04417 [cs, eess].

Yu, B., Yin, H., Zhu, Z., 2019. ST-UNet: A Spatio-Temporal U-Network for Graph-structured Time Series Modeling. arXiv:1903.05631 [cs, stat].

Zhang, W., Li, W., Han, L., 2019. A Three-Dimensional Convolutional-Recurrent Network for Convective Storm Nowcasting, in: 2019 IEEE International Conference on Big Knowledge (ICBK). Presented at the 2019 IEEE International Conference on Big Knowledge (ICBK), pp. 333–340. https://doi.org/10.1109/ICBK.2019.00052