

Neural Networks for Nowcasting of Weather

AI60002 Lecture 27 April

Nowcasting

- Forecasting at short time ranges (order of hours)
- Usually done for precipitation, thunderstorms, lightnings etc
- Real-time performance very important
- Spatial and temporal precision very important
- Usually based on observations from satellites, radars, automatic weather station
- Usually done with numerical models – slow and computationally expensive
- Can machine learning help?

Short-term Rainfall Forecasting Using Multi-layer Perceptron

Pengcheng Zhang, Yangyang Jia, Jerry Gao, Wei Song, Hareton Leung

Abstract—Rainfall forecasting is crucial in the field of meteorology and hydrology. However, existing solutions always achieve low prediction accuracy for short-term rainfall forecasting. Numerical forecasting models perform worse in many conditions. Machine learning approaches neglect the influences of physical factors in upstream or downstream regions, which make forecasting accuracy fluctuate in different areas. To improve the overall forecasting accuracy for short-term rainfall, this paper proposes a novel solution called Dynamic Regional Combined short-term rainfall Forecasting approach (DRCF) using Multi-layer Perceptron (MLP). First, Principal Component Analysis (PCA) is used to reduce the dimension of thirteen physical factors, which serves as the input of MLP. Second, a greedy algorithm is applied to determine the structure of MLP. The surrounding sites are perceived based on the forecasting site. Finally, to solve the clutter interference which is caused by the extension of the perception range, DRCF is enhanced with several dynamic strategies. Experiments are conducted on data from 56 real-world meteorology sites in China, and we compare DRCF with atmospheric models and other machine learning approaches. The experimental results show that DRCF outperforms existing approaches in both threat score (TS) and root mean square error (RMSE).

Index Terms—Rainfall forecast, deep neural network, multi-layer perceptron, short-term, atmospheric models



Multi-layer Perceptron for Rainfall Nowcasting

3.3 Adjustment and Constructive Algorithm of MLP

MLP is a kind of feed-forward network. It is calculated from the input layer to the output layer successively. Each node at the same level is calculated at the same time, and it does not interfere with each other [42]. The value of every node is equal to the weighted summation of all nodes in the previous layer. This calculation process is called the feed-forward process of MLP. If there is a MLP which contains m hidden layers, its input and output dimensions are respectively equal to n_1 and n_{m+2} . The number of nodes in each hidden layer is n_2, n_3, \dots, n_{m+1} , respectively. In the feed-forward process of this MLP, each node value is calculated using the following formula:

$$x_{ij} = f(W_i X_{i-1} + b_{i-1}) \quad (2)$$

where X_{ij} represents the value of the j neuron in the i layer. W_i represents the weight vector of the j neurons in layer $i - 1$ to layer i . X_{i-1} represents the value vector of all neurons in layer $i - 1$. b_{i-1} represents the bias of the $i - 1$ layer, and f is the activation function.

MLP is a supervised learning algorithm. There is an ideal output corresponding to any input. The loss function of the ideal output and the actual value is defined as:

$$J(W, b; x, y) = \frac{1}{2} \|h_{W,b}(x) - y\|^2 \quad (4)$$

where h represents the output value, y represents the actual value, and $\|\cdot\|$ represents any distance norm.

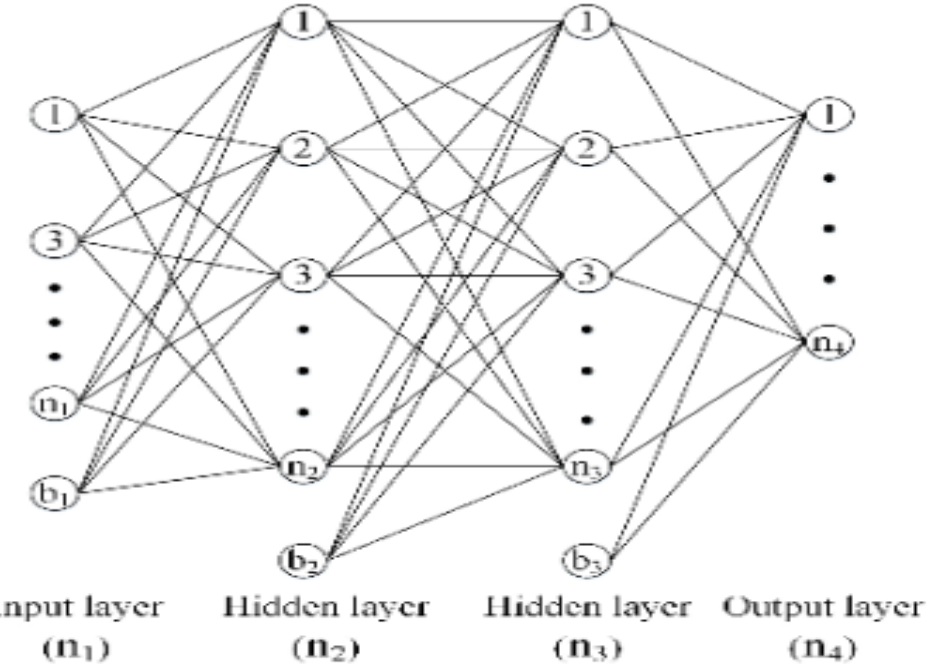


Fig. 1. Structure of MLP

we use Stochastic gradient descent (SGD) with momentum to adjust the parameters of MLP. The calculation method of gradient and SGD with momentum are described as follows:

$$\nabla W = -\frac{\partial J(W, b; x, y)}{\partial W} \quad (5)$$

$$\delta W_t = \alpha \nabla W_t + \beta \delta W_{t-1} \quad (6)$$

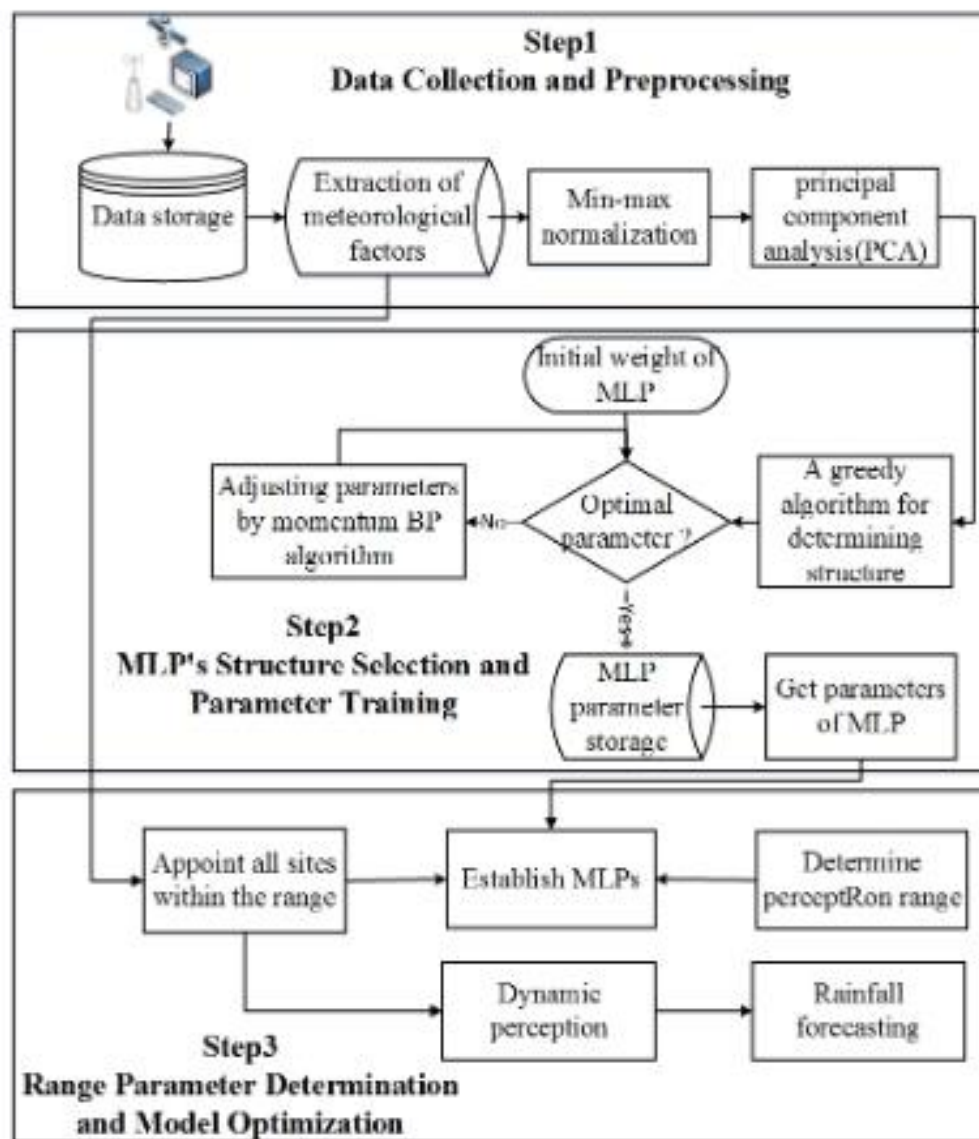


TABLE 1
A list of factors used in the model

Factor	Input value
500hPa height(X_1)	Forecast area value
500hPa temperature(X_2)	Forecast area value
500hPa temperature dew point difference(X_3)	Forecast area value
500hPa wind direction(X_4)	Forecast area value
500hPa wind speed(X_5)	Forecast area value
Total cloud amount(X_6)	Forecast area value – neighbouring area value
Surface wind speed(X_7)	Forecast area value – neighbouring area value
Surface wind direction(X_8)	Forecast area value – neighbouring area value
Surface air pressure(X_9)	Forecast area value – neighbouring area value
Surface 3 hour pressure change(X_{10})	Forecast area value – neighbouring area value
Surface temperature dew point difference(X_{11})	Forecast area value – neighbouring area value
Surface temperature(X_{12})	Forecast area value – neighbouring area value
Rainfall over past 3 hours(X_{13})	Neighbouring area value

Nowcasting based on Satellite and Radar Data

- Satellites and Radars do not directly measure atmospheric variables
- These variables need to be “derived” from their outputs
- The “derivation” is usually a complex mapping
- There may be disagreements between the measurements from different sources
- Can machine learning help us to estimate the correct values from multiple sources?

Geophysical Research Letters

RESEARCH LETTER

10.1029/2019GL084771

Key Points:

- Conventional parametric relationships between radar reflectivity Z and rain rate R are not sufficient to capture precipitation variabilities
- A hybrid deep neural network system is designed for improved space radar rainfall estimation

Supporting Information:

- Supporting Information S1


Correspondence to:

H. Chen,
haonan.chen@noaa.gov

Citation:

Chen, H., Chandrasekar, V., Tan, H., & Cifelli, R. (2019). Rainfall estimation

Rainfall Estimation From Ground Radar and TRMM Precipitation Radar Using Hybrid Deep Neural Networks

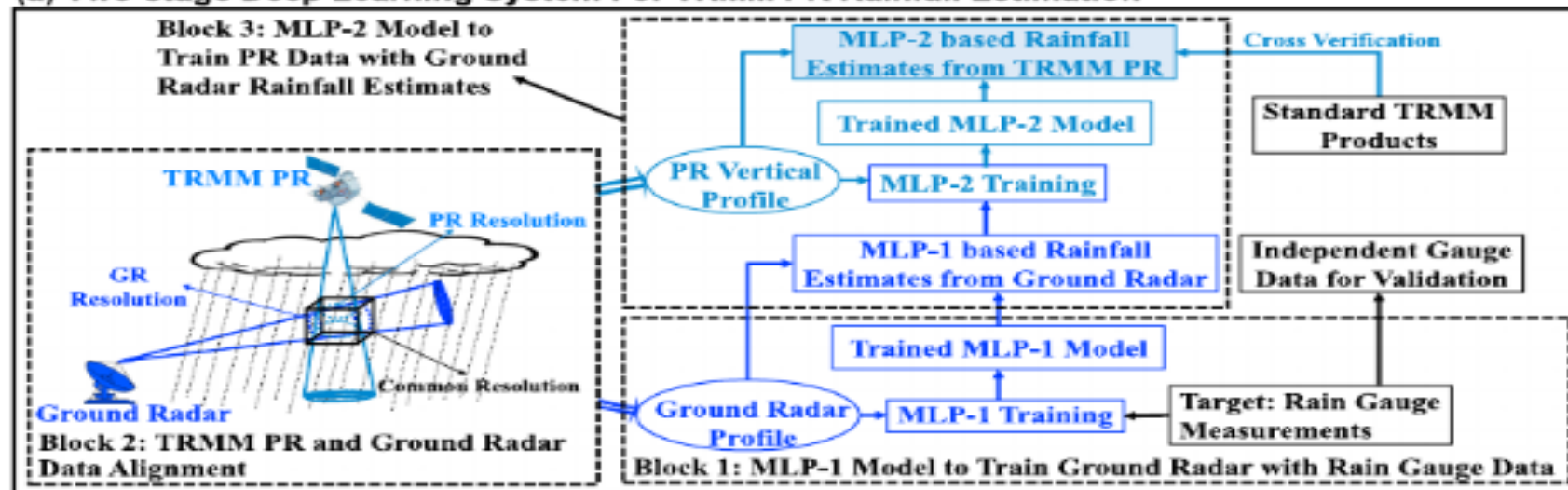
Haonan Chen^{1,2} , V. Chandrasekar¹, Haiming Tan¹, and Robert Cifelli²

¹Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA, ²NOAA/Earth System Research Laboratory, Boulder, CO, USA

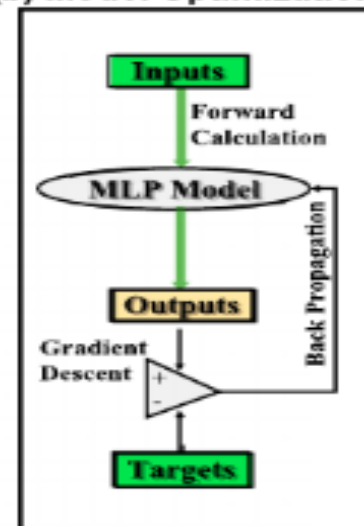
Abstract Remote sensing of precipitation is critical for regional, continental, and global water and climate research. This study develops a deep learning mechanism to link between point-wise rain gauge measurements, ground-based, and spaceborne radar reflectivity observations. Two neural network models are designed to construct a hybrid rainfall system, where the ground radar is used to bridge the scale gaps between rain gauge and satellite. The first model is trained for ground radar using rain gauge data as target labels, whereas the second model is for spaceborne Tropical Rainfall Measuring Mission (TRMM) Precipitation Radar (PR) using ground radar estimates as training labels. Data from 1 year of observations in Florida during 2009 are utilized to illustrate the application of this hybrid rainfall system. Validation using independent data in 2009, as well as 2-year comparison against the standard PR products, demonstrates the promising performance and generality of this innovative rainfall algorithm.

Plain Language Summary The Tropical Rainfall Measuring Mission (TRMM) Precipitation Radar (PR) was the first spaceborne active sensor for observing precipitation over the tropics and subtropics. During its 17 years (1997–2014) in orbit and beyond, PR has been an important tool to characterize tropical precipitation microphysics and quantify rainfall rate over the globe. Ground validation is a critical component in the development of TRMM products. However, the ground-based sensors have different characteristics from PR in terms of resolution, viewing angle, and uncertainties in the sensing environments, which are not taken into account in the operational parametric rainfall relations applied to PR measurements. This study develops a nonparametric machine learning technique for PR rainfall estimation. In the regions where substantial gauge and ground radar data are available, this approach can produce better rainfall estimates compared to the standard PR algorithm. In areas such as ocean and remote regions where no gauge or radar available, the proposed rainfall algorithm is easy to implement, and it can still produce reasonable estimates. With more and more gauges and radars being deployed and many of them become operational, this algorithm can be trained at different locations represented by different atmosphere properties to further improve the performance and generality.

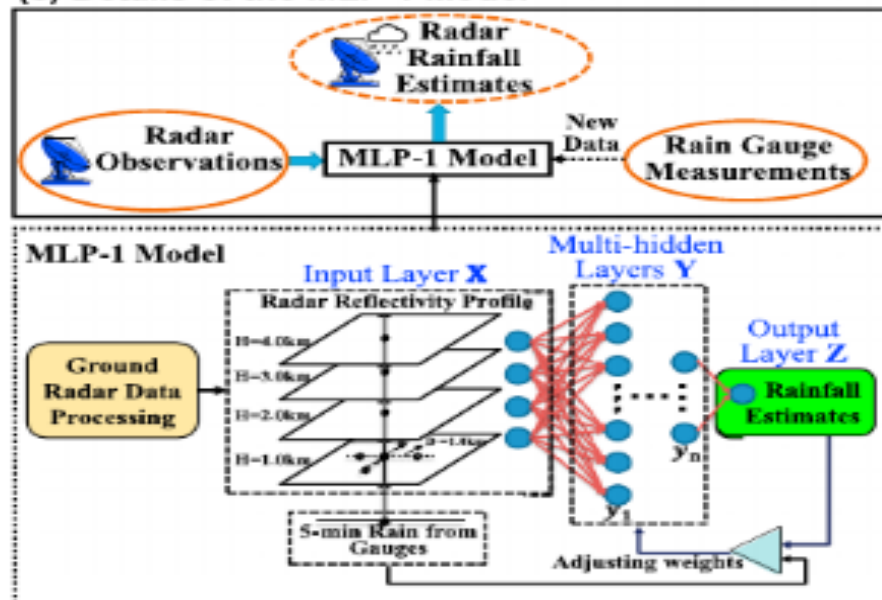
(a) Two-stage Deep Learning System For TRMM PR Rainfall Estimation



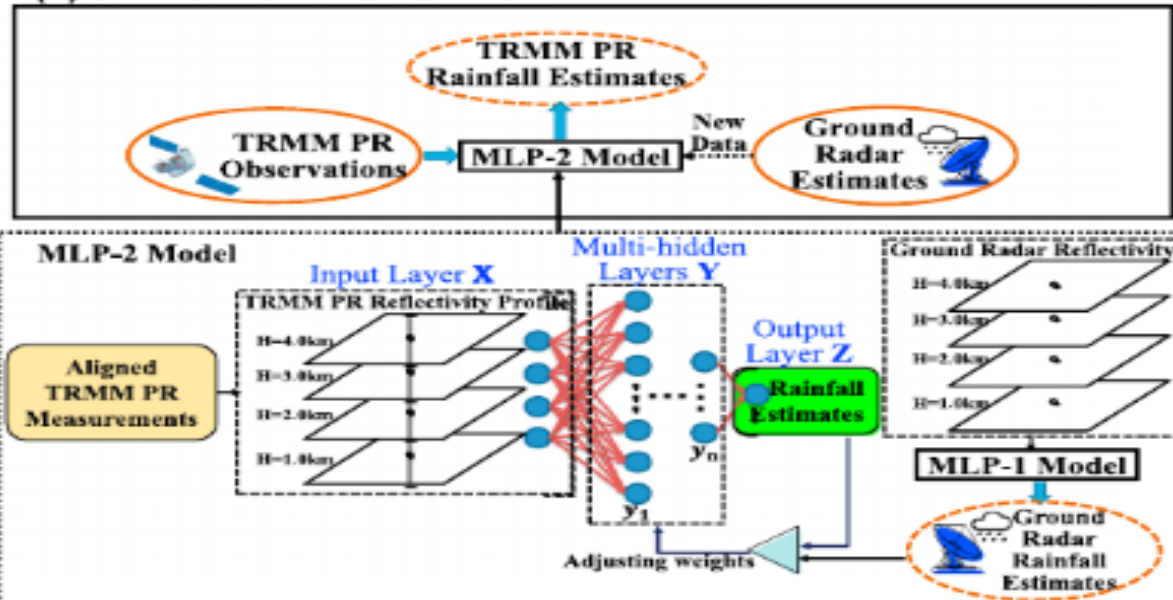
(b) Model Optimization

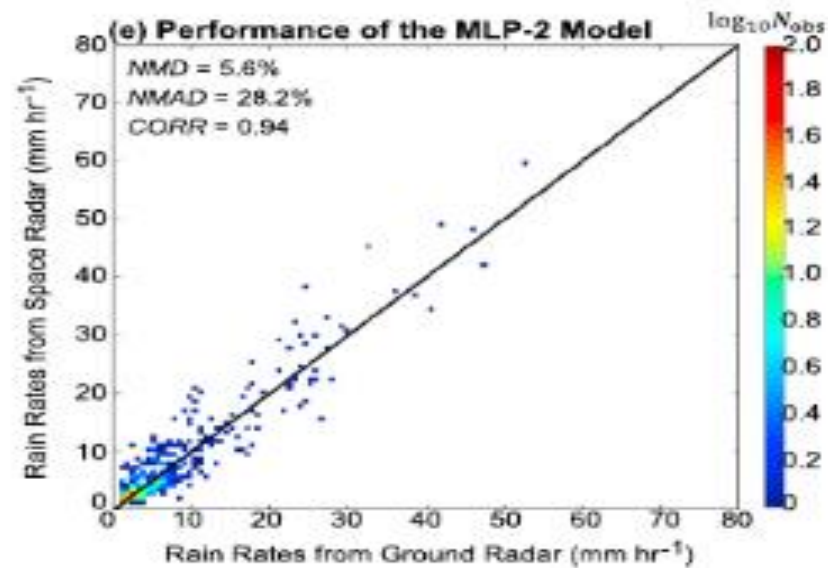
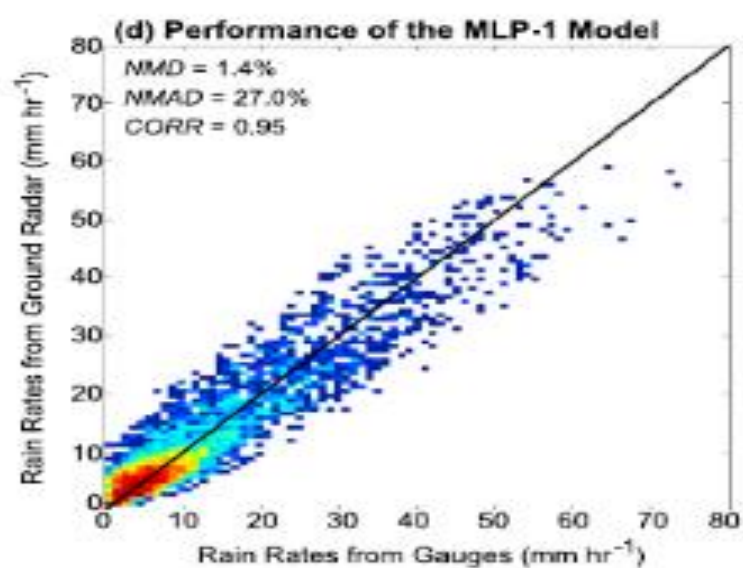
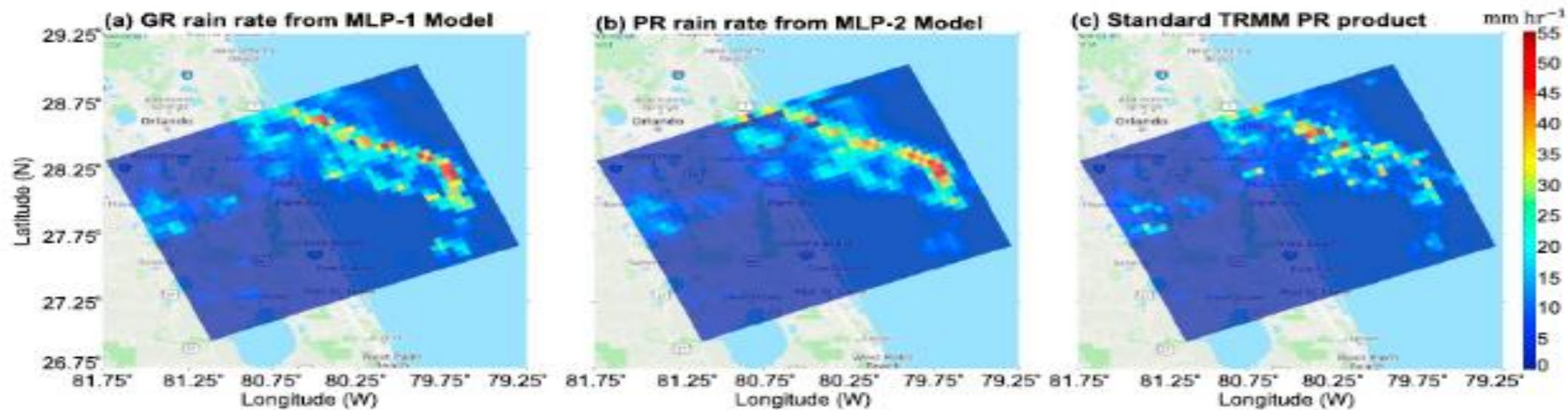


(c) Details of the MLP-1 Model



(d) Details of the MLP-2 Model





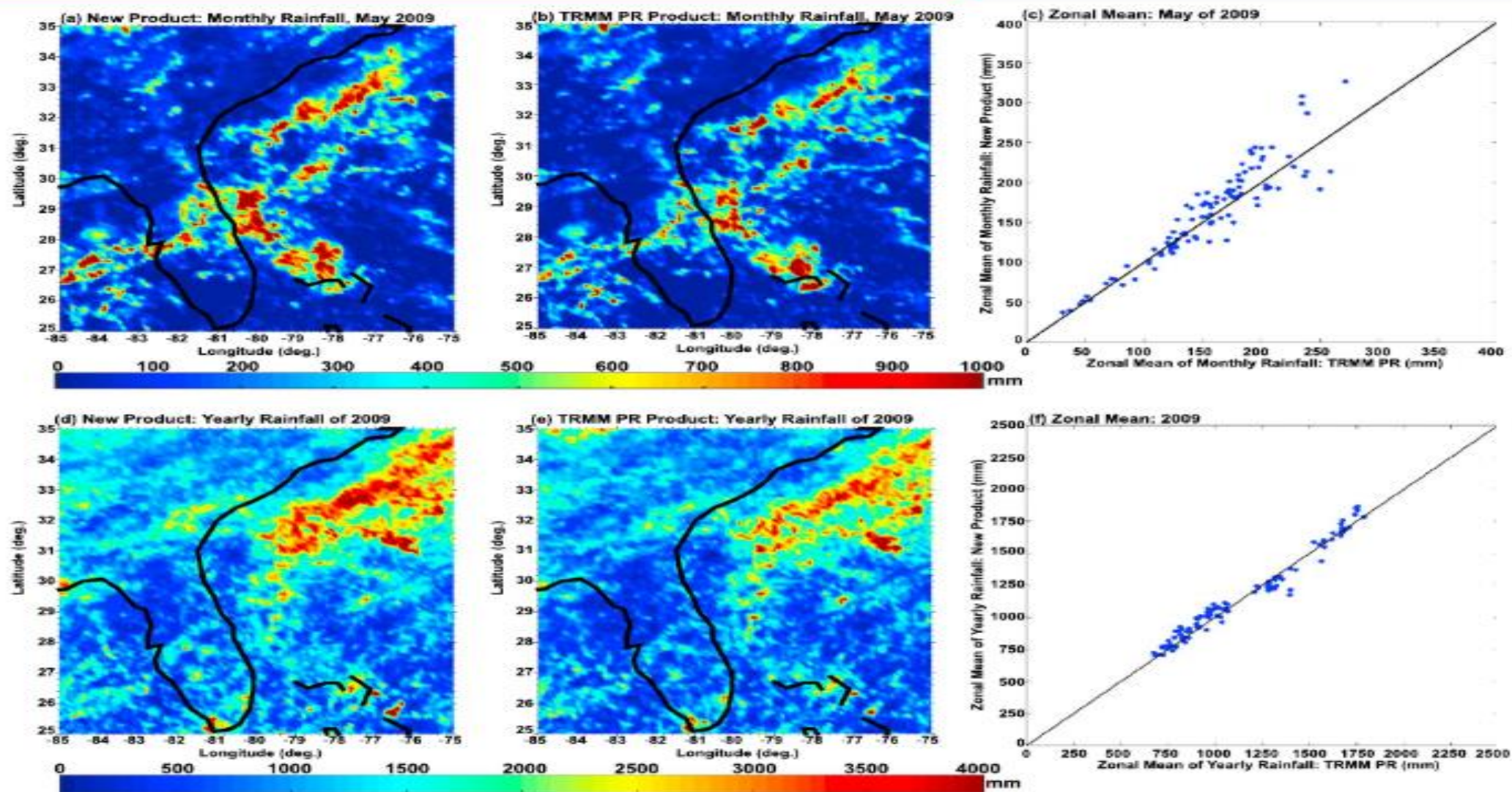


Figure 3. Sample monthly and yearly rainfall product for the region near Melbourne, FL, in 2009. Monthly rainfall map (May) derived using (a) the hybrid neural network system and (b) the standard Tropical Rainfall Measuring Mission (TRMM) Precipitation Radar (PR) product (i.e., 3A26); yearly rainfall derived using (d) the hybrid neural network system and (e) the standard TRMM PR product; (c) and (f) are the scatter plots of zonal means of rainfall estimates in (a) and (b) and (d) and (e), respectively.

Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting

Xingjian Shi Zhouong Chen Hao Wang Dit-Yan Yeung
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
{xshiab, zchenbb, hwangaz, dyyeung}@cse.ust.hk

Wai-kin Wong Wang-chun Woo
Hong Kong Observatory
Hong Kong, China
{wkwong, wcwoo}@hko.gov.hk

Abstract

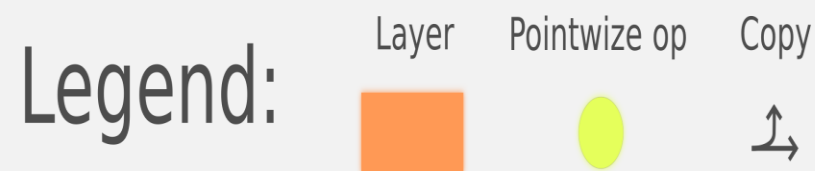
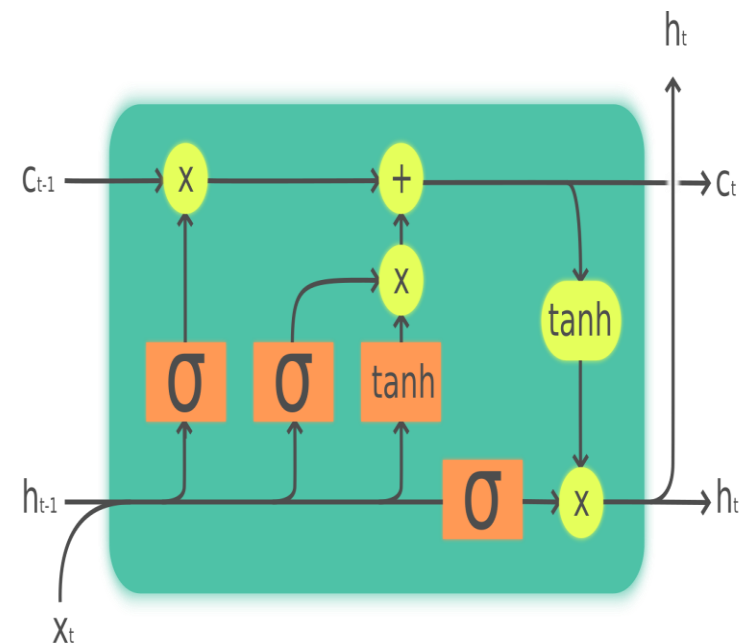
The goal of precipitation nowcasting is to predict the future rainfall intensity in a local region over a relatively short period of time. Very few previous studies have examined this crucial and challenging weather forecasting problem from the machine learning perspective. In this paper, we formulate precipitation nowcasting as a spatiotemporal sequence forecasting problem in which both the input and the prediction target are spatiotemporal sequences. By extending the *fully connected LSTM* (FC-LSTM) to have convolutional structures in both the input-to-state and state-to-state transitions, we propose the *convolutional LSTM* (ConvLSTM) and use it to build an end-to-end trainable model for the precipitation nowcasting problem. Experiments show that our ConvLSTM network captures spatiotemporal correlations better and consistently outperforms FC-LSTM and the state-of-the-art operational ROVER algorithm for precipitation nowcasting.

Suppose we observe a dynamical system over a spatial region represented by an $M \times N$ grid which consists of M rows and N columns. Inside each cell in the grid, there are P measurements which vary over time. Thus, the observation at any time can be represented by a tensor $\mathcal{X} \in \mathbb{R}^{P \times M \times N}$, where \mathbb{R} denotes the domain of the observed features. If we record the observations periodically, we will get a sequence of tensors $\hat{\mathcal{X}}_1, \hat{\mathcal{X}}_2, \dots, \hat{\mathcal{X}}_t$. The spatiotemporal sequence forecasting problem is to predict the most likely length- K sequence in the future given the previous J observations which include the current one:

$$\tilde{X}_{t+1}, \dots, \tilde{X}_{t+K} = \arg \max_{\tilde{X}_{t+1}, \dots, \tilde{X}_{t+K}} p(X_{t+1}, \dots, X_{t+K} \mid \hat{X}_{t-J+1}, \hat{X}_{t-J+2}, \dots, \hat{X}_t) \quad (1)$$

For precipitation nowcasting, the observation at every timestamp is a 2D radar echo map. If we divide the map into tiled non-overlapping patches and view the pixels inside a patch as its measurements (see Fig. 1), the nowcasting problem naturally becomes a spatiotemporal sequence forecasting problem.

We note that our spatiotemporal sequence forecasting problem is different from the one-step time series forecasting problem because the prediction target of our problem is a sequence which contains both spatial and temporal structures. Although the number of free variables in a length- K sequence can be up to $O(M^K N^K P^K)$, in practice we may exploit the structure of the space of possible predictions to reduce the dimensionality and hence make the problem tractable.



LSTM for Sequence-to-sequence prediction

- Seq2Seq: takes in a sequence as input, converts it into an “intermediate representation”, generates output sequence from it
- Input Sequence -> Encoder Network -> Intermediate Representation -> Output Network -> Output sequence
- Recurrent Neural Network: maintains a “hidden state” as memory unit, which is updated with every step in the input sequence
- Older values get updated with newer values, long sequences not handles well
- LSTM: has “hidden state” as short-term memory, and “Cell state” as long-term memory

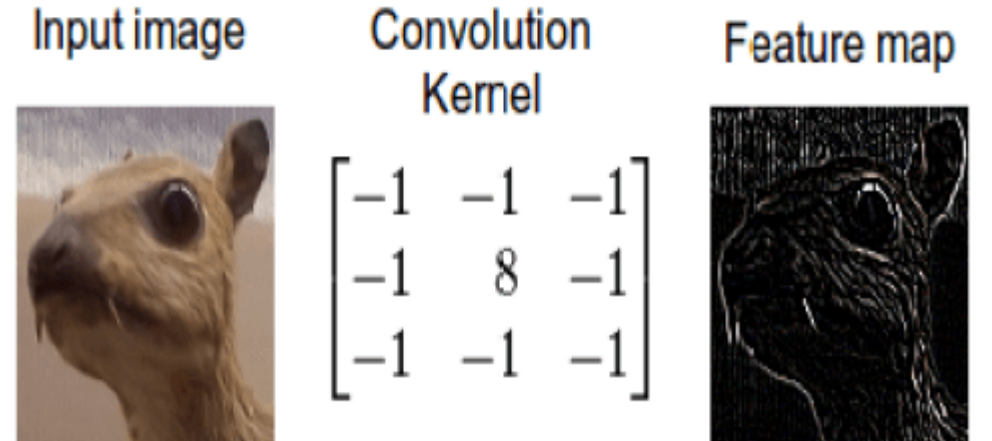
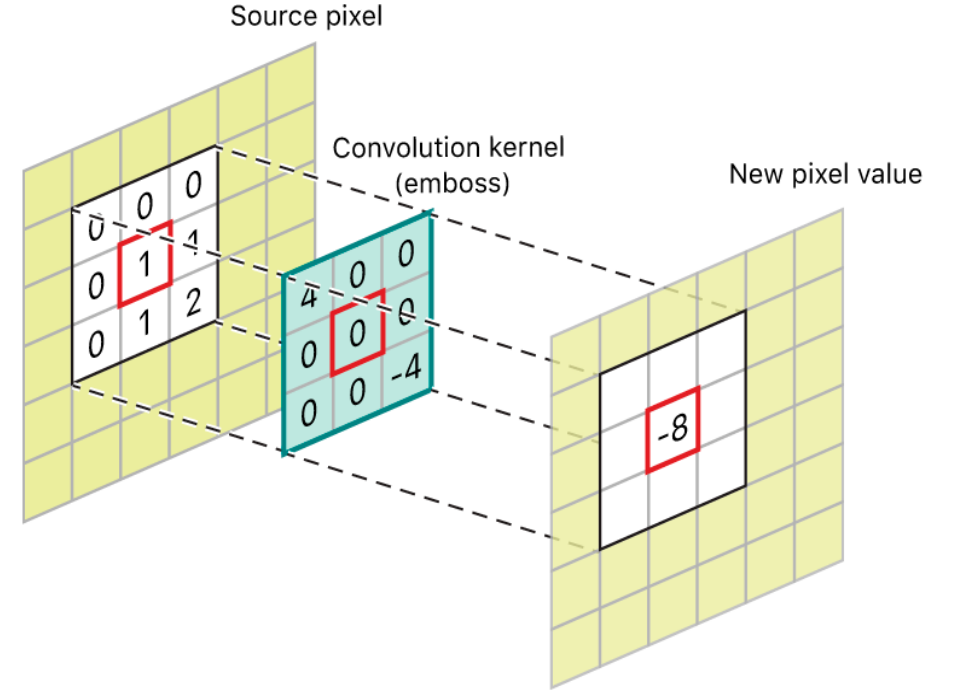
LSTM for spatio-temporal prediction

- All input, output and intermediate representations in RNNs and LSTMs: vectors
- Obtained by transformations by matrices
- Spatio-temporal sequence: each input is a matrix or tensor
- LSTM “vectorizes” the inputs, loses out spatial information!
- Solution: make the output and intermediate representations as tensors of input’s dimensions
- All the matrix-based transformations to be replaced by “Convolutions”
- LSTM -> Conv-LSTM!

3.1 Convolutional LSTM

The major drawback of FC-LSTM in handling spatiotemporal data is its usage of full connections in input-to-state and state-to-state transitions in which no spatial information is encoded. To overcome this problem, a distinguishing feature of our design is that all the inputs $\mathcal{X}_1, \dots, \mathcal{X}_t$, cell outputs $\mathcal{C}_1, \dots, \mathcal{C}_t$, hidden states $\mathcal{H}_1, \dots, \mathcal{H}_t$, and gates i_t, f_t, o_t of the ConvLSTM are 3D tensors whose last two dimensions are spatial dimensions (rows and columns). To get a better picture of the inputs and states, we may imagine them as vectors standing on a spatial grid. The ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors. This can easily be achieved by using a convolution operator in the state-to-state and input-to-state transitions (see Fig. 2). The key equations of ConvLSTM are shown in (3) below, where ‘ $*$ ’ denotes the convolution operator and ‘ \circ ’, as before, denotes the Hadamard product:

$$\begin{aligned}
 i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
 f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \\
 \mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\
 o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
 \mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
 \end{aligned} \tag{3}$$



Sequence Prediction with ConvLSTM

- (Hidden State, Cell State): tensors that are updated at each input step
- Two or more LSTMs may be stacked together to encode long sequences
- Multiple LSTMS layers can “remember” more values
- (H_T , C_T) from each LSTM layer: intermediate representation! (T is the length of input sequence)
- The intermediate representation is copied into the decoder for output generation
- Outputs generated one step-by- one step
- Output of one step becomes input to next step

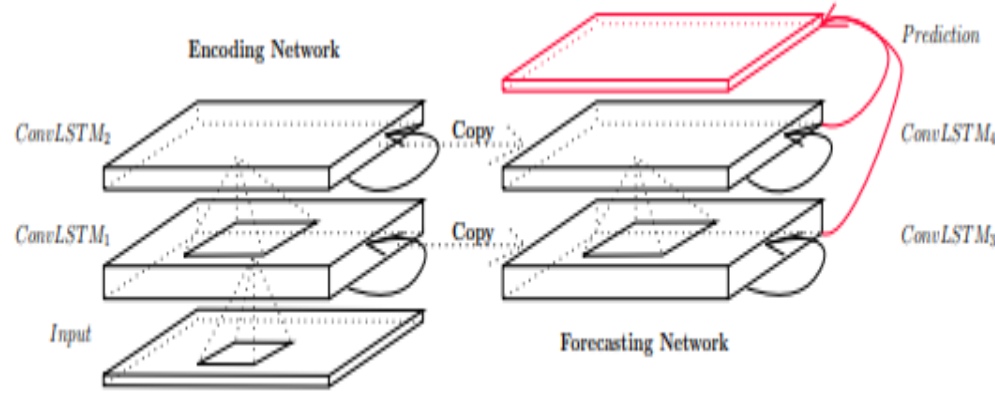


Figure 3: Encoding-forecasting ConvLSTM network for precipitation nowcasting

state to give the final prediction:

$$\begin{aligned}
 \tilde{\mathcal{X}}_{t+1}, \dots, \tilde{\mathcal{X}}_{t+K} &= \arg \max_{\mathcal{X}_{t+1}, \dots, \mathcal{X}_{t+K}} p(\mathcal{X}_{t+1}, \dots, \mathcal{X}_{t+K} | \hat{\mathcal{X}}_{t-J+1}, \hat{\mathcal{X}}_{t-J+2}, \dots, \hat{\mathcal{X}}_t) \\
 &\approx \arg \max_{\mathcal{X}_{t+1}, \dots, \mathcal{X}_{t+K}} p(\mathcal{X}_{t+1}, \dots, \mathcal{X}_{t+K} | f_{\text{encoding}}(\hat{\mathcal{X}}_{t-J+1}, \hat{\mathcal{X}}_{t-J+2}, \dots, \hat{\mathcal{X}}_t)) \quad (4) \\
 &\approx g_{\text{forecasting}}(f_{\text{encoding}}(\hat{\mathcal{X}}_{t-J+1}, \hat{\mathcal{X}}_{t-J+2}, \dots, \hat{\mathcal{X}}_t))
 \end{aligned}$$

Table 2: Comparison of the average scores of different models over 15 prediction steps.

Model	Rainfall-MSE	CSI	FAR	POD	Correlation
ConvLSTM(3x3)-3x3-64-3x3-64	1.420	0.577	0.195	0.660	0.908
Rover1	1.712	0.516	0.308	0.636	0.843
Rover2	1.684	0.522	0.301	0.642	0.850
Rover3	1.685	0.522	0.301	0.642	0.849
FC-LSTM-2000-2000	1.865	0.286	0.335	0.351	0.774

4.2 Radar Echo Dataset

The radar echo dataset used in this paper is a subset of the three-year weather radar intensities collected in Hong Kong from 2011 to 2013. Since not every day is rainy and our nowcasting target is precipitation, we select the top 97 rainy days to form our dataset. For preprocessing, we first transform the intensity values Z to gray-level pixels P by setting $P = \frac{Z - \min\{Z\}}{\max\{Z\} - \min\{Z\}}$ and crop the radar maps in the central 330×330 region. After that, we apply the disk filter⁵ with radius 10 and resize the radar maps to 100×100 . To reduce the noise caused by measuring instruments, we further remove the pixel values of some noisy regions which are determined by applying K -means clustering to the monthly pixel average. The weather radar data is recorded every 6 minutes, so there are 240 frames per day. To get disjoint subsets for training, testing and validation, we partition each daily sequence into 40 non-overlapping frame blocks and randomly assign 4 blocks for training, 1 block for testing and 1 block for validation. The data instances are sliced from these blocks using a 20-frame-wide sliding window. Thus our radar echo dataset contains 8148 training sequences, 2037 testing sequences and 2037 validation sequences and all the sequences are 20 frames long (5 for the input and 15 for the prediction). Although the training and testing instances sliced from the same day may have some dependencies, this splitting strategy is still reasonable because in real-life nowcasting, we do have access to all previous data, including data from the same day, which allows us to apply online fine-tuning of the model. Such data splitting may be viewed as an approximation of the real-life “fine-tuning-enabled” setting for this application.

We set the patch size to 2 and train a 2-layer ConvLSTM network with each layer containing 64 hidden states and 3×3 kernels. For the ROVER algorithm, we tune the parameters of the optical