15

Deep Learning and the Weather Forecasting Problem: Precipitation Nowcasting

Zhihan Gao, Xingjian Shi, Hao Wang, Dit-Yan Yeung, Wang-chun Woo, and Wai-Kin Wong

15.1 Introduction

Precipitation nowcasting refers to the forecasting of rainfall and other types of precipitation up to 6 hours ahead (as defined by the World Meteorological Organization)[1]. Since rainfall can be localized and highly changeable, users of precipitation nowcast typically demand to know the exact time, location and intensity of rainfall. It is therefore necessary to make very high resolution, both spatially and temporally, precipitation nowcast products in a timely manner, typically in the order of minutes. The most important use of precipitation nowcast is to support the operations of rainstorm warning systems managed by meteorological services around the world. Rainstorm warning systems provide early alerts to the public, disaster risk reduction agencies, government departments in particular those related to public security and works, as well as managers of infrastructures and facilities. Upon the issuance of rainstorm warnings, these parties take actions according to their own standard operating procedures with a view to saving lives and protecting properties. It has tremendous impact on various areas from aviation service and public safety to people's daily life.

For example, commercial airlines rely on precipitation nowcasting to predict extreme weather events and ensures flight safety. On land, heavy rainfall can severely affect the road conditions and increases the risk of traffic accidents, which can be avoided with the help of precipitation nowcasting. For local businesses, the number of customers and their feedback about a restaurant are largely related to the weather (Bujisic et al. 2019), especially the rain rate. Thus, accurate and timely prediction of rainfall helps restaurants predict and adjust their sales strategies. Therefore, the past years have seen an ever-growing need for real-time, large-scale, timely and fine-grained precipitation nowcasting (Xingjian et al. 2015; Shi et al. 2017; Lebedev et al. 2019b; Agrawal et al. 2019). Due to the inherent complexities of the atmosphere and relevant dynamical processes, the problem imposes new challenge to the meteorological community (Sun et al. 2014).

Traditionally, precipitation nowcasting is approached by either *Optical Flow* (OF)-based methods (Li et al. 2000; Reyniers 2008) or numerical methods (Weisman et al. 2008; Sun et al. 2014; Benjamin et al. 2016). OF-based methods first estimate the flow field, which

1 See WMO-No.1198, accessible from https://library.wmo.int/doc_num.php?explnum_id=3795

Deep Learning for the Earth Sciences: A Comprehensive Approach to Remote Sensing, Climate Science, and Geosciences, First Edition. Edited by Gustau Camps-Valls, Devis Tuia, Xiao Xiang Zhu, and Markus Reichstein.
© 2021 John Wiley & Sons Ltd. Published 2021 by John Wiley & Sons Ltd.

represents the convective motion of the precipitation, with the observed weather data (e.g., the *Constant Altitude Plan Position Indicator* (CAPPI) radar echo maps (Douglas 1990)) and then use the flow field for extrapolation (Woo and Wong 2017). The numerical methods build mathematical models of the atmosphere on top of the physical principles such as the dynamic and thermodynamic laws. Future rainfall intensities are predicted by numerically solving partial differential equations within the mathematical models. However, both approaches have deficiencies that limit their success. The OF-based methods attempt to identify the convective motion of the cloud, but they fail to represent cloud initiation or decay and lack the ability of expressing strong nonlinear dynamics. In addition, the flow field estimation step and the radar echo extrapolation step are separated, making it challenging to determine the best model parameters. Numerical methods can provide reliable forecast but require meticulous simulation of the physical equations. The inference time of numerical models usually take several hours and they are therefore not suitable for generating fine-grained predictions required by precipitation nowcasting.

Recently, a new approach, *deep learning for precipitation nowcasting*, has emerged in the area and shown promising results. Shi et al. (Xingjian et al. 2015) first formulated precipitation nowcasting as a spatiotemporal sequence forecasting problem and proposed a DL-based model, dubbed *Convolutional Long Short-Term Memory* (ConvLSTM), to directly predict the future rainfall intensities based on the past radar echo maps. The model is learned end-to-end with a large amount of historical weather data and performs substantially better than the OF-based algorithm in the operational *Short-range Warning of Intense Rainstorms in Localized Systems* (SWIRLS) developed by the *Hong Kong Observatory* (HKO) (Li et al. 2000; Woo and Wong 2017). After this seminal work, researchers start to explore DL-based methods for precipitation nowcasting and have built models with state-of-the-art performance (Hernández et al. 2016; Shi et al. 2017; Qiu et al. 2017; Tran and Song 2019; Lebedev et al. 2019b; Chen et al. 2019; Agrawal et al. 2019). In essence, precipitation nowcasting is well-suited for DL for three reasons. Firstly, the problem satisfies the big data requirement of DL. Numerous amount of weather data are generated on a daily basis and can be used to train the nowcasting model. For example, in *National Oceanic and Atmospheric Administration* (NOAA), tens of terabytes of data are generated in a single day (Szura 2018). Secondly, DL is suitable for modeling complex dynamical systems (Goodfellow et al. 2016); a single-hidden-layer *Multi-Layer Perceptron* (MLP), which is the most basic form of DL models, is a universal functional approximator (Csáji et al. 2001). Thirdly, the inference speed of DL models is faster than numerical methods (Agrawal et al. 2019). Moreover, in the inference stage, we can dynamically update the DL model with the newly observed data (Shi et al. 2017), making the model more adaptive to the emerging weather patterns.

In this chapter, we introduce the current progress of DL-based methods for precipitation nowcasting. In section 15.2, we describe how to mathematically formulate precipitation nowcasting as a spatiotemporal sequence forecasting problem. In section 15.3, we review the high-level strategies for constructing and learning DL models for precipitation nowcasting; because precipitation nowcasting requires predicting rainfall intensities for multiple timestamps ahead, we introduce various strategies to learn such a multi-step forecasting model. In section 15.4.1 and section 15.4.2, we introduce the DL models in two

categories: *Feed-forward Neural Network* (FNN)-based models and *Recurrent Neural Network* (RNN)-based models. In section 15.5, we describe the first systematic benchmark of the DL models for precipitation nowcasting, the HKO-7 benchmark. We conclude this chapter and discuss the potential future works along this area in section 15.6.

15.2 Formulation

Precipitation nowcasting can be formulated as a spatiotemporal sequence forecasting problem. Suppose that the meteorological system is defined over an $M \times N$ grid, in which there are M rows and N columns. Within each cell (i, j) of the grid, there are D measurements that vary over time. By taking snapshots of the system at timestamps $t_1, t_2, ..., t_T$, we get a spatiotemporal sequence that can be denoted as a sequence of tensors $\mathbf{X}_{t_1:t_T} = [\mathbf{X}_{t_1}, \mathbf{X}_{t_2}, \dots, \mathbf{X}_{t_T}]$. Here, $\mathbf{X}_{t_i} \in \mathcal{R}^{D \times M \times N}$ is the observed meteorological data at timestamp t_i. In most DL models for precipitation nowcasting, the meteorological observations $\mathbf{X}_{t_i}$s are $2D$ CAPPI radar echo maps (Xingjian et al. 2015; Shi et al. 2017), satellite images (Lebedev et al. 2019b), or data from another *Quantitative Precipitation Estimation* (QPE) product (Agrawal et al. 2019; Zhang et al. 2016a). Thus, in most cases, the grid is regular and each pixel covers a region in the geographical map, e.g., a 1 km×1 km local area. Some works (Hernández et al. 2016; Qiu et al. 2017) mainly explore the impact of multi-modal meteorological data without considering the spatial correlations. In this case, $M = N = 1$ and the spatiotemporal sequence forecasting problem degenerates to sequence forecasting problem, in which $\mathbf{X}_{t_i}$ degenerates to a vector $\boldsymbol{x}_{t_i} \in \mathcal{R}^D$. In addition, in most scenarios, the time difference between two consecutive snapshots, i.e. $t_{i+1} - t_i$, is always the same. Thus, we are able to simplify the definition and denote $\mathbf{X}_{t_i}$ as $\mathbf{X}_i$.

The spatiotemporal sequence forecasting problem is to predict the most likely length-L sequence in the future given the previous J observations including the current one (Xingjian et al. 2015). The mathematical definition is given in Equation 15.1, in which $\tilde{\mathbf{X}}_{t+1:t+L}$ are the predictions, $\mathbf{X}_{t-J+1:t}$ are the observations, and $p(\mathbf{X}_{t+1:t+L} \mid \mathbf{X}_{t-J+1:t})$ is the model. In the terminology of precipitation nowcasting, the goal is to use the previously observed sequence to predict short term rainfalls of a local region (e.g., Hong Kong, Shanghai, New York, or Tokyo) in the future. In most nowcasting systems, radar echo map is the mainstay of the meteorological observations due to its high spatial and temporal resolutions. The radar maps are usually taken from the weather radar every 6–10 minutes and predictions are given for the following 0–6 hours. If we record one radar frame every six minutes, the task is to predict for 0–60 frames ahead:

$$\tilde{\mathbf{X}}_{t+1:t+L} = \underset{\mathbf{X}_{t+1:t+L}}{\operatorname{argmax}}\, p(\mathbf{X}_{t+1:t+L} \mid \mathbf{X}_{t-J+1:t}). \tag{15.1}$$

The spatiotemporal sequence forecasting problem is different from the conventional multi-variate time series forecasting problem because the prediction target of our problem is a sequence that contains both spatial and temporal structures. Because the number of possible sequences grows exponentially with respect to both the spatial and temporal dimensionality, we have to, in practice, exploit the structure of the spatiotemporal space to reduce the dimensionality and hence make the problem tractable.

15.3 Learning Strategies

Precipitation nowcasting is intrinsically a multi-step forecasting problem. Learning a model for multi-step forecasting is challenging because the elements in the predicted sequence $\hat{\mathbf{X}}_{t+1:t+L}$ are not independent and identically distributed (i.i.d.). Nevertheless, predicting the rainfall for multiple timestamps ahead is a crucial requirement of precipitation nowcasting and DL-based methods adopt different ways to solve the issue. In this section, we introduce the learning strategies for multi-step forecasting. We first explain and compare two basic strategies called *Iterative Multi-step Estimation* (IME) and *Direct Multi-step Estimation* (DME) (Chevillon 2007) and then introduce one extension called *Scheduled Sampling* (SS) that bridges the gap between IME and DME.

Iterative Multi-Step Estimation The IME strategy trains a single-step forecasting model and iteratively feeds the generated samples to the forecaster to get multi-step-ahead predictions. The IME model can either be deterministic or probabilistic. Here, we denote the model as $p(\mathbf{X}_{t+1} \mid \mathbf{X}_{1:t}; \boldsymbol{\theta})$ to cover both cases, in which the deterministic model has a delta distribution. In the terminology of precipitation nowcasting, $\mathbf{X}_{1:t}$ is the sequence of past weather data, $\mathbf{X}_{t+1}$ is the rainfall intensity that the model will predict at timestamp $t+1$, and $\boldsymbol{\theta}$ is the parameter of the model. To train the model, we factorize the distribution $p(\mathbf{X}_{t+1:t+L} \mid \mathbf{X}_{t-J+1:t})$ as $\prod_{i=1}^{L} p(\mathbf{X}_{t+i} \mid \mathbf{X}_{t-J+1:t+i-1}; \boldsymbol{\theta})$. The optimal parameter $\boldsymbol{\theta}^\star$ can be estimated by maximizing the likelihood:

$$\boldsymbol{\theta}^\star = \arg\max_{\phi} \mathbb{E}_{\hat{p}_{\text{data}}} \left[\sum_{i=1}^{L} \log p(\mathbf{X}_{t+i} \mid \mathbf{X}_{t-J+1:t+i-1}; \boldsymbol{\theta}) \right]. \tag{15.2}$$

There are two advantages of the IME approach: (i) The objective function in equation 15.2 is easy to train because it only requires optimizing for the one-step-ahead forecasting error and (ii) we can predict for an arbitrary horizons in the future by recursively applying the basic forecaster. However, there is an intrinsic discrepancy between training and testing in IME. In the training phase, we use the ground-truths from $t+1$ to $t+i-1$ to predict the regional rainfall at timestamp $t+i$, which is also known as teacher-forcing (Goodfellow et al. 2016). While in the testing phase, we feed the model predictions instead of the ground-truths back to the forecaster. This makes the model prone to accumulative errors in the forecasting process (Bengio et al. 2015). Usually, the optimal forecaster for timestamp $t+i$, which is obtained by maximizing $\mathbb{E}_{\hat{p}_{\text{data}}} \left[\log p(\mathbf{X}_{t+i} \mid \mathbf{X}_{t-J+1:t}; \boldsymbol{\theta}) \right]$, is not the same as recursively applying the optimal one-step-ahead forecaster when the model is nonlinear. This is because the forecasting error at earlier timestamps will propagate to later timestamps (Lin and Granger 1994).

Direct Multi-Step Estimation The main motivation behind DME is to avoid the error drifting problem in IME by directly minimizing the long-term prediction error. Instead of training a single model, DME trains a different model $p(\mathbf{X}_{t+i} \mid \mathbf{X}_{t-J+1:t}; \boldsymbol{\theta}_i)$ for each forecasting horizon i, in which $\boldsymbol{\theta}_i$ is the parameter. There can thus be L models in the DME approach. The set of optimal parameters $\{\boldsymbol{\theta}_1^\star, \ldots, \boldsymbol{\theta}_L^\star\}$ can be estimated from the following optimization

problem:

$$\theta_1^\star, \dots, \theta_L^\star = \arg\max_{\theta_1,\dots,\theta_L} \mathbb{E}_{\hat{p}_{\text{data}}} \left[\sum_{i=1}^{L} \log p(\mathbf{X}_{t+i} \mid \mathbf{X}_{t-J+1:t}; \theta_i) \right] \tag{15.3}$$

To disentangle the model size from the number of forecasting steps L, we can also construct $p(\mathbf{X}_{t+i} \mid \mathbf{X}_{t-J+1:t}; \theta_i)$ by recursively applying the single-step forecaster $p(\mathbf{X}_{t+1} \mid \mathbf{X}_{1:t}; \theta)$. In this case, the model parameters $\{\theta_1, ..., \theta_L\}$ are shared. For example, when the single-step forecasting model is deterministic and predicts $\tilde{\mathbf{X}}_{t+1}$ as $m(\mathbf{X}_{1:t}; \theta)$, we can obtain the second step prediction by feeding in the predicted rainfall intensity, i.e., $\tilde{\mathbf{X}}_{t+2} = m\left(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_t, m(\mathbf{X}_{1:t}; \theta); \theta\right)$. By repeating the process for L times, we obtain the predictions $\tilde{\mathbf{X}}_{t+1:t+L}$. The optimal parameter $\theta^\star$ can be estimated by minimizing the distance between the prediction and the ground-truth, i.e., $\theta^\star = \arg\min_\theta \mathbb{E}_{\hat{p}_{\text{data}}} d(\tilde{\mathbf{X}}_{t+1:t+L}, \mathbf{X}_{t+1;t+L})$, in which $d(\cdot, \cdot)$ is a distance function. We need to emphasize here that the aforementioned objective function directly optimizes the multi-step-ahead forecasting error and is different from Equation 15.2, which only minimizes the one-step-ahead forecasting error.

Scheduled Sampling According to Chevillon (2007), DME leads to more accurate predictions when (i) the model is misspecified, (ii) the sequences are non-stationary, or (iii) the training set is too small. However, DME is more computationally expensive than IME. For DME, if the θ_hs are not shared, we need to store and train L models. If the θ_hs are shared, we need to recursively apply the basic forecasting model for $O(L)$ steps (Chevillon 2007; Bengio et al. 2015; Lamb et al. 2016). Both cases require larger memory storage or longer running time than solving the IME objective. Overall, IME is easier to train but less accurate for multi-step forecasting, while DME is more difficult to train but more accurate.

Schedules sampling (SS) Bengio et al. (2015) tries to bridge the gap between IME and DME. The idea of SS is to first train the model with IME and then gradually replace the ground-truths in the objective function with samples generated by the model itself. When all ground-truth samples are replaced with model-generated samples, the training objective falls back into the DME objective. The generation process of SS is described in Equation 15.4:

$$\begin{aligned} &\forall 1 \le i \le L, \\ &\tilde{\mathbf{X}}_{t+i} \sim p(\mathbf{X}_{t+i} \mid \hat{\mathbf{X}}_{t-J+1:t}, \overline{\mathbf{X}}_{t+1:t+i-1}; \theta), \\ &\overline{\mathbf{X}}_{t+i} = (1 - \tau_{t+i})\hat{\mathbf{X}}_{t+i} + \tau_{t+i}\tilde{\mathbf{X}}_{t+i}, \\ &\tau_{t+i} \sim \text{Binomial}(1, \epsilon_k). \end{aligned} \tag{15.4}$$

Here, $\tilde{\mathbf{X}}_{t+i}$ and $\hat{\mathbf{X}}_{t+i}$ are correspondingly the generated sample and the ground-truth at timestamp $t + i$. $p(\mathbf{X}_{t+i} \mid \hat{\mathbf{X}}_{t-J+1:t}, \overline{\mathbf{X}}_{t+1:t+i-1}; \theta)$ is the basic single-step forecasting model. Meanwhile, τ_{t+h} is generated from a binomial distribution and controls whether to use the ground-truth or the generated sample. ϵ_k is the probability of choosing the ground-truth at the kth iteration. In the training phase, SS minimizes the distance between $\tilde{\mathbf{X}}_{t+1:t+L}$ and $\hat{\mathbf{X}}_{t+1:t+L}$. In the testing phase, τ_{t+i}s are fixed to 0, meaning that the model-generated samples are always used.

SS lies in the mid-ground between IME and DME. If ϵ_k equals to 1, the ground-truths are always chosen, and the objective function will be the same as in the IME strategy. If ϵ_k is 0, the generated samples are always chosen, and the optimization objective will be the same as in the DME strategy. In practice (Bengio et al. 2015; Wang et al. 2019b), ϵ_k is gradually decayed during the training phase to make the optimization objective shift smoothly from IME to DME, which is a type of *curriculum learning* (Bengio et al. 2009).

When applied for precipitation nowcasting, existing DL models adopt either of these three learning strategies. We will introduce the detailed architectures of these two types of models in section 15.4.1 and section 15.4.2 and give an overview of the learning strategy that each model uses in section 15.6.

15.4 Models

15.4.1 FNN-based Models

FNN refers to deep learning models that construct the mapping $\mathbf{Y} = f(\mathbf{X}; \theta)$ by stacking various basic blocks such as the *Fully-Connected* (FC) layer, the convolution layer, the deconvolution layer, and the activation layer. Common types of FNNs include *Multilayer Perceptron* (MLP) (Rosenblatt 1962; Goodfellow et al. 2016), which stacks multiple FC layers and nonlinear activations and *Convolutional Neural Network* (CNN) (Krizhevsky et al. 2012), which stacks multiple convolution layers, pooling layers, deconvolution layers, FC layers, activation layers, normalization layers (Ioffe and Szegedy 2015; Wu and He 2018) and other transformations. The parameters of FNN are usually estimated by minimizing the loss function plus some regularization terms, i.e., $\theta^\star = \arg\min_\theta \mathbb{E}_{\hat{p}_{\text{data}}} \left[l(\mathbf{Y}, f(\mathbf{X}; \theta)) \right] + \Omega(\theta)$ where $l(\cdot, \cdot)$ is the loss function and $\Omega(\cdot)$ is the regularization function such as the L_2 or L_1 loss (Goodfellow et al. 2016). Usually, the optimization problem is solved via stochastic-gradient-based methods (Goodfellow et al. 2016), in which the gradient is computed by backpropagation (Nocedal and Wright 2006).

The convolution layer takes advantage of the translational invariance property of image data. The convolution layer computes the output by scanning over the input and applying the same set of linear filters. Although the input can have an arbitrary dimensionality (Tran et al. 2015), we mainly focus on 2D convolution, since for precipitation nowcasting, the convolution layer is mainly used for extracting the spatial correlation in meteorological images. For input $\mathbf{X} \in \mathcal{R}^{C_i \times H_i \times W_i}$, the output of the convolution layer $\mathbf{H} \in \mathcal{R}^{C_o \times H_o \times W_o}$, which is also known as feature map. Ayzel et al. (2019) proposed DozhdyaNet for precipitation nowcasting. DozhdyaNet consists of only convolution layers and is a type of all convolutional network (Springenberg et al. 2014). The input of the model is a sequence of radar images $\mathbf{X}_{t-J+1:t}$. To facilitate the 2D convolution layer to deal with the sequence of 3D tensors, the author concatenates all frames along the temporal dimension and treats them as different channels:

$$\mathbf{X}_{in} = \text{concat}(\mathbf{X}_{t-J+1}, \mathbf{X}_{t-J+2}, ...\mathbf{X}_t), \tag{15.5}$$

where $\mathbf{X}_{in} \in \mathcal{R}^{C_i' \times H_i \times W_i}$ is the network input and the number of channels C_i' is the product of the number of observations within each local grid and the input sequence length, i.e.,

$C_i' = C_i \times J$. Notice that in this manner, the channel C_i' is determined by the input length, hence unlike the RNN-based models, which will be explained in detail in section 15.4.2, the input length must be fixed for the FNN-base models. The radar image at the next timestamp $\mathbf{X}_{t+1}$ is predicted by feeding the preprocessed input sequence $\mathbf{X}_{in}$ into the FNN: $\mathbf{X}_{t+1} = f(\mathbf{X}_{in}; \theta)$. Also, to predict for multiple steps ahead, the author adopted the IME strategy by feeding the predicted radar image back to the network in the inference phase. Also, the author compared different transformation techniques for preprocessing the 2D radar images and used two radar images taken with the 5min interval as the input to predict the next radar image.

Agrawal et al. (2019) also concatenates the input images in the temporal dimension and used the U-Net (Ronneberger et al. 2015a) architecture for prediction. U-Net combines down-sampling, up-sampling and skip connections to learn better hidden representations. Figure 15.1 illustrates how these building blocks are organized. The iterative down-sampling part extracts more global and more abstract representations and the up-sampling part gradually refines the representation and adds the finer details to the generated output. The skip connection helps preserve high-resolution details and facilitates gradient backpropagation. The author used QPE data from the *Multi-Radar Multi-Sensor* (MRMS) system (Zhang et al. 2016a) for training and testing the model. The input is a sequence of radar images taken with 2min interval for one hour and the output is the sequence of radar images for the next several hours. Experiments show that the U-Net based DL model outperforms OF-based model and the HRRR model by NOAA (Benjamin et al. 2016).

Klein et al. Klein et al. (2015) designed the dynamic convolution layer to replace the conventional convolution layer. Instead of using a data-independent filter, the dynamic

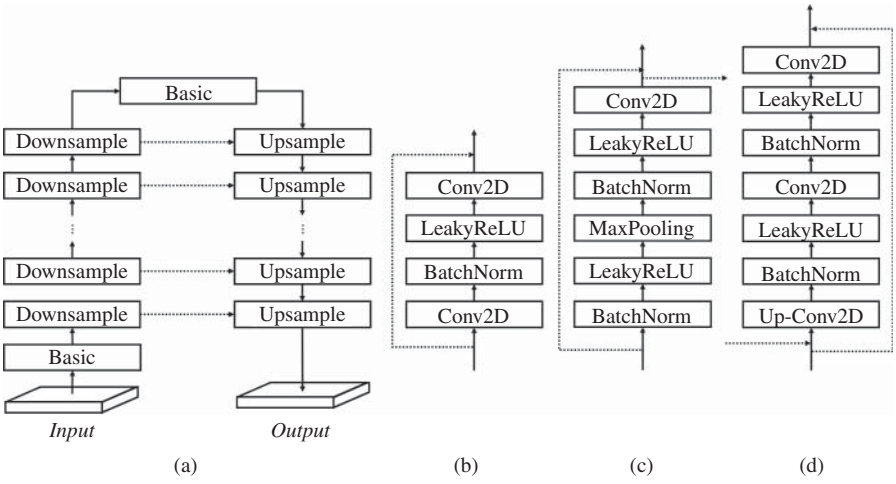

Figure 15.1 (a) The overall structure of the U-NET in Agrawal et al. (2019). Solid lines indicate input connections between layers. Dashed lines indicate skip connections. (b) The operations within the basic layer. (c) The operations within our down-sample layers. (d) The operations within the up-sample layers.

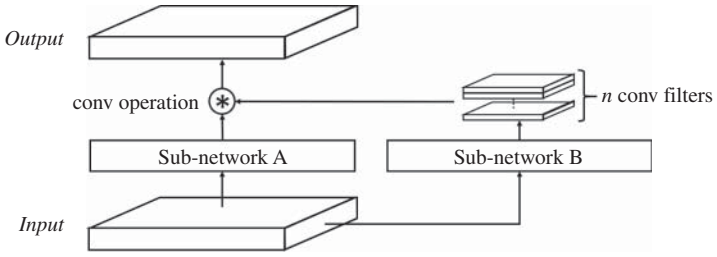

Figure 15.2 The dynamic convolutional layer. The input is fed into two sub-networks. The features are the result of sub-network A while the convolution filters are obtained from sub-network B. The final output of the dynamic convolution layer is computed by convolving the filters from sub-network B across the features from sub-network A.

convolution layer generates both the feature maps and the filters from the input and convolves the filters with the feature maps to get the output. The feature maps and the filters are obtained from the input with two sub-networks. Because the filters are dependent on the input, they will vary from one sample to another in the testing phase. The author concatenates four radar images as the input to predict the next radar image. Also, the author proposed a patch-by-patch synthesis technique which predicts a 10×10 patch in the output from a sequence of 70×70 patches in the input. Figure 15.2 illustrates the workflow of the dynamic convolution layer. Notice that this layer is different from the dynamic filter (Jia et al. 2016) layer that is introduce in section 15.4.2. In dynamic convolution layer, the filter is shared for all locations in the input, while they are adaptively selected in the dynamic filter layer.

Besides the radar images, satellite images are also commonly used as input in FNN-based models. In Lebedev et al. (2019b), satellite images and the observations from Global Forecast System (GFS) (Center 2003) are combined and used as the input. These two types of data are in different modalities and are misaligned with regard to spatial and temporal resolution. Thus, the author remapped them into the same spatial and temporal grid by interpolation. Lebedev et al. (2019b) also applied the U-Net architecture.

Similar to Lebedev et al. (2019b), Hernández et al. (2016) and Qiu et al. (2017) also deal with meteorological data from multiple modalities, including temperature, humidity, wind speed, barometric pressure, Dew point, etc. However, they do not consider the spatial dimension of these data. FNNs with 1D convolution layers and FC layers are built for the $1 \times D$ input data. The weather nowcasting problem is formulated as learning a deterministic mapping $\boldsymbol{Y}_{t+1} = f(\boldsymbol{X}_t)$ that maps the current meteorological observation $\boldsymbol{X}_t$ to the precipitation at next step $\boldsymbol{Y}_{t+1}$. Since the formulation has not fully utilized the spatiotemporal structure of the data, we will not go into the details here.

15.4.2 RNN-based Models

As mentioned in section 15.2, precipitation nowcasting can be formulated as a spatiotemporal sequence forecasting problem with the sequence of past radar maps as input and the sequence of future radar maps as output. In the advancement of DL, RNN-based architectures, such as *Gated Recurrent Unit* (GRU) (Cho et al. 2014) and LSTM (Hochreiter and Schmidhuber 1997), are proven to be effective for modeling sequential data (Sutskever et al.

2014; Karpathy and Fei-Fei 2015; Ranzato et al. 2014; Srivastava et al. 2015; Xu et al. 2015). Different from FNN-based models, which are designed for modeling inputs with static shapes, RNN-based models are designed for modeling dynamic systems. In this section, we introduce the RNN-based models for precipitation nowcasting. We first introduce the encoder-forecaster structure which is the common approach for constructing RNN-based models for spatiotemporal sequence forecasting. Then we introduce the *Convolutional LSTM* (ConvLSTM) network (Xingjian et al. 2015), which combines the advantage of CNN and RNN and is the first DL-based model for precipitation nowcasting. After that, we introduce other RNN-based models like the ConvLSTM with star-shaped bridge (Cao et al. 2019; Chen et al. 2019), *Predictive RNN* (PredRNN) (Wang et al. 2017d), *Memory In Memory* (MIM) Network (Wang et al. 2019b), and the *Trajectory GRU* (TrajGRU) (Shi et al. 2017), which improves upon ConvLSTM from different directions.

15.4.3 Encoder-forecaster Structure

The *Encoder-Forecaster* (EF) structure (Srivastava et al. 2015; Xingjian et al. 2015) is a widely-used neural network architecture for sequence forecasting. It first encodes the observations into a state with an encoder. The state can be a single vector, multiple vectors, or other mathematical objects. Based on the state, it generates the predictions with a forecaster. Following the same notation as in section 15.2, we can formulate the EF structure as follows:

$$\mathbf{H} = f(\mathbf{X}_{t-J+1:t}; \boldsymbol{\theta}_1), \quad \hat{\mathbf{X}}_{t+1:t+L} = g(\mathbf{H}; \boldsymbol{\theta}_2). \tag{15.6}$$

Here, $f(\cdot; \boldsymbol{\theta}_1)$ is the encoder parameterized by $\boldsymbol{\theta}_1$, $g(\cdot; \boldsymbol{\theta}_2)$ is the forecaster parameterized by $\boldsymbol{\theta}_2$, $\hat{\mathbf{X}}_{t+1:t+L}$ are the predictions.

15.4.4 Convolutional LSTM

In the DL community, *Fully-Connected LSTM* (FC-LSTM) is a type of RNN with gates and memory cells for dealing with the vanishing gradient problem (Goodfellow et al. 2016). The formula of FC-LSTM is given as follows:

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{w}_{ci} \odot \boldsymbol{c}_{t-1} + \boldsymbol{b}_i), \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{w}_{cf} \odot \mathbf{c}_{t-1} + \boldsymbol{b}_f), \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tau_h(\boldsymbol{W}_{xc}\boldsymbol{x}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c), \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{w}_{co} \odot \boldsymbol{c}_t + \boldsymbol{b}_o), \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tau_o(\boldsymbol{c}_t),
\end{aligned} \tag{15.7}$$

in which $\boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t$ are correspondingly the input gate, forget gate, and output gate. The $\boldsymbol{c}_{t-1}, \boldsymbol{c}_t$ are the memory cells. $\tau_h(\cdot)$ and $\tau_o(\cdot)$ are the activations, e.g., the "tanh" function. $\boldsymbol{x}_t$s and $\boldsymbol{h}_t$s are input vectors and the hidden states.

However, FC-LSTM is not suitable for precipitation nowcasting in which the input and output are spatiotemporal sequences. The major drawback of FC-LSTM in handling spatiotemporal data is its usage of full-connections in input-state and state-state transitions that loses the spatial structure.

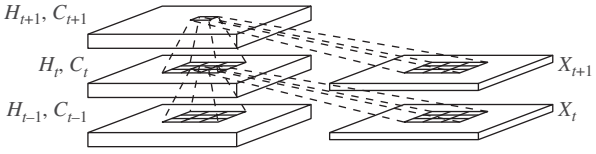

Figure 15.3 Inner structure of ConvLSTM. Source: (Xingjian et al. 2015).

To overcome the problem of FC-LSTM, ConvLSTM (Xingjian et al. 2015) extends FC-LSTM by having convolutional structures in both the input-state and state-state transitions. The key equations of ConvLSTM is given in Equation 15.8. A distinguishing feature of the design is that all the inputs $\mathbf{X}_t$s, cell states $\mathbf{C}_t$s, hidden states $\mathbf{H}_t$s, and gates $\mathbf{I}_t, \mathbf{F}_t, \mathbf{O}_t$ 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. Figure 15.3 illustrates the connection structure of ConvLSTM.

$$
\begin{aligned}
\mathbf{I}_t &= \sigma(\mathbf{W}_{xi} * \mathbf{X}_t + \mathbf{W}_{hi} * \mathbf{H}_{t-1} + \mathbf{W}_{ci} \odot \mathbf{C}_{t-1} + \boldsymbol{b}_i) \\
\mathbf{F}_t &= \sigma(\mathbf{W}_{xf} * \mathbf{X}_t + \mathbf{W}_{hf} * \mathbf{H}_{t-1} + \mathbf{W}_{cf} \odot \mathbf{C}_{t-1} + \boldsymbol{b}_f) \\
\mathbf{C}_t &= \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tau_h(\mathbf{W}_{xc} * \mathbf{X}_t + \mathbf{W}_{hc} * \mathbf{H}_{t-1} + \boldsymbol{b}_c) \\
\mathbf{O}_t &= \sigma(\mathbf{W}_{xo} * \mathbf{X}_t + \mathbf{W}_{ho} * \mathbf{H}_{t-1} + \mathbf{W}_{co} \odot \mathbf{C}_t + \boldsymbol{b}_o) \\
\mathbf{H}_t &= \mathbf{O}_t \odot \tau_o(\mathbf{C}_t)
\end{aligned}
\tag{15.8}
$$

In order to ensure that each state has the same number of rows and columns, the author uses zero-padding in the convolution operator and views it as initializing the state of the outside world to be all zero. Also, the traditional FC-LSTM can be viewed as a special case of ConvLSTM with all features standing on a single cell.

The author adopts ConvLSTM as the building block for the EF architecture. The initial states and cell outputs of the forecasting network are copied from the last state of the encoding network. Both encoder and forecaster are formed by stacking several ConvLSTM layers. All states in the forecasting network are concatenated and fed into a 1×1 convolutional layer to generate the final prediction.

The nowcasting model based on ConvLSTM is compared with the OF-based ROVER (Woo and Wong 2017) algorithm operated in HKO and the FC-LSTM based model on a 97-day radar echo data in Hong Kong. Experiments show that ConvLSTM outperforms both two baselines. Also, the results showed that setting the kernel size of the state-state convolution to be larger than 1 is essential for the final performance.

15.4.5 ConvLSTM with Star-shaped Bridge

To make the feature flow in multi-layer ConvLSTM more robust, Cao et al. (2019) proposed the connection structure called star-shaped bridge. In this structure, the states of all ConvLSTM layers at timestamp t are concatenated and passed to a convolution layer with kernel size 1×1 to obtain a global state. The global state has residual connections to all ConvLSTM states at timestamp $t + 1$. The detailed structure is shown in Figure 15.4.

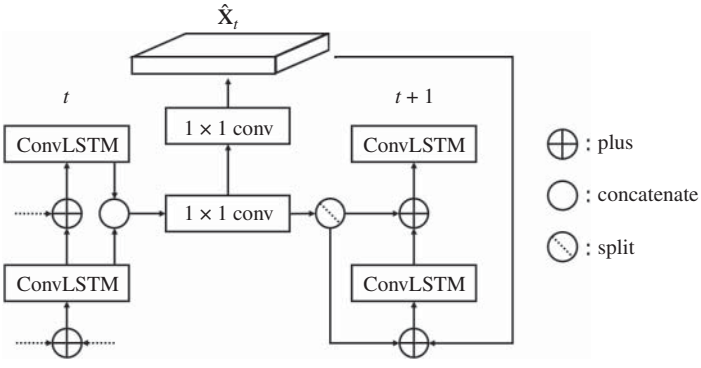

Figure 15.4 Connection structure of the star-shaped bridge.

Apart from the star-shaped bridge, the author also inserts *Group Normalization* (GN) (Wu and He 2018) between ConvLSTM layers. Ablation study shows that the best performance is obtained by combining ConvLSTM, star-shaped bridge, and GN. Also, experiments on 4-year radar echo data from Shanghai, China showed that the learning-based model outperforms the conventional COTREC method (Chen et al. 2019).

15.4.6 Predictive RNN

With memory cells being updated every time step inside the ConvLSTM block, the encoder-forecaster architecture is able to model the underlying temporal dynamics. However, the memory cells across different layers lack mutual communication, which are hence not powerful enough for capturing and memorizing spatial correlations. If we directly stack multiple ConvLSTM layers, the information goes only upwards and makes the features more and more abstract. However, since the network needs to predict a spatiotemporal sequence with fine details, information from the lower-level features, including the raw inputs, should be maintained. To solve the issue, Wang et al. (2017d) proposed *Spatiotemporal LSTM* (ST-LSTM) which keeps an extra memory cell $\mathbf{M}_t^l$ to enhance the memory capacity. The external memory is updated in a zigzag direction illustrated in Figure 15.5 and the author named the whole multi-layer architecture as *Predictive RNN* (PredRNN). The updating rule of ST-LSTM is formulated as follows:

$$
\begin{aligned}
\mathbf{I}_t &= \sigma(\mathbf{W}_{xi} * \mathbf{X}_t + \mathbf{W}_{hi} * \mathbf{H}_{t-1}^l + \boldsymbol{b}_i),\\
\mathbf{F}_t &= \sigma(\mathbf{W}_{xf} * \mathbf{X}_t + \mathbf{W}_{hf} * \mathbf{H}_{t-1}^l + \boldsymbol{b}_f),\\
\mathbf{C}_t^l &= \mathbf{F}_t \odot \mathbf{C}_{t-1}^l + \mathbf{I}_t \odot \tanh(\mathbf{W}_{xc} * \mathbf{X}_t + \mathbf{W}_{hc} * \mathbf{H}_{t-1}^l + \boldsymbol{b}_c),\\
\mathbf{I}_t' &= \sigma(\mathbf{W}_{xi}' * \mathbf{X}_t + \mathbf{W}_{mi} * \mathbf{M}_t^{l-1} + \boldsymbol{b}_i'),\\
\mathbf{F}_t' &= \sigma(\mathbf{W}_{xf}' * \mathbf{X}_t + \mathbf{W}_{mf} * \mathbf{M}_t^{l-1} + \boldsymbol{b}_f'),\\
\mathbf{M}_t^l &= \mathbf{F}_t \odot \mathbf{M}_t^{l-1} + \mathbf{I}_t \odot \tanh(\mathbf{W}_{xm} * \mathbf{X}_t + \mathbf{W}_{hm} * \mathbf{M}_t^{l-1} + \boldsymbol{b}_m),\\
\mathbf{O}_t &= \sigma(\mathbf{W}_{xo} * \mathbf{X}_t + \mathbf{W}_{ho} * \mathbf{H}_{t-1}^l + \mathbf{W}_{co} \odot \mathbf{C}_t + \mathbf{W}_{mo} \odot \mathbf{M}_t^l + \boldsymbol{b}_o),\\
\mathbf{H}_t &= \mathbf{O}_t \odot \tanh(\mathbf{W}_{1\times1} * [\mathbf{C}_t^l, \mathbf{M}_t^l]).
\end{aligned}
\tag{15.9}
$$

Here, $\mathbf{I}_t, \mathbf{F}_t, \mathbf{H}_t, \mathbf{O}_t$ have the same meaning as in Equation 15.8. $\mathbf{C}_t^l$ means the cell state at layer l at timestamp t, and $\mathbf{M}_t^l$ means the external memory cell that will be updated in a

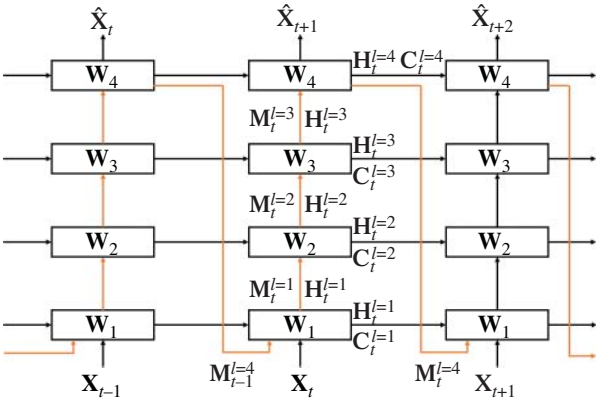

Figure 15.5 Connection structure of PredRNN. The orange arrows in PredRNN denote the flow of the spatiotemporal memory $\mathbf{M}_t^l$.

zigzag order. For the bottom ST-LSTM with $l = 1$, the memory cell from the previous layer is defined as $\mathbf{M}_t^{l-1} = \mathbf{M}_{t-1}^L$, which results in a zigzag update flow. Experiments show that PredRNN outperforms the ConvLSTM structure in precipitation nowcasting. The experiment is conducted on a dataset with 10,000 consecutive radar observations recorded every 6 minutes in Guangzhou, China. 10 frames are used as the input to predict the future 10 frames.

15.4.7 Memory in Memory Network

Temporal dynamics of spatiotemporal processes are usually non-stationary. Most RNN-based models approximate the non-stationary dynamics in a stationary manner. To learn a better representation of the underlying high-order non-stationary structure, *Memory In Memory* (MIM) (Wang et al. 2019b) extends the ST-LSTM by replacing the forget gate with another two embedded long short-term memories. It leverages the differential information between neighboring hidden states in the recurrent paths, and can gradually stationarize the spatiotemporal process by stacking multiple MIM blocks.

As shown in Figure 15.6, two cascaded temporal memory recurrent modules are designed to replace the temporal forget gate $\mathbf{F}_t$ in ST-LSTM. The first module additionally taking $\mathbf{H}_{t-1}^{l-1}$ as input is used to capture the non-stationary variations based on the difference $\mathbf{H}_t^{l-1} - \mathbf{H}_{t-1}^{l-1}$ between two consecutive hidden representations. Thus, it is named as the non-stationary module (shown as MIM-N in Figure 15.7). It generates differential features $\mathbf{D}_t^l$ based on the difference-stationary assumption. The other recurrent module takes the output $\mathbf{D}_t^l$ of the MIM-N module and the outer temporal memory $\mathbf{C}_{t-1}^l$ as inputs to capture the approximately stationary variations in spatiotemporal sequences. Thus, it is named as the stationary module (shown as MIM-S in Figure 15.7). By replacing the forget gate with the final output $\mathbf{T}_t^l$ of the cascaded non-stationary and stationary modules, the non-stationary dynamics can be captured more effectively. The complete formula of MIM is given as follows:

$$
\begin{aligned}
\mathbf{I}_t &= \sigma(\mathbf{W}_{xi} * \mathbf{X}_t + \mathbf{W}_{hi} * \mathbf{H}_{t-1}^l + \boldsymbol{b}_i), \\
\mathbf{F}_t &= \sigma(\mathbf{W}_{xf} * \mathbf{X}_t + \mathbf{W}_{hf} * \mathbf{H}_{t-1}^l + \boldsymbol{b}_f), \\
\mathbf{D}_t^l &= \text{MIM-N}(\mathbf{H}_t^{l-1}, \mathbf{H}_{t-1}^{l-1}, \mathbf{N}_{t-1}^l), \\
\mathbf{T}_t^l &= \text{MIM-S}(\mathbf{D}_t^l, \mathbf{C}_{t-1}^l, \mathbf{S}_{t-1}^l),
\end{aligned}
$$

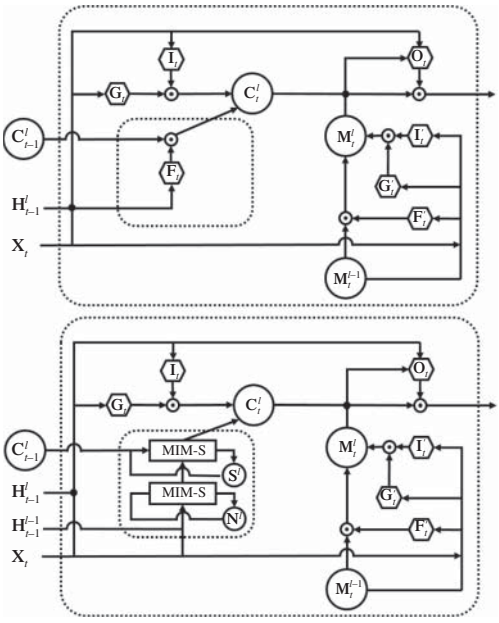

Figure 15.6 ST-LSTM block (top) and Memory In Memory block (bottom). For brevity, $\mathbf{G}_t = \tanh(\mathbf{W}_{xc} * \mathbf{X}_t + \mathbf{W}_{hc} * \mathbf{H}^l_{t-1} + \boldsymbol{b}_c)$, $\mathbf{G}'_t = \tanh(\mathbf{W}_{xm} * \mathbf{X}_t + \mathbf{W}_{hm} * \mathbf{M}^{l-1}_t + \boldsymbol{b}_m)$. MIM is designed to introduce two recurrent modules (yellow squares) to replace the forget gate (dashed box) in ST-LSTM. MIM-N is the non-stationary module and MIM-S is the stationary module.

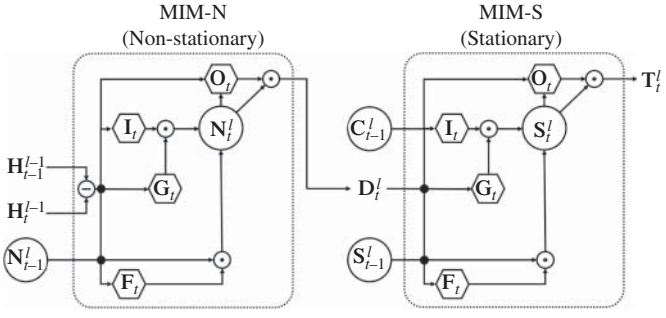

Figure 15.7 The non-stationary module (MIM-N) and the stationary module (MIM-S), which are interlinked in a cascaded structure in the MIM block. Non-stationarity is modeled by differencing.

$$
\begin{aligned}
\mathbf{C}^l_t &= \mathbf{F}_t \odot \mathbf{T}^l_t + \mathbf{I}_t \odot \tanh(\mathbf{W}_{xc} * \mathbf{X}_t + \mathbf{W}_{hc} * \mathbf{H}^l_{t-1} + \boldsymbol{b}_c),\\
\mathbf{I}'_t &= \sigma(\mathbf{W}'_{xi} * \mathbf{X}_t + \mathbf{W}_{mi} * \mathbf{M}^{l-1}_t + \boldsymbol{b}'_i),\\
\mathbf{F}'_t &= \sigma(\mathbf{W}'_{xf} * \mathbf{X}_t + \mathbf{W}_{mf} * \mathbf{M}^{l-1}_t + \boldsymbol{b}'_f),\\
\mathbf{M}^l_t &= \mathbf{F}_t \odot \mathbf{M}^{l-1}_t + \mathbf{I}_t \odot \tanh(\mathbf{W}_{xm} * \mathbf{X}_t + \mathbf{W}_{hm} * \mathbf{M}^{l-1}_t + \boldsymbol{b}_m),\\
\mathbf{O}_t &= \sigma(\mathbf{W}_{xo} * \mathbf{H}^{l-1}_t + \mathbf{W}_{ho} * \mathbf{H}^l_{t-1} + \mathbf{W}_{co} \odot \mathbf{C}_t + \mathbf{W}_{mo} \odot \mathbf{M}^l_t + \boldsymbol{b}_o),\\
\mathbf{H}_t &= \mathbf{O}_t \odot \tanh(\mathbf{W}_{1\times1} * [\mathbf{C}^l_t, \mathbf{M}^l_t]),
\end{aligned}
\tag{15.10}
$$

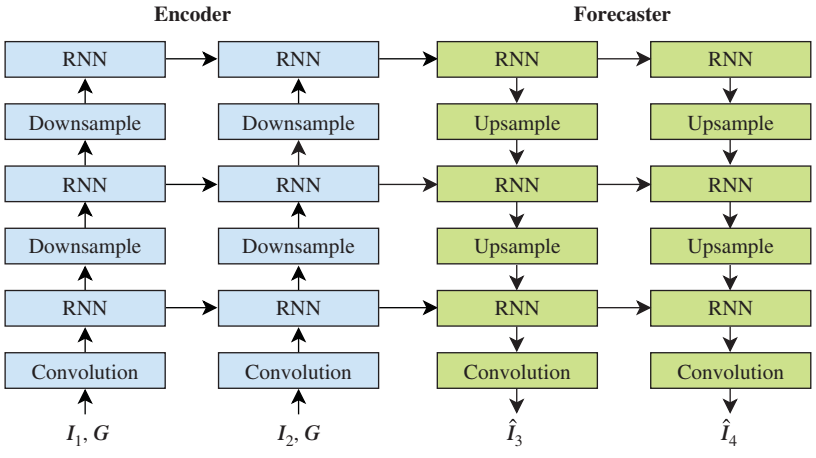

Figure 15.8 Encoder-forecaster architecture adopted in Shi et al. (2017). Source: Shi et al. (2017).

where $\mathbf{S}$ and $\mathbf{N}$ denote the horizontally-transited memory cells in the non-stationary module (MIM-N) and stationary module (MIM-S) respectively; $\mathbf{D}_t^l$s are the differential features learned by MIM-N; $\mathbf{T}_t^l$ is the memory passing the virtual "forget gate". MIM-N is a ConvLSTM with $\mathbf{H}_t^{l-1} - \mathbf{H}_{t-1}^{l-1}$ as the hidden state input. MIM-S is a ConvLSTM with $\mathbf{D}_t^l$ as the hidden state input. The detailed formula are omitted here and readers can refer to Wang et al. (2019b) for more details.

15.4.8 Trajectory GRU

Shi et al. (2017) proposed a U-Net-like modification to the EF architecture. As Figure 15.8 illustrates, the order of the forecaster network is reversed comparing to that in Xingjian et al. (2015). There are downsampling and upsampling layers between the RNNs, which are implemented by strided convolution and deconvolution. In this structure, the encoder adopts a local-to-global feature extraction process while the decoder adopts a coarse-to-fine generation process. There are "skip-connections" between the encoder and the forecaster to preserve the details from the raw inputs.

Along with the new EF structure, Shi et al. (2017) also pointed out a side-effect of the convolution operation in ConvLSTM. In essence, the location-invariant convolution filters are inefficient to capture location-variant spatiotemporal relationship. To overcome this problem, Shi et al. (2017) proposed the *Trajectory GRU* (TrajGRU) model which uses a sub-network to output the state-state connection structures before state transitions. TrajGRU extends upon *Convolutional GRU* (ConvGRU), which is a variant of ConvLSTM, and allows the state to be aggregated along some learned trajectories. The formula of ConvGRU is given as follows:

$$
\begin{aligned}
\mathbf{Z}_t &= \sigma(\mathbf{W}_{xz} * \mathbf{X}_t + \mathbf{W}_{hz} * \mathbf{H}_{t-1} + \boldsymbol{b}_z), \\
\mathbf{R}_t &= \sigma(\mathbf{W}_{xr} * \mathbf{X}_t + \mathbf{W}_{hr} * \mathbf{H}_{t-1} + \boldsymbol{b}_r), \\
\mathbf{H}'_t &= f(\mathbf{W}_{xh} * \mathbf{X}_t + \mathbf{R}_t \odot (\mathbf{W}_{hh} * \mathbf{H}_{t-1} + \boldsymbol{b}_h)), \\
\mathbf{H}_t &= (1 - \mathbf{Z}_t) \odot \mathbf{H}'_t + \mathbf{Z}_t \odot \mathbf{H}_{t-1}.
\end{aligned} \tag{15.11}
$$

Here, $\mathbf{H}_t, \mathbf{R}_t, \mathbf{Z}_t, \mathbf{H}'_t \in \mathbb{R}^{C_h \times H \times W}$ are the memory state, reset gate, update gate, and new information, respectively.

As stated in Shi et al. (2017), when used for capturing spatiotemporal correlations, the deficiency of ConvGRU and ConvLSTM is that the connection structure and weights are fixed for all the locations. The convolution operation basically applies a *location-invariant* filter to the input. If the inputs are all zero and the reset gates are all one, the author pointed out that the calculation process of $\mathbf{H}'_t$ at a specific location (i, j), i.e, $\mathbf{H}'_{t,:,i,j}$, can be rewritten as follows:

$$\mathbf{H}'_{t,:,i,j} = f(\sum_{l=1}^{|\mathcal{N}^h_{i,j}|} \boldsymbol{W}^l_{hh} \mathbf{H}_{t-1,:,p(l,i,j),q(l,i,j)}), \tag{15.12}$$

in which $\mathcal{N}^h_{i,j}$ is the ordered neighborhood set at location (i, j) defined by the hyperparameters of the state-state convolution such as kernel size, dilation and padding (Yu and Koltun 2016). $(p(l, i, j), q(l, i, j))$ is the lth neighborhood location corresponding to position (i, j).

Based on this observation, TrajGRU uses the current input and previous state to generate the local neighborhood set for each location at each timestamp. The detailed formula is given in Equation 15.13. Here, L is the number of allowed links. $\mathbf{U}_t, \mathbf{V}_t \in \mathbb{R}^{L \times H \times W}$ are the flow fields that store the local connection structure generated by $\gamma(\mathbf{X}_t, \mathbf{H}_{t-1})$. The $\mathbf{W}^l_{hz}, \mathbf{W}^l_{hr}, \mathbf{W}^l_{hh}$ are the weights for projecting the channels and were chosen as 1×1 convolutions in the paper. The warp$(\mathbf{H}_{t-1}, \mathbf{U}_{t,l}, \mathbf{V}_{t,l})$ function selects the positions pointed out by $\mathbf{U}_{t,l}, \mathbf{V}_{t,l}$ from $\mathbf{H}_{t-1}$ via the bilinear sampling kernel (Jaderberg et al. 2015; Ilg et al. 2017; Shi et al. 2017).

$$\begin{aligned}
\mathbf{U}_t, \mathbf{V}_t &= \gamma(\mathbf{X}_t, \mathbf{H}_{t-1}), \\
\mathbf{Z}_t &= \sigma(\mathbf{W}_{xz} * \mathbf{X}_t + \sum_{l=1}^{L} \mathbf{W}^l_{hz} * \text{warp}(\mathbf{H}_{t-1}, \mathbf{U}_{t,l}, \mathbf{V}_{t,l})), \\
\mathbf{R}_t &= \sigma(\mathbf{W}_{xr} * \mathbf{X}_t + \sum_{l=1}^{L} \mathbf{W}^l_{hr} * \text{warp}(\mathbf{H}_{t-1}, \mathbf{U}_{t,l}, \mathbf{V}_{t,l})), \\
\mathbf{H}'_t &= f(\mathbf{W}_{xh} * \mathbf{X}_t + \mathbf{R}_t \odot (\sum_{l=1}^{L} \mathbf{W}^l_{hh} * \text{warp}(\mathbf{H}_{t-1}, \mathbf{U}_{t,l}, \mathbf{V}_{t,l}))), \\
\mathbf{H}_t &= (1 - \mathbf{Z}_t) \odot \mathbf{H}'_t + \mathbf{Z}_t \odot \mathbf{H}_{t-1}.
\end{aligned} \tag{15.13}$$

The advantage of such a structure is that it could learn the connection topology by learning the parameters of the subnetwork γ. γ has only a small number of parameters and thus adds nearly no cost to the overall computation. Compared to a ConvGRU with $K \times K$ state-state convolution, TrajGRU is able to learn a more efficient connection structure with $L < K^2$. For ConvGRU and TrajGRU, the number of model parameters is dominated by the size of the state-state weights, which is $O(L \times C_h^2)$ for TrajGRU and $O(K^2 \times C_h^2)$ for ConvGRU. If L is chosen to be smaller than K^2, the number of parameters of TrajGRU can also be smaller than the ConvGRU and the TrajGRU model is able to use the parameters more efficiently. Illustration of the recurrent connection structures of ConvGRU and TrajGRU is given in Figure 15.9.

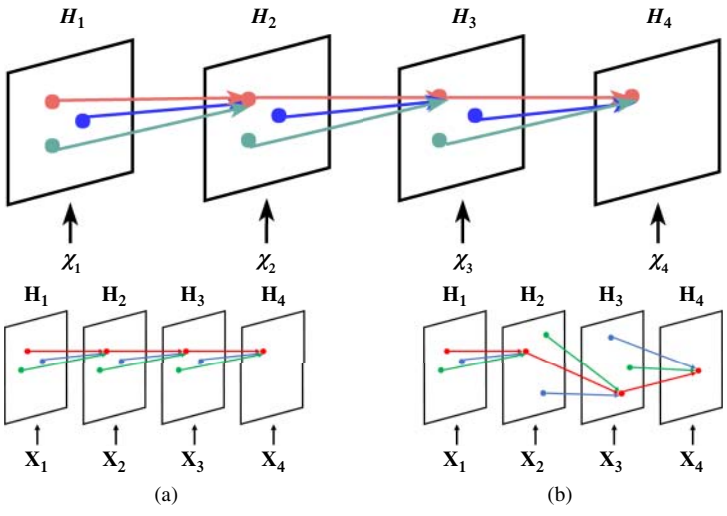

Figure 15.9 Top: For convolutional RNN, the recurrent connections are fixed over time. Bottom: For trajectory RNN, the recurrent connections are dynamically determined. Comparison of the connection structures of convolutional RNN and trajectory RNN. Links with the same color share the same transition weights. (Best viewed in color). Source of figure: Shi et al. (2017).

Experiments in the paper showed that TrajGRU outperforms ConvGRU, 2D CNN, 3D CNN, and the ROVER algorithm in precipitation nowcasting.

15.5 Benchmark

Despite the rapid development of DL models in solving this problem, the way to evaluate the models has some deficiencies. Firstly, the deep learning models are only evaluated on relatively small dataset containing limited data frames. Secondly, different models report evaluation results on different criteria. As the needs of real-world precipitation nowcasting system diverge from indicating raining or not to rainstorms alert, single criterion is not sufficient for demonstrating the algorithm's overall performance. Thirdly, in the real-world scenario, the meteorological data arrive in a stream and the nowcasting algorithm should be able to actively adapt to the new-coming sequences. Considering this online setting is not less important than considering offline setting with fixed-length input. In fact, as the area *deep learning for precipitation nowcasting* is still in its early stages, it is not clear how models should be evaluated to meet the needs of real-world applications.

Shi et al. (2017) proposed the large-scale HKO-7 benchmark for precipitation nowcasting to address this problem. HKO-7 benchmark is built on the HKO-7 dataset containing radar echo data from 2009 to 2015 near Hong Kong. Since the radar echo maps arrive in a stream in the real-world scenario, the nowcasting algorithms can adopt online learning to adapt to the newly emerging patterns dynamically. To take this setting into account, there are two testing protocols in this benchmark: the offline setting in which the algorithm can only use a fixed window of the previous radar echo maps and the *online setting* in which the algorithm is free to use all the historical data and any online learning algorithm. Another

issue for the precipitation nowcasting task is that the proportions of rainfall events at different rain-rate thresholds are highly imbalanced. Heavier rainfall occurs less often but has a higher real-world impact. *Balanced Mean Squared Error* (B-MSE) and *Balanced Mean Absolute Error* (B-MAE) measures are thus introduced for training and evaluation, which assign more weights to heavier rainfalls in the calculation of MSE and MAE. Empirical study showed that the balanced variants of the loss functions are more consistent with the overall nowcasting performance at multiple rain-rate thresholds than the original loss functions. Moreover, training with the balanced loss functions is essential for deep learning models to achieve good performance at higher rain-rate thresholds.

Using the new dataset, testing protocols, and training loss, there are seven models being extensively evaluated, including a simple baseline model which always predicts the last frame, two OF-based models (ROVER and its nonlinear variant), and four representative deep learning models (TrajGRU, ConvGRU, 2D CNN, and 3D CNN). This large-scale benchmark for precipitation nowcasting is the first comprehensive benchmark of deep learning models for the precipitation nowcasting problem.

15.5.1 HKO-7 Dataset

The HKO-7 dataset used in the benchmark contains radar echo data from 2009 to 2015 collected by HKO. The radar CAPPI reflectivity images, which have resolution of 480 × 480 pixels, are taken from an altitude of 2 km and cover a 512 km×512 km area centered in Hong Kong. The data are recorded every 6 minutes and hence there are 240 frames per day. The raw logarithmic radar reflectivity factors are linearly transformed to pixel values via $\text{pixel} = \lfloor 255 \times \frac{\text{dBZ}+10}{70} + 0.5 \rfloor$ and are clipped to be between 0 and 255. The radar reflectivity values are converted to rainfall intensity values (mm/h) using the Z-R relationship: $\text{dBZ} = 10 \log a + 10b \log R$ where R is the rain-rate level, $a = 58.53$, and $b = 1.56$. As rainfall events occur sparsely, the rainy days are selected based on the rain barrel information to form the final dataset, which has 812 days for training, 50 days for validation and 131 days for testing.

The raw radar echo images generated by Doppler weather radar are noisy due to factors like ground clutter, sea clutter, anomalous propagation and electromagnetic interference (Lee and Kim 2017). To alleviate the impact of noise in training and evaluation, Noisy pixels are filtered out by generating the noise masks with a two-stage process.

15.5.2 Evaluation Methodology

As the radar echo maps arrive in a stream, nowcasting algorithms can apply online learning to adapt to the newly emerging spatiotemporal patterns. The evaluation protocol of HKO-7 benchmark consists of two settings: (i) the offline setting in which the algorithm always receives 5 frames as input and predicts 20 frames ahead, and (ii) the online setting in which the algorithm receives segments of length 5 sequentially and predicts 20 frames ahead for each new segment received. The testing environment guarantees that the same set of sequences is tested in both the offline and online settings for fair comparison.

For both settings, models are evaluated according to the skill scores for multiple thresholds that correspond to different rainfall levels to give an all-round evaluation of the algorithms' nowcasting performance. Table 15.1 shows the distribution of different

Table 15.1 Rain rate statistics in the HKO-7 benchmark.

Rain Rate (mm/h)			Proportion (%)	Rainfall Level
0 $\leq$	x	< 0.5	90.25	No / Hardly noticeable
0.5 $\leq$	x	< 2	4.38	Light
2 $\leq$	x	< 5	2.46	Light to moderate
5 $\leq$	x	< 10	1.35	Moderate
10 $\leq$	x	< 30	1.14	Moderate to heavy
30 $\leq$	x		0.42	Rainstorm warning

Source: (Shi et al. 2017).

rainfall levels in HKO-7 dataset. The thresholds 0.5, 2, 5, 10, 30 are selected to calculate the CSI and Heidke Skill Score (HSS) (Hogan et al. 2010). For calculating the skill score at a specific threshold τ, which is 0.5, 2, 5, 10 or 30, the pixel values in prediction and ground-truth are first converted to 0/1 by thresholding with τ. Then calculate the TP (prediction=1, truth=1), FN (prediction=0, truth=1), FP (prediction=1, truth=0), and TN (prediction=0, truth=0). The CSI score is calculated as $\frac{\text{TP}}{\text{TP+FN+FP}}$ and the HSS score is calculated as $\frac{\text{TP}\times\text{TN}-\text{FN}\times\text{FP}}{(\text{TP+FN})(\text{FN+TN})+(\text{TP+FP})(\text{FP+TN})}$. During the computation, the masked noisy points are ignored.

As shown in Table 15.1, the frequencies of different rainfall levels are highly imbalanced. Using weighted loss function helps solve this problem. Specifically, a weight $w(x)$ is assigned to each pixel according to its rainfall intensity x:

$$w(x) = \begin{cases} 1, & x < 2 \\ 2, & 2 \leq x < 5 \\ 5, & 5 \leq x < 10 \\ 10, & 10 \leq x < 30 \\ 30, & x \geq 30 \end{cases}.$$

Also, the masked pixels have weight 0. The resulting B-MSE and B-MAE scores are computed as $\text{B-MSE} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{480}\sum_{j=1}^{480} w_{n,i,j}(x_{n,i,j} - \hat{x}_{n,i,j})^2$ and $\text{B-MAE} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{480}\sum_{j=1}^{480} w_{n,i,j}|x_{n,i,j} - \hat{x}_{n,i,j}|$, where N is the total number of frames and $w_{n,i,j}$ is the weight corresponding to the (i,j)th pixel in the nth frame. For the conventional MSE and MAE measures, all the weights are simply set to 1 except the masked points.

15.5.3 Evaluated Algorithms

There are seven nowcasting algorithms for evaluation in HKO-7 benchmark, including the simplest model which always predicts the last frame, two optical flow based methods (ROVER and its nonlinear variant), and four deep learning methods (TrajGRU, ConvGRU, 2D CNN, and 3D CNN). Specifically in the online setting, models are fine-tuned using AdaGrad (Duchi et al. 2011) with learning rate equal to 10^{-4}. The training objective during offline training and online fine-tuning is the sum of B-MSE and B-MAE. During the offline

training process, all models are optimized by the Adam optimizer with learning rate equal to 10^{-4} and momentum equal to 0.5, with early-stopping on the sum of B-MSE and B-MAE. The ConvGRU model is also trained with the original MSE and MAE loss, which is named "ConvGRU-nobal" in the paper (Shi et al. 2017), to evaluate the improvement by training with the B-MSE and B-MAE loss.

15.5.4 Evaluation Results

The experiment results show that training with balanced loss functions is essential for good nowcasting performance of heavier rainfall. The ConvGRU model that is trained without balanced loss, which best represents the model in Xingjian et al. (2015), has a worse nowcasting score than the optical flow based methods at the 10 mm/h and 30 mm/h thresholds. Also, all the deep learning models that are trained with the balanced loss outperform the optical flow based models. Among the deep learning models, TrajGRU performs the best and 3D CNN outperforms 2D CNN, which shows that an appropriate network structure is crucial to achieving good performance. The improvement of TrajGRU over the other models is statistically significant because the differences in B-MSE and B-MAE are larger than three times their standard deviation. Moreover, the performance with online fine-tuning is consistently better than that without online fine-tuning, which verifies the effectiveness of online learning at least for this task.

The results of the Kendall's τ coefficients (Kendall 1938) between the MSE, MAE, B-MSE, B-MAE and the CSI, HSS at different thresholds show that B-MSE and B-MAE have stronger correlations with the CSI and HSS in most cases.

15.6 Discussion

In this chapter, we reviewed the DL-based methods for precipitation nowcasting. The architecture, building block, training objective function, metrics, and data source of the reviewed methods are summarized in Table 15.2. Precipitation nowcasting is formulated as a spatiotemporal sequence forecasting problem from the machine learning perspective. Thanks to the increased computational power and the amounts of data, the area is making rapid progress. Machine learning, specifically deep learning, facilitates the large amount of weather data and provides promising research directions for better modeling and understanding of precipitation nowcasting problem. Despite the success of DL-based methods achieved on precipitation nowcasting, this problem is still challenging. Below we list several major future research directions that are not solved or have not been explored:

- **Utilization of multi-source meteorological data**
  While existing DL-based models mainly focus on single-source data, typically radar echo maps or satellite images, multi-source meteorological data have become available thanks to rapidly developing sensing techniques as well as increasing data storage. Although DL-based models extract spatiotemporal features effectively, precipitation nowcasting using only single-source data is essentially ill-posed. The models are not offered complete knowledge on the dynamics of the meteorological system, hence they

Table 15.2 Summary of reviewed methods. The first half are FNN-based models and the second half are RNN-based models.

Method	Building Block	Architecture	Objectives	Metrics	Data Sources
Klein et al. (2015)	Dynamic CNN	Stacked CNN and dynamic CNN	MSE	MSE	Radar
Ayzel et al. (2019)	CNN	DozhdyaNet	MAE, MSE, Logcosh	MAE, CSI	Radar
Agrawal et al. (2019)	CNN	U-Net	Cross-entropy for three binary classifications	Precision, Recall	
Lebedev et al. (2019b)	CNN	U-Net	Cross-entropy for three binary classifications, Dice loss (Sudre et al. 2017)	Accuracy, Precision, Recall, F1 score	Satellite
Hernández et al. (2016)	MLP	AE, MLP	MSE	MSE	47 observational features
Qiu et al. (2017)	1D-CNN	Stacked 1D-CNN and FC layer	MSE with Frobenius norm	MSE, CSI, Correlation	Multi-site observational features
Xingjian et al. (2015)	ConvLSTM	Spatiotemporal encoder-forecaster	Cross-entropy	MSE, CSI, FAR, POD, Correlation	Radar
Shi et al. (2017)	TrajGRU	Encoder-forecaster with downsampling and upsampling	MSE, MAE, B-MSE, B-MAE	HKO-7 benchmark	Radar
Tran and Song (2019)	ConvRNN	encoder-forecaster with downsampling and upsampling	SSIM, MS-SSIM (Wang et al. 2003; Wang and Bovik 2009)	MSE, MAE, SSIM, MS-SSIM, PCC (Inamdar et al. 2018)	Radar
Cao et al. (2019)	ConvLSTM	Star-shaped Bridge	Multi-Sigmoid Loss	MSE, CSI	Radar
Wang et al. (2017d)	ST-LSTM	PredRNN	MSE, MAE	MSE	Radar
Wang et al. (2019b)	MIM block	MIM network	MSE	MSE, CSI	Radar

fail to accurately model it and infer its future evolution. Multi-source data, in contrast, provide multi-modal and multiscale meteorological information, giving the model a more holistic view of the system. Therefore exploring DL models that are able to jointly process complementary multi-source data can certainly help learn better representations of the observing systems.

- **Handling uncertainty**
  Precipitation nowcasting involves complex physics dynamics. According to chaos theory, chaotic behaviors in a meteorological system make it unpredictable due to high degree of uncertainty. Learning to capture the internal uncertainty is one of the major challenge in modeling and understanding the latent dynamics. However, most DL models address precipitation nowcasting in deterministic manner, which averages all possible futures into a single output, without describing its whole distribution. For some application scenarios such as rainstorm alert, not only the average and the most likely futures are concerned, but also some possible extreme cases should be considered. There are some recent works (Xue et al. 2016; Babaeizadeh et al. 2018; Denton and Fergus 2018; Lee et al. 2018a) that developed stochastic spatiotemporal models to predict different possible futures through variational inference. Though stochastic spatiotemporal models are not yet evaluated on precipitation nowcasting tasks, they are inspiring potential solutions for handling uncertainty in precipitation nowcasting.
- **Integration with numerical methods**
  Compared with theory-driven quantitative precipitation forecast (QPF) methods with clear physical meanings, deep learning models are data-driven and typically suffer from poor interpretability. Although theory-driven precipitation nowcasting models are derived from physical theories, they are essentially phenomenological models built by summarizing the empirical relationship of observations instead of deriving from first principles, which means theory-driven models are not entirely different from but in essence analogous to data-driven models. Theory-driven models consist of interpretable components to describe the observed data, while keeping consistent with physical laws including conservation of mass, momentum, energy, etc. They are determined by human experts and are hence hard to adjust according to different data from different distributions. On the contrary, data-driven models are equipped with high flexibility to adapt to different datasets since they directly learn parameters from data under few constraints. These two approaches are complementary in respect of interpretability and flexibility. Integrating theory-driven and data-driven approaches provides new opportunities in future precipitation nowcasting research, including but not limited to model calibration, recognizing unidentified observations, etc.

Appendix

The deep learning precipitation nowcast models introduced in this chapter have been utilized to support development of the operational rainfall nowcasting system, namely SWIRLS (Short-range Warning of Intense Rainstorms in Localized Systems) of the Hong Kong Observatory (HKO). In particular, TrajGRU has also been made available in the community version of SWIRLS (a.k.a. Com-SWIRLS) as part of core components under the Regional Specialized Meteorological Centre (RSMC) for Nowcasting of HKO, see https://rsmc.hko.gov.hk. The rainfall nowcasting models including TrajGRU are shared with the National Meteorological and Hydrological Services (NMHSs) of the World Meteorological

Organization to promote the development of nowcasting techniques. More information on Com-SWIRLS can be found at the website, see https://com-swirls.org/.

Acknowledgement

This research has been partially supported by General Research Fund 16207316 from the Research Grants Council of Hong Kong.