Yan BY, Yang C, Chen F *et al.* FDNet: A deep learning approach with two parallel cross encoding pathways for precipitation nowcasting. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 38(5): 1002-1020 Sept. 2023. DOI: 10.1007/s11390-021-1103-8

FDNet: A Deep Learning Approach with Two Parallel Cross Encoding Pathways for Precipitation Nowcasting

Bi-Ying Yan[1, 2] (闫碧莹), *Member, CCF*, Chao Yang[3, 4, *] (杨　超), *Senior Member, CCF, Member, ACM, IEEE*
Feng Chen[2, 5] (陈　峰), Kohei Takeda[6], and Changjun Wang[7]

[1] *University of Chinese Academy of Sciences, Beijing 100049, China*

[2] *Institute of Software, Chinese Academy of Sciences, Beijing 100190, China*

[3] *School of Mathematical Sciences, Peking University, Beijing 100871, China*

[4] *Peng Cheng Laboratory, Shenzhen 518052, China*

[5] *Guiyang Academy of Information Technology, Guiyang 550081, China*

[6] *NTT DATA Corporation, Tokyo 163-8001, Japan*

[7] *NTT DATA Institute of Management Consulting Inc., Tokyo 163-8001, Japan*

E-mail: biying@iscas.ac.cn; chao_yang@pku.edu.cn; chenfeng@iscas.ac.cn; Kohei.Takeda@nttdata.com
wanvc@nttdata-strategy.com

Received October 22, 2020; accepted May 6, 2021.

Abstract With the goal of predicting the future rainfall intensity in a local region over a relatively short period time, precipitation nowcasting has been a long-time scientific challenge with great social and economic impact. The radar echo extrapolation approaches for precipitation nowcasting take radar echo images as input, aiming to generate future radar echo images by learning from the historical images. To effectively handle complex and high non-stationary evolution of radar echoes, we propose to decompose the movement into optical flow field motion and morphologic deformation. Following this idea, we introduce Flow-Deformation Network (FDNet), a neural network that models flow and deformation in two parallel cross pathways. The flow encoder captures the optical flow field motion between consecutive images and the deformation encoder distinguishes the change of shape from the translational motion of radar echoes. We evaluate the proposed network architecture on two real-world radar echo datasets. Our model achieves state-of-the-art prediction results compared with recent approaches. To the best of our knowledge, this is the first network architecture with flow and deformation separation to model the evolution of radar echoes for precipitation nowcasting. We believe that the general idea of this work could not only inspire much more effective approaches but also be applied to other similar spatio-temporal prediction tasks.

Keywords spatio-temporal predictive learning, precipitation nowcasting, neural network

1 Introduction

Precipitation nowcasting is the task of providing precise and timely prediction of rainfall intensity in a local region over a very short range (e.g., 0 hour–6 hour) based on radar echo maps, rain gauge, and other observation data. Accurate prediction plays a vital role in our daily life as well as many public safety scenarios such as road condition alarms and flight schedules. Since the accuracy and timeliness are highly desired, compared with other traditional forecasting tasks like weekly average temperature prediction, precipitation nowcasting has become a more challenging mission in the field of weather forecasting. Of particular note is the forecasting of heavier rainfall, which occurs less often but has a higher real-world impact.

Regular Paper

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2018YFC0831500, the Beijing Natural Science Foundation under Grant No. JQ18001, and the Beijing Academy of Artificial Intelligence.

*Corresponding Author

Forecasting has become extremely difficult because of the low frequency of heavier rainfall events.

Existing methods for precipitation nowcasting include numerical weather prediction (NWP) based methods and radar echo extrapolation based methods[1]. NWP is one of the most successful approaches to conducting medium- and long-range (up to six days) weather prediction[1, 2]. The core of NWP is the complex and meticulous simulation of the physical equations in the atmosphere model. These physical equations are coincident with our current beliefs about the dynamical behaviors of the atmosphere. The NWP-based approaches have high computing complexity and their accuracy strongly relies on the initial conditions, which could make them lose competitiveness to the extrapolation-based methods in a short period forecast.

Recently, radar echo extrapolation based methods have been noticed and widely adopted[3, 4] these years. These methods[3, 4] rely on the extrapolation of observations by ground-based radars via optical flow techniques or neural network models. Optical flow-based methods[4–6], as typical extrapolation-based methods, have drawn increasingly more attention, owing to their fast speeds and high accuracies. Optical flow-based methods are conducted in two stages by the extrapolation of radar observations. Firstly, the wind is estimated by comparing two or more precipitation fields as seen by radar based on the optical flow estimation algorithms developed in computer vision. Secondly, the precipitation field is moved along the estimated directions of the wind. However, the success of these optical flow-based methods is limited because the flow estimation step and the radar echo extrapolation step are separated and it is not easy to determine the most appropriate model parameters.

Neural network models consider precipitation nowcasting as a real application of spatio-temporal predictive learning which generates images conditioned on given consecutive frames and has shown the models' advantages[7–9] in many real-world datasets. In spatio-temporal predictive learning, there are two crucial aspects: spatial correlations and temporal dynamics, and previous studies[7, 9] mainly focus on how to model these two sides in a unified architecture. The Convolutional LSTM (ConvLSTM)[7] architecture naturally considers these two aspects jointly by plugging the convolutional operations in recurrent connections. Predictive RNN (PredRNN)[9] extends ConvLSTM and involves a new spatio-temporal LSTM (ST-LSTM) unit aiming to memorize both spatial appearances and temporal variations in a unified memory pool. Just the opposite of PredRNN, MCNet[10] captures the spatial layout of an image and the corresponding temporal dynamics independently.

In general, the evolution of radar echoes can be resolved into optical flow field motion and morphologic deformation. As shown in Fig.1, the pixels are moving from left to right as a whole. But zooming in to the detail, the accumulation, deformation, and dissipation of the radar echoes are happening in every region at every moment. Therefore, it is difficult to capture all these fine-grained evolution patterns with a uniform encoding structure.

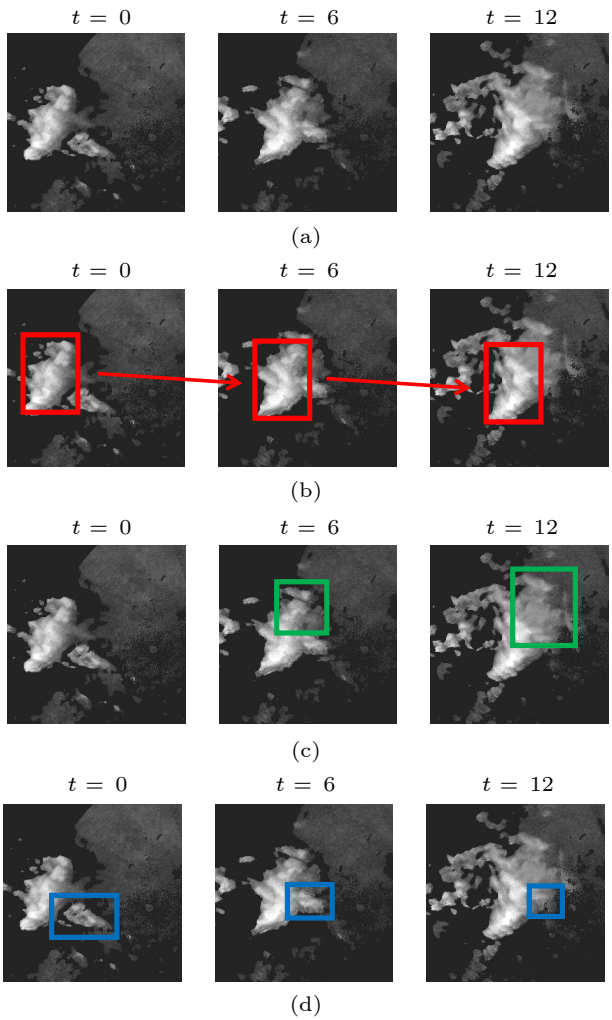


Fig.1. Example of 12 consecutive radar echo maps to illustrate the optical flow field motion and morphologic deformation of radar echoes, where t is the time step. (a) Visualizations of a typical radar echo sequence. (b) Translational motion of the main body of radar echoes (in red boxes). (c) Accumulation and deformation of the radar echoes (in green boxes). (d) The radar echoes in blue boxes dissipate and blend into the main body.

Unfortunately, previous work only considers these two aspects of evolution together, which may lose sight of fine-grained variation. Since atmospheric motion is a complex physical process and the shapes of radar echoes may expand, contract or change rapidly, modeling morphologic deformation is significant for the prediction of radar echo maps. Motivated by this, we present a new prediction architecture called Flow Deformation Network (FDNet), which considers optical flow field motion and morphologic deformation separately, aiming to capture the fine-grained evolution of radar echoes. FDNet separates the information streams (the position and the shape) into different encoding pathways. It applies a neural optical flow estimation method to the position stream to extract the spatial coherence motion and a differencing operation on the shape stream to capture the fine-grained spatial deformation. The proposed architecture is evaluated on two real-world radar echo datasets, and the results show that it outperforms previous approaches[8–12], especially for longer future time steps.

The rest of the paper is organized as follows. A brief review of related work is given in Section 2 and some preliminaries are introduced in Section 3. The detailed structure of the proposed network is described in Section 4. Implementation details and experimental results on challenging benchmarks are illustrated in Section 5. The paper is concluded in Section 6.

2 Related Work

The precipitation nowcasting task can be formulated as a spatio-temporal prediction problem in which both the input and the prediction target are spatio-temporal sequences. Convolutional neural networks (CNNs)[13] and recurrent neural networks (RNNs)[14] have been widely used for learning spatial correlations and temporal dependencies from a spatio-temporal sequence. Naturally, CNNs are applied to extract spatial features. As for temporal dynamics, the existing architecture can be generally divided into two types: 1) CNNs-based models which treat the spatio-temporal sequence as multiple channels[15, 16] or the depth dimension of the image[17] and apply CNNs to capture the spatial features; 2) RNNs-based models[7–11, 18–20] which use RNNs to learn the variations over time.

ConvLSTM[7] firstly integrates CNN and RNN together by replacing the fully connection with convolutional structures in both the input-to-state and state-to-state transitions in the long short-term memory recurrent neural network (LSTM). The proposal of ConvLSTM has become a milestone in the field of spatio-temporal prediction and most of the subsequent approaches are built upon ConvLSTM. Our model also employs ConvLSTM to capture the spatio-temporal correlations.

From the perspective of state transition functions, dynamic filter network (DFN)[21] extends ConvLSTM[7] by generating the filters between state transitions dynamically conditioned on input data. TrajGRU[8] actively learns the location-variant structure for recurrent connections and performs better on rotation motion. MIM[11] involves stacked multiple LSTM-similar blocks to replace the saturating forget gate in the LSTM unit and exploits the differential signals between adjacent recurrent states to model the non-stationary process. In the aspect of the encoder-forecaster architecture, PredRNN[9] adds a new memory (called spatio-temporal memory) cell in each LSTM unit which is allowed to zigzag through all RNN states across different RNN layers, aiming to extract and memorize spatial and temporal representations simultaneously.

In addition, E3D-LSTM[22] integrates 3D-convolution into RNNs to make local perceptrons of RNNs motion-aware. Conv-TT-LSTM[23] involves a higher-order convolution LSTM with a tensor-train module that combines convolutional features across time to learn long-term forecasting. Self-Attention ConvLSTM[24] introduces the self-attention mechanism into convolution LSTM to extract spatial features with both global and local dependencies. There are also efforts that extend the training data from 2D radar images to 3D images, while adapting the above ConvRNN models for the 3D-radar-extrapolation problem[25].

However, all above work considers the motion and deformation together, which may lose sight of fine-grained evolutions. To the best of our knowledge, only MCNet[10] decomposes the motion and content which independently captures the spatial layout and the corresponding temporal dynamics, but it does not consider the variation of the content such as accumulation, deformation, and dissipation. By contrast, our model is characterized to learn not only the tendency of global motion but also the variation of local deformation.

3 Preliminaries

3.1 Structured Sequence Predictive Learning

Sequence predictive learning is the problem of predicting the most likely future length-K sequence given the previous J observations:

$$\hat{\boldsymbol{x}}_{t+1}, ..., \hat{\boldsymbol{x}}_{t+K} = \underset{\boldsymbol{x}_{t+1},...,\boldsymbol{x}_{t+K}}{\operatorname{argmax}} \ p(\boldsymbol{x}_{t+1}, ..., \boldsymbol{x}_{t+K} | \boldsymbol{x}_{t-J+1}, ..., \boldsymbol{x}_t),$$

where $\boldsymbol{x}_t \in D$ is an observation at time t and D denotes the domain of the observed features. The structured sequence is a type of special sequences where features of the observations $\boldsymbol{x}_t$ are not independent but linked by pairwise or spatial relationships. Such structures may be regular grid-structured like the 2D radar echo map or graph-structured like the traffic network.

In this paper, we mainly focus on the regular grid-structured sequence predictive learning problem. $\boldsymbol{x}_t$ can be viewed as signals on an $M \times N$ grid which consists of M rows and N columns. Inside each cell in the grid, P measurements are varying over time. Thus, $\boldsymbol{x}_t$ can be represented by a tensor $\mathcal{X} \in \mathbb{R}^{P \times M \times N}$. For precipitation nowcasting, $\boldsymbol{x}_t$ is a 2D radar echo map.

3.2 Convolutional LSTM

Convolutional LSTM (ConvLSTM)[7] is a popular model for regular grid-structured sequences, which explicitly encodes the structured information into tensors by replacing the multiplications of dense matrices in classical LSTM with convolutions. The main equations of ConvLSTM are shown as follows:

$$\begin{aligned}
\boldsymbol{g}_t &= \tanh(\boldsymbol{W}_{xg} * \mathcal{X}_t + \boldsymbol{W}_{hg} * \mathcal{H}_{t-1} + \boldsymbol{b}_g), \\
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{xi} * \mathcal{X}_t + \boldsymbol{W}_{hi} * \mathcal{H}_{t-1} + \boldsymbol{W}_{ci} \odot \mathcal{C}_{t-1} + \boldsymbol{b}_f), \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{xf} * \mathcal{X}_t + \boldsymbol{W}_{hf} * \mathcal{H}_{t-1} + \boldsymbol{W}_{cf} \odot \mathcal{C}_{t-1} + \boldsymbol{b}_f), \\
\mathcal{C}_t &= \boldsymbol{f}_t \odot \mathcal{C}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t, \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{xo} * \mathcal{X}_t + \boldsymbol{W}_{ho} * \mathcal{H}_{t-1} + \boldsymbol{W}_{co} \odot \mathcal{C}_t + \boldsymbol{b}_o), \\
\mathcal{H}_t &= \boldsymbol{o}_t \odot \tanh(\mathcal{C}_t),
\end{aligned}$$

where $\sigma(\cdot)$ is the sigmoid activation function, and $*$ and $\odot$ denote the convolution operator and the Hadamard product, respectively. Here, $\boldsymbol{i}, \boldsymbol{f}$ and $\boldsymbol{o}$ are the input gate, forget gate and output gate respectively, $\boldsymbol{W}_{(\cdot)}$ are the parameters, $\boldsymbol{b}$ is the bias term, $\mathcal{X}$ is the input, $\mathcal{H}$ is the hidden state and $\mathcal{C}$ is the cell memory.

ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors, which is achieved by using a convolution operator in the state-to-state and input-to-state transitions. If the states are viewed as the hidden representations of moving objects, a ConvLSTM with a larger transitional kernel should be able to capture faster motions while one with a smaller kernel can capture slower motions[7]. ConvLSTM has been adopted as a building block in many complex structures[9–11] since its proposal. We also employ ConvLSTM as an elementary building block in our proposed FDNet architecture.

4 Model Architecture of FDNet

We take an end-to-end learning approach to predict future radar echo maps: given a dataset consisting of the previous J radar echo map sequences $\boldsymbol{x}_{t-J}, \boldsymbol{x}_{t-J+1}, ..., \boldsymbol{x}_t$ and the ground truth future length-K radar echo map sequences $\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+2}, ..., \boldsymbol{x}_{t+K}$, we train a network to predict the future length-K sequence directly from the previous length-J sequence. The overall architecture of the proposed Flow-Deformation Network (abbreviated as FDNet) is described in Fig.2.

FDNet is comprised of five components: position encoder, shape encoder, flow encoder, deformation encoder, and combination & decoder. The position encoder and the shape encoder extract meaningful position features and shape features of the radar echoes from a single frame, respectively. The flow encoder takes the spatial correspondence between two adjacent frames as an input, and feeds it into ConvLSTM layers to produce the hidden representation encoding the optical flow of the sequence. The deformation encoder takes the difference between the current moment shape representation and the predicted shape representation (computed from the previous shape representation), and the optical flow representation as an input, and outputs the hidden representation of the morphologic deformation. Finally, the combination & decoder takes the outputs from the deformation encoder and the shape encoder as inputs, and combines them to produce a pixel-level prediction of the next frame. The prediction of multiple frames can be achieved by recursively performing the above procedures over K time steps. We will describe the detailed configuration of the proposed architecture as follows.

4.1 Position Encoder and Shape Encoder

The position encoder extracts typical position features of pixels from a single frame in a sequence by

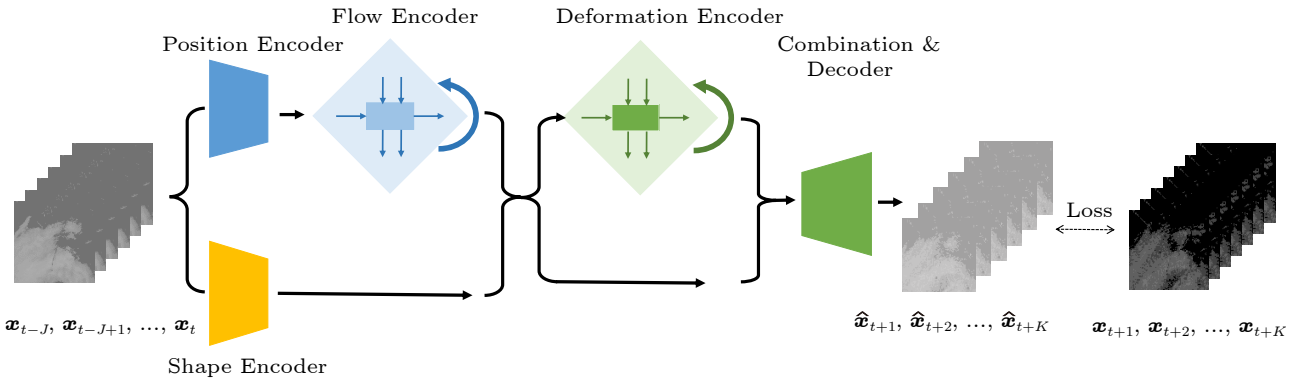


Fig.2. Architecture of FDNet. FDNet is a sequence-to-sequence trainable network architecture with flow and deformation separation to model the spatio-temporal dynamics for pixel-level future prediction in radar echo maps.

$\boldsymbol{m}_t = f^{\text{pos}}(\boldsymbol{x}_t)$, where $\boldsymbol{m}_t \in \mathbb{R}^{w \times h \times c}$ is the representation of the position feature of the current frame. f^{pos} is implemented by a CNN including convolution layers and activation layers.

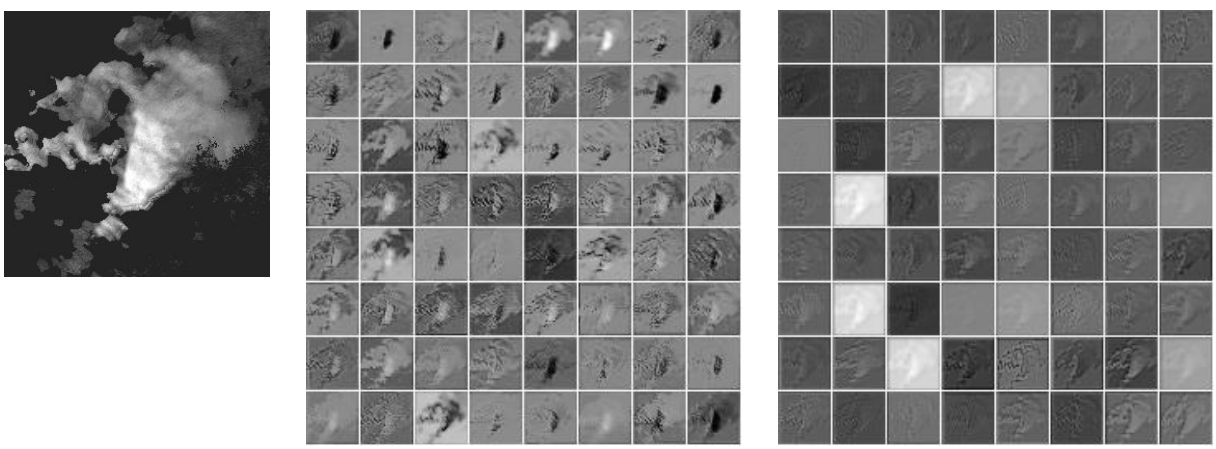
The shape encoder has the similar structure of the position encoder. It extracts important shape features from a single frame by $\boldsymbol{s}_t = f^{\text{shape}}(\boldsymbol{x}_t)$, where $\boldsymbol{s}_t \in \mathbb{R}^{w' \times h' \times c'}$ is the representation of the shape feature of the current frame. f^{shape} is also implemented by a CNN.

In order to examine whether the shape encoder and the position encoder can extract the characteristics of the shape and the position, Fig.3 visualizes the output feature maps from them. We can see that the model has learned to extract shape features and position features, and is working in the way as expected. The shape encoder learns the fine shape details including pixel light and shade, while the position encoder captures coarse localization and outline of the image.

4.2 Flow Encoder

The flow encoder aims to capture the optical flow field motion between consecutive frames without considering the deformation of the content in the frames. But how to capture the features of the flow effectively? TrajGRU[8] tackles the problem by stacking both input images together and feeds them through a rather generic convolutional neural network, allowing the network itself to decide how to process the image pair to extract the motion information. But it only roughly uses the previous time period optical flow to do the next time period transformation.

Considering that the optical flow is variable and the variation is coherent, it could be beneficial to design a strategy to extract the features of the optical flow and feed them into a time series modeling net-



(a) (b) (c)

Fig.3. Visualizations of the output from the shape encoder and the position encoder. (a) Original radar echo image. (b) Feature maps from the shape encoder. (c) Feature maps from the position encoder. From the output results of the position encoder, we can roughly see the outline of the radar echo. The shape encoder focuses more on pixel-level details, which is obviously different from the position encoder.

work to learn the variation. But what could be considered as the hidden representation of the optical flow? It is natural to take the correlation between two consecutive images directly as the hidden feature of the optical flow. However, how would the network find this correlation?

We take a similar "corr" function as that in [26] to perform multiplicative patch comparisons between two feature maps. Let $\boldsymbol{m}_{t-1} \in \mathbb{R}^{w\times h\times c}$, $\boldsymbol{m}_t \in \mathbb{R}^{w\times h\times c}$ be multi-channel feature maps in time step $t-1$ and time step t, respectively, and w, h and c are their width, height and the number of channels respectively. Given a maximum displacement d, for each location $p_{x,y}$ in the first feature map $\boldsymbol{m}_{t-1}$, the corr function will compute the correlations between the feature of $p_{x,y}$ and every point q in the region $[x-d, y-d]\times[x+d, y+d]$ in the second feature map $\boldsymbol{m}_t$. The "correlation" between point p_{x_1,y_1} in the first feature map and point q_{x_2,y_2} in the second feature map is defined as the dot product of feature vectors in point p_{x_1,y_1} and point q_{x_2,y_2}:

$$\texttt{corr}(p_{x_1,y_1}, q_{x_2,y_2}) = \boldsymbol{m}_{t-1}[x_1][y_1]\cdot\boldsymbol{m}_t[x_2][y_2].$$

Computing correlations among all patch combinations involves $w^2\times h^2\times d^2$ computations, which is time consuming. For computational reasons we limit the maximum displacement $d \approx w/3$. Specifically, if the size of the feature map is 64×64, we set $d=21$, and if the size of the feature map is 32×32, we set $d=11$. Besides, we introduce striding in both feature maps, and in our experiment we set the stride to 2.

The result produced by the corr function is three-dimensional: for every combination of two 2D positions we obtain a correlation value, and there are $(2d+1)^2$ combinations, and thus we obtain a final output of size $w\times h\times(2d+1)^2$.

The output of the corr function will be fed into a ConvLSTM (see Subsection 3.2) layer to encode the correlations. The advantage of such a structure is that we could learn the variation of the optical flow by learning the parameters of the ConvLSTM.

After that, we take a one-hidden-layer convolutional neural network to extract the optical flow. TrajGRU[8] claims that motion patterns have different neighborhood sets for different locations and therefore the "optical flow" should not be just one but a set. In our experiments, we test both single optical flow and multiple optical flows. But the results show that the single optical flow approach achieves a better performance.

As described in Fig.4(a), the flow encoder takes the position encoders $\boldsymbol{m}_{t-2}$ and $\boldsymbol{m}_{t-1}$ at time step $t-2$ and time step $t-1$ as input, respectively, and produces the hidden representations of the optical flow at time step t, which are denoted by $\boldsymbol{u}_{t,l}, \boldsymbol{v}_{t,l}$. Fig.5(a) visualizes the "flow" information that the flow encoder outputs. We can see that the network has learned reasonable optical flow field motion information.

4.3 Deformation Encoder

The evolution of the radar echo map sequence is a complex process. The shapes of radar echo maps may expand, contract, or change rapidly due to the complex atmospheric environment, which is quite different from that of other video sequences like moving digits[7] and KTH actions[9]. Therefore, it is necessary to distinguish the changes of shapes from the translational motions of radar echoes, and this idea inspires the design of the deformation encoder.

But what could be considered as the hidden representation of the deformation? In our study, the transformation of two adjacent radar echo maps is divided into two aspects: the optical flow fields movement and the shape deformation. Therefore the shape feature of the former image is conducted by a "warp" function which is the same with that in [8] to let the precipitation field move along the estimated optical flow that the flow encoder outputs, and then the deformation is computed by the difference between the result and the shape feature of the latter image.

Let $\boldsymbol{s}_{t-1}$ be the shape feature of the image at time step $t-1$, and $\boldsymbol{u}_{t,l}, \boldsymbol{v}_{t,l}$ be l optical flows that the flow encoder outputs. $warp(\boldsymbol{s}_{t-1}, \boldsymbol{u}_{t,l}, \boldsymbol{v}_{t,l})$ selects the positions pointed out by $\boldsymbol{u}_{t,l}, \boldsymbol{v}_{t,l}$ from $\boldsymbol{s}_{t-1}$ via the bilinear sampling kernel[12, 27]. If we denote $\mathcal{M} = warp(\mathcal{I}, \boldsymbol{U}, \boldsymbol{V})$ where $\mathcal{M}, \mathcal{I} \in \mathbb{R}^{c\times h\times w}$ and $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{h\times w}$, we have:

$$\mathcal{M}_{c,i,j} = \sum_{m=1}^{h}\sum_{n=1}^{w}\mathcal{I}_{c,m,n}\times\max(0, 1-|i+V_{i,j}-m|)\times \max(0, 1-|i+U_{i,j}-n|).$$

Here, $\mathcal{M}_{c,i,j}$ is an element of $\mathcal{M}$, and $U_{i,j}$, $V_{i,j}$ are elements of optical flows $\boldsymbol{U}, \boldsymbol{V}$, respectively. $\mathcal{M}, \mathcal{I}$ are output and input feature maps of "warp" function, respectively, and $\mathcal{I}_{c,\,m,\,n}$ is an element of $\mathcal{I}$.

We denote the output of the warp function on $\boldsymbol{s}_{t-1}$ as $\boldsymbol{w}_t$. Intuitively, $\boldsymbol{w}_t$ can be viewed as the inferred

shape feature at time step t, which is generated by transforming $\boldsymbol{s}_{t-1}$ using the predicted optical flow $\boldsymbol{u}_{t,l}, \boldsymbol{v}_{t,l}$. We deliver $\boldsymbol{w}_t$ and shape features of the image at time step t to a "`diff`" function, which is defined as element-wise subtraction. The output of the `diff` function will be fed into a stack of the ConvLSTM layers to encode the deformation, denoted as $\boldsymbol{d}_{t+1}$, as described in Fig.4(b). We use the hidden features out from the ConvLSTM layers rather than the `diff` function as the encoder of the deformation, aiming to learn the evolution of the deformation over time.

The output feature maps of the deformation encoder are visualized in Fig.5(b). It can be observed that the deformation encoder captures radar echo changes at different hierarchies. For example, the feature maps in orange boxes pay more attention to the darkest pixels, while the feature maps in red boxes are sensitive to the pixel changes in multiple scattered areas.

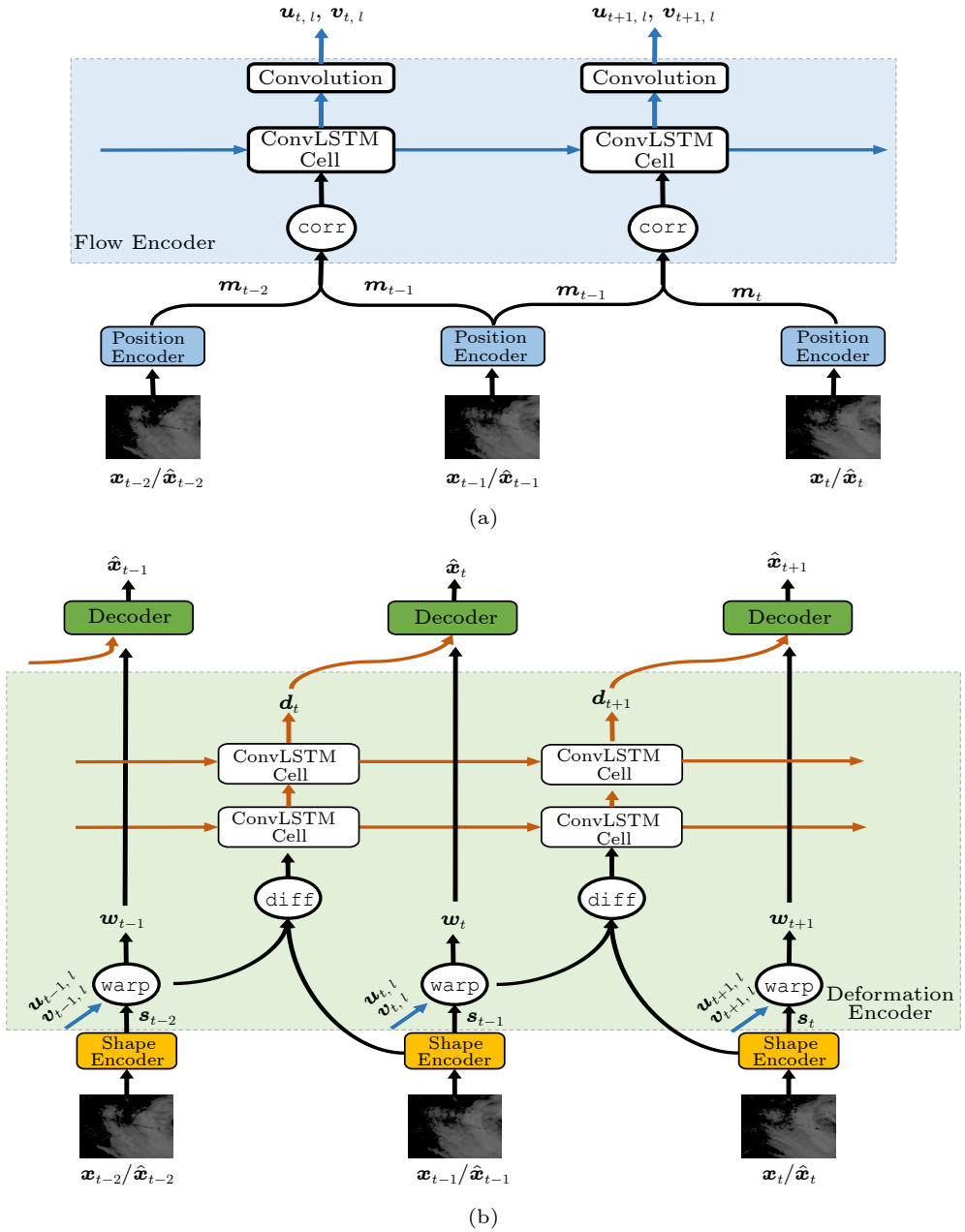
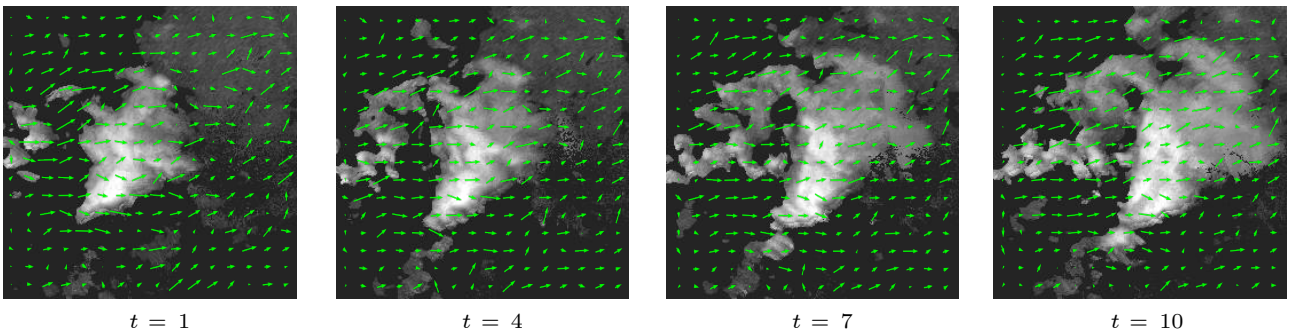
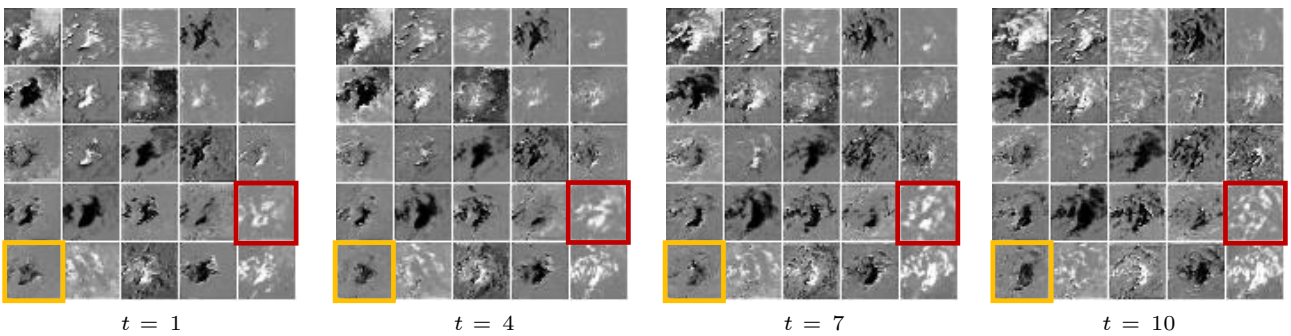


Fig.4. Detail of FDNet. (a) Position encoder and flow encoder. Blue arrows carry optical flow field motion information. (b) Shape encoder, deformation encoder, and combination & decoder. Brown arrows carry morphologic deformation information. The input can be either the ground truth frame for the input sequence, or the generated frame at the previous time step. One frame is generated at each time step as the output.



(a)



(b)

Fig.5. Visualization of the output from the flow encoder and the deformation encoder. (a) Visualization of the optical flow field motion that the network has learned. From left to right, we select the radar echo images at $t = 1$, 4, 7, and 10 respectively. It can be seen that the optical flow field information learned by the model is basically consistent with the actual motion of the radar echoes. (b) Visualization of output feature maps of the deformation encoder. We select the corresponding feature maps with the same time step as that in (a).

4.4 Combination & Decoder

The outputs from the above deformation encoder, $\boldsymbol{w}_t$ and $\boldsymbol{d}_t$, encode high-level representation of the inferred image after optical flow field movement and shape deformation, respectively. Given the representation, the goal of the decoder is to generate a pixel-level prediction of the next frame $\hat{\boldsymbol{x}}_{t+1} \in \mathbb{R}^{w \times h \times c}$. For this purpose, it first combines these two representations into a unified representation by $\boldsymbol{c}_t = g^{\text{comb}}([\boldsymbol{d}_t, \boldsymbol{w}_t])$, where $[\boldsymbol{d}_t, \boldsymbol{w}_t] \in \mathbb{R}^{w' \times h' \times 2c'}$ denotes the concatenation of $\boldsymbol{d}_t$ and $\boldsymbol{w}_t$ in the depth dimension, and $\boldsymbol{c}_t \in \mathbb{R}^{w' \times h' \times c'}$ denotes the combined high-level representation of optical fields motion. g^{comb} is implemented by a CNN layer.

Then $\boldsymbol{c}_t$ is delivered to the decoder, which places $\boldsymbol{c}_t$ back into the original pixel space by $\hat{\boldsymbol{x}}_{t+1} = g^{\text{dec}}(\boldsymbol{c}_t)$.

We employ the deconvolution network[28] for our decoder network g^{dec}, which is composed of multiple successive operations of deconvolution, rectification, and convolution.

5 Experiments

5.1 Dataset Description

We verify our model on two real-world radar echo datasets, HKO-7[8] and SRAD2020[①], collected by Hong Kong Observatory and Shenzhen Meteorological Bureau, respectively. Each dataset contains radar CAPPI reflectivity images recorded every six minutes, as detailed below.

5.1.1 HKO-7

The HKO-7 dataset contains radar echo data from 2009 to 2015 collected by Hong Kong Observatory. The radar CAPPI reflectivity images, which have resolution of 480 × 480 pixels, are taken from an altitude of 2 km and cover an area of 512 km × 512 km centered in Hong Kong. The raw logarithmic radar reflectivity factors are linearly transformed to pixel values via $pixel = \lfloor 255 \times (R + 10)/70 + 0.5 \rfloor$ where R is

[①] https://competition.huaweicloud.com/information/1000040092/circumstance?track=107, July 2023.

the radar CAPPI reflectivity factor and are clipped to be between 0 and 255. The dataset contains 800 days for training, 50 days for validation and 120 days for test. We use the previous five time steps radar echo maps to predict 20 time steps into the future, covering the next two hours. The distribution of different rainfall intensities of the data is shown in Table 1.

Table 1. Rain Rate Statistics of the HKO-7 Dataset

Rain Rate (mm/h)	Proportion (%)	Rainfall Level
$0 \leqslant x < 0.5$	89.94	No/hardly noticeable
$0.5 \leqslant x < 2$	4.32	Light
$2 \leqslant x < 5$	2.51	Light to moderate
$5 \leqslant x < 10$	1.63	Moderate
$10 \leqslant x < 30$	1.17	Moderate to heavy
$x \geqslant 30$	0.43	Rainstorm warning

Note: x means the rain rate.

5.1.2 SRAD2020

The SRAD2020 dataset contains radar echo data collected by Shenzhen Meteorological Bureau in recent years. The radar CAPPI reflectivity images which have the resolution of 256 × 256 pixels are taken from an altitude of 2.5 km and covering an area of 255 km × 255 km. There are 20 000 radar echo data cases in the SRAD2020 dataset, each of which covers four hours and is a 41-length radar echo sequence. We filter the noisy data cases which have abrupt all-zero radar echo data in a sequence. The final dataset has 15 939 data cases for training, 1 002 data cases for validation and 2 541 data cases for test. We predict 20 time steps into the future, covering the next two hours by observing 21 time steps. The statistical distribution of radar reflectivity values in the dataset is shown in Table 2.

5.2 Experimental Setting

5.2.1 Evaluated Algorithms

We compare our method with five competitive precipitation nowcasting models (including ConvLSTM[7], TrajGRU[8], PredRNN[9], MIM[11] and ConvTTLSTM[23]). We use one layer ConvLSTM with 128 hidden states for the flow encoder and two layers of ConvLSTM with 128 hidden states for the deformation encoder in FDNet. For the ConvLSTM model and the TrajGRU model, we use a 3-layer encoding-forecasting structure with the number of hidden states for the RNNs set to 64, 192, 192 as in [8]. We

Table 2. Radar Reflectivity Statistics of the SRAD2020 Dataset

Radar Reflectivity Factor	Proportion (%)	Rainfall Level
$0 \leqslant x < 20$	77.53	No to light
$20 \leqslant x < 30$	11.35	Light to moderate
$30 \leqslant x < 40$	8.55	Moderate to heavy
$40 \leqslant x < 50$	2.37	Heavy to rainstorm
$50 \leqslant x \leqslant 80$	0.20	Large rainstorm to extraordinary rainstorm

Note: x means the radar reflectivity factor.

also use a stack of three layers of ST-LSTM with 128 hidden states for the PredRNN model and the MIM model. For the Conv-TT-LSTM model, we use a stack of eight layers of Conv-TT-LSTM with 64 hidden states for the first layer and 128 hidden states for the other layers. The order of CTTD[23] is set to 3, the rank of CTTD is set to 8 and the time step of CTTD is set to 3 for the Conv-TT-LSTM model. The convolution filters inside ConvLSTMs, ST-LSTMs and Conv-TT-LSTMs are all set to 3×3. Since the ConvLSTM model and the TrajGRU model reduce the size of feature maps at higher layers, for fair comparison, we use dilated convolution at higher layers to get a larger receptive field for FDNet, PredRNN, MIM and Conv-TT-LSTM.

Considering the memory and computation factors, we conduct downsampling on the original images to reduce the resolution, and after prediction upsampling is used to restore the resolution. The 2D-CNN encoders and the 2D-CNN decoders are similar in FDNet, PredRNN, MIM and Conv-TT-LSTM. Table 3 shows the details of the encoder and the decoder.

5.2.2 Loss Function

We use an objective function composed of multiple weighted losses for all models. Given the training data $D = \{\boldsymbol{x}^{(n)}_{1,\ldots,t,\ldots,T}\}_{n=1}^{N}$ where $\boldsymbol{x}^{(n)}_t \in \mathbb{R}^{w \times h}$, we use $\boldsymbol{x}^{(n)}_1, \ldots, \boldsymbol{x}^{(n)}_t$ to predict $\boldsymbol{x}^{(n)}_{t+1}, \ldots, \boldsymbol{x}^{(n)}_T$. Our model is trained to minimize the prediction loss by

$$\begin{aligned}\mathcal{L} &= \sum_{n=1}^{N} \mathcal{L}_{\text{img}}(\boldsymbol{x}^{(n)}, \hat{\boldsymbol{x}}^{(n)}) \\ &= \lambda_{\text{pixel}} \mathcal{L}_{\text{pixel}}(\boldsymbol{x}^{(n)}, \hat{\boldsymbol{x}}^{(n)}) + \lambda_{\text{gdl}} \mathcal{L}_{\text{gdl}}(\boldsymbol{x}^{(n)}, \hat{\boldsymbol{x}}^{(n)}).\end{aligned}$$

Here, $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are the target and predicted frames, respectively. λ_{pixel} and λ_{gdl} are hyperparameters that control the effect of each sub-loss during optimization. $\mathcal{L}_{\text{pixel}}$ is designed to guide the model to

Table 3. Details of the Encoder and the Decoder

Module	Layer	Kernel	Stride	Padding	Output-Padding	Channel I/O	Type
Encoder	econv1	3×3	2×2	1×1	-	1/8	2D-Conv, GroupNorm, LeakyReLU
	econv2	3×3	1×1	1×1	-	8/16	2D-Conv, GroupNorm, LeakyReLU
	econv3	3×3	2×2	1×1	-	16/32	2D-Conv, GroupNorm, LeakyReLU
	econv4	3×3	1×1	1×1	-	32/32	2D-Conv, GroupNorm, LeakyReLU
	econv5	3×3	2×2	1×1	-	32/64	2D-Conv, GroupNorm, LeakyReLU
	econv6	3×3	1×1	1×1	-	64/64	2D-Conv, GroupNorm, LeakyReLU
Decoder	dconv1	3×3	1×1	1×1	1	128/64	Transposed 2D-Conv, GroupNorm[②], LeakyReLU[29]
	dconv2	3×3	2×2	1×1	2	64/32	Transposed 2D-Conv, GroupNorm, LeakyReLU
	dconv3	3×3	1×1	1×1	1	32/32	Transposed 2D-Conv, GroupNorm, LeakyReLU
	dconv4	3×3	2×2	1×1	2	32/16	Transposed 2D-Conv, GroupNorm, LeakyReLU
	dconv5	3×3	1×1	1×1	1	16/8	Transposed 2D-Conv, GroupNorm, LeakyReLU
	dconv6	3×3	2×2	1×1	1	8/1	Transposed 2D-Conv

Note: "-" means no padding, "2D-Conv" means 2D-Convolution, "Transposed 2D-Conv" means Transposed 2D-Convolution and "GroupNorm" means Group Normalization.

match the average pixel values directly, while $\mathcal{L}_{\text{gdl}}$ is expected to guide the model to match the gradients of such pixel values, to alleviate the image blurring tendency of predicted frames. In our experiments, we set $\lambda_{\text{pixel}}=1$ and $\lambda_{\text{gdl}}=1$, which means that these two aspects have the same effect.

As shown in Fig.1 and Fig.2, the frequencies of different rainfall levels are highly imbalanced. It is known that heavier rainfalls always have much greater social impacts. Since it has been proved that training with the balanced loss function is essential for good nowcasting performance of heavier rainfall[8], we use the weighted loss function in the sub-loss $\mathcal{L}_{\text{pixel}}$. Therefore we get

$$\mathcal{L}_{\text{pixel}}(\boldsymbol{y},\boldsymbol{z})=\sum_{p}\sum_{k=1}^{T}\sum_{i,\,j}^{w,\,h}w_{k,i,j}||y_{k,i,j}-z_{k,i,j}||_p^p,$$

where $\boldsymbol{y},\boldsymbol{z}\in\mathbb{R}^{T\times w\times h}$ are two T-length frame sequences, $y_{k,i,j}$ is the (i,j)-th pixel value in the k-th frame in $\boldsymbol{y}$, and $z_{k,i,j}$ is the (i,j)-th pixel value in the k-th frame in $\boldsymbol{z}$. $w_{k,i,j}$ is the weight corresponding to the (i,j)-th pixel in the k-th frame, which is related to its rainfall intensity or radar reflectivity value.

In our experiments, we set $p=1,2$ respectively, and thus we get

$$\mathcal{L}_{\text{pixel}}(\boldsymbol{y},\boldsymbol{z})=\sum_{k=1}^{T}\sum_{i,\,j}^{w,\,h}w_{k,i,j}(|y_{k,i,j}-z_{k,i,j}|+|y_{k,i,j}-z_{k,i,j}|^2).$$

As for the HKO-7 dataset, if we define r as the corresponding rainfall intensity of pixel x, we set the weight of pixel x as:

$$w(r)=\begin{cases}1, & \text{if } r<2,\\ 2, & \text{if } 2\leqslant r<5,\\ 5, & \text{if } 5\leqslant r<10,\\ 10, & \text{if } 10\leqslant r<30,\\ 30, & \text{if } r\geqslant 30.\end{cases}\tag{1}$$

As for the SRAD2020 dataset, we set the weight of pixel x as:

$$w(x)=\begin{cases}1, & \text{if } 0\leqslant x<20,\\ 2, & \text{if } 20\leqslant x<30,\\ 5, & \text{if } 30\leqslant x<40,\\ 10, & \text{if } 40\leqslant x<50,\\ 30, & \text{if } 50\leqslant x\leqslant 80.\end{cases}\tag{2}$$

However, both $\mathcal{L}_1$ loss and $\mathcal{L}_2$ loss produce blurry predicted frames and the generated images are getting much blurrier in longer future predictions. To sharpen the image prediction, we take a gradient difference loss (GDL)[15] as a mitigation strategy. The GDL loss $\mathcal{L}_{\text{gdl}}(\boldsymbol{y},\boldsymbol{z})$ is defined as:

$$\mathcal{L}_{\text{gdl}}(\boldsymbol{y},\boldsymbol{z})=\sum_{k=1}^{T}\sum_{i,\,j}^{w,\,h}(|(|y_{k,i,j}-y_{k,i-1,j}|-|z_{k,i,j}-z_{k,i-1,j}|)|^{\lambda}+|(|y_{k,i,j}-y_{k,i,j-1}|-|z_{k,i,j}-z_{k,i,j-1}|)|^{\lambda}),$$

where $\lambda\geqslant 1$ and λ is an integer. Considering the training time, we only set $\lambda=1$, which means that we only take the differencing between adjacent pixels.

We also conduct experiments without GDL loss on the SRAD2020 dataset, and the results show that the GDL loss improves the CSI score and the HSS score by 0.007 for all models on average.

5.2.3 *Optimizer and Learning Rate*

We train all compared models using Pytorch and

[②] https://arxiv.org/pdf/1803.08494.pdf, Sept. 2023.

optimize them to converge using ADAM[30]. For ConvLSTM, TrajGRU, PredRNN and MIM, we set the initial learning rate to 0.001, and for FDNet, we set the initial learning rate to 0.000 1. We use gradient clipping with clipping value of 50 for these models. As Su *et al.*[23] reported, we find that the Conv-TT-LSTM model is unstable at a high learning rate such as 0.001, but learns poorly at a low learning rate such as 0.000 1. Therefore we use gradient clipping with the learning rate of 0.001 and the clipping value of 1 as in [23].

We apply the scheduled sampling strategy[31] to all of the models, to gently change the training process from a fully guided scheme using the true previous data, towards a less guided scheme which mostly uses the generated data instead.

5.2.4 *Parameter Initialization*

All convolutional kernels are initialized by Xavier's normalized initializer[32], and the initial hidden/cell states in ConvLSTM, ST-LSTM and Conv-TT-LSTM are initialized as zeros.

5.3 Experimental Results

To evaluate the performance of our model, we measure the balanced mean square error (BMSE) and balanced mean average error (BMAE)[8]. BMSE is defined as:

$$BMSE = \frac{1}{N}\sum_{k=1}^{N}\sum_{i,\ j}^{w,\ h} w(x_{k,i,j})(x_{k,i,j} - \hat{x}_{k,i,j})^2,$$

and BMAE is defined as:

$$BMAE = \frac{1}{N}\sum_{k=1}^{N}\sum_{i,\ j}^{w,\ h} w(x_{k,i,j})|x_{k,i,j} - \hat{x}_{k,i,j}|,$$

where N is the total number of frames and $w(\cdot)$ is the weight defined in (1) and (2).

We also calculate the critical success index (CSI) and Heidke skill score (HSS) for multiple thresholds that correspond to different rainfall levels. We choose to use the thresholds of 20 dBZ, 30 dBZ, 40 dBZ and 50 dBZ respectively. The CSI score is defined as $CSI = TP/(TP + FN + FP)$ and the HSS score is defined as

$$HSS = (TP \times TN - FN \times FP)/((TP + FN)\times (FN + TN) + (TP + FP)(FP + TN)),$$

where TP means true positive, FP means false positive, TN means true negative and FN means false negative. A higher CSI score or a higher HSS score indicates a better prediction result.

Table 4 shows the comparisons of different approaches for the BMSE score of the average of all 20 time steps (up to two hours), 30 minutes, 60 minutes, 90 minutes, and two hours ahead precipitation on both datasets. We can observe that FDNet outperforms the compared models on the average BMSE of 20-time-step predictions. Besides, though FDNet is not the best one for short-term prediction (30 minutes), it shows a stable superiority for long-term predictions.

Table 5 shows the comparisons of different methods under CSI, HSS, BMSE and BMAE on the SRAD2020 dataset. It is worth noting that the CSI and HSS scores for thresholds of 40 dBZ and 50 dBZ reflect whether the models' predictions of heavy precipitation are accurate, and therefore it is often more concerned by meteorological experts. FDNet performs the best under the CSI and HSS score for the thresholds of 30 dBZ, 40 dBZ and 50 dBZ. We also give a frame-wise comparison of the HSS score under thresholds of 30 dBZ, 40 dBZ and 50 dBZ in Fig.6. HSS-40 and HSS-50 indicate the probabilities of severe weather conditions. FDNet performs better than baseline methods in most cases (except for MIM under HSS-50 at time steps 4–9).

Table 4. Performance Comparisons of Different Approaches for Precipitation Nowcasting

Model	HKO-7					SRAD2020				
	AVG	30 min	60 min	90 min	120 min	AVG	30 min	60 min	90 min	120 min
ConvLSTM[7]	5 806.72	3 714.66	5 861.75	7 609.86	9 120.39	1 753.22	**1 174.59**	1 801.63	2 284.70	2 669.52
TrajGRU[8]	5 818.12	3 717.82	5 872.09	7 619.20	9 170.61	1 721.95	1 197.60	1 767.28	2 273.62	2 546.11
PredRNN[9]	5 785.60	**3 698.48**	5 865.60	7 597.90	9 048.29	1 730.27	1 195.47	1 777.83	2 219.29	2 623.05
MIM[11]	5 784.17	3 701.48	5 854.50	7 571.12	9 045.05	1 835.40	1 292.99	1 885.64	2 336.53	2 713.84
Conv-TT-LSTM[23]	6 104.29	4 012.97	6 096.10	7 852.03	9 373.00	1 970.42	1 312.84	2 013.77	2 549.43	2 993.48
FDNet	**5 781.21**	3 716.69	**5 842.25**	**7 564.48**	**9 043.71**	**1 671.10**	1 174.67	**1 722.38**	**2 126.85**	**2 504.76**

Note: We take BMSE as the metrics for all these settings. AVG means the average BMSE of 20 time steps predictions. A lower value means a better prediction performance. We mark the best result within a specific setting in bold.

Table 5. Performance Comparisons of Different Approaches under CSI, HSS, BMSE, and BMAE on the SRAD2020 Dataset

Model	CSI↑				HSS↑				BMSE↓	BMAE↓
	$20 \leqslant x < 30$	$30 \leqslant x < 40$	$40 \leqslant x < 50$	$x \geqslant 50$	$20 \leqslant x < 30$	$30 \leqslant x < 40$	$40 \leqslant x < 50$	$x \geqslant 50$		
ConvLSTM[7]	**0.604 8**	0.456 1	0.218 6	0.011 8	**0.666 5**	0.556 3	0.333 3	0.021 7	1 753	8 659
TrajGRU[8]	0.578 2	0.439 3	0.207 8	0.025 7	0.646 1	0.552 6	0.321 2	0.047 3	1 721	**8 540**
PredRNN[9]	0.595 5	0.450 7	0.217 2	0.034 2	0.652 5	0.554 2	0.329 2	0.061 4	1 730	8 769
MIM[11]	0.584 3	0.451 2	0.221 6	0.046 4	0.635 9	0.554 5	0.334 7	0.083 9	1 835	9 151
Conv-TT-LSTM[23]	0.576 2	0.428 2	0.203 2	0.010 7	0.638 4	0.543 4	0.271 3	0.020 0	1 920	9 016
FDNet	0.597 7	**0.457 7**	**0.226 0**	**0.050 9**	0.660 3	**0.566 5**	**0.341 5**	**0.091 0**	**1 671**	8 698

Note: Each cell contains the mean score of the 20 predicted frames. "↑" means that the higher the score, the better. "↓" means that the lower the score, the better. "$x \geqslant \tau$" means the skill score at the τ dBZ. We mark the best result within a specific setting in bold.

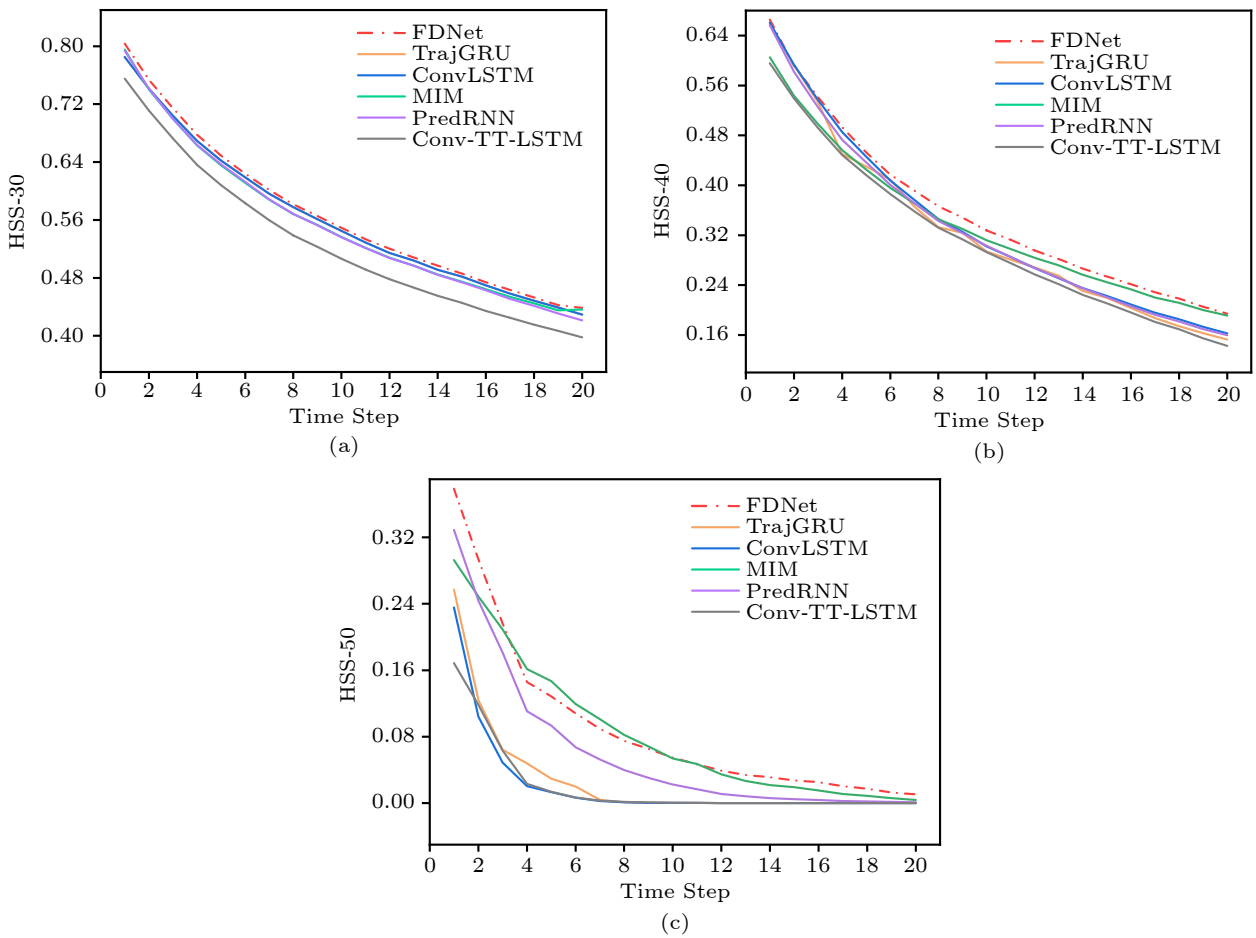


Fig.6. Frame-wise comparison of the HSS score on the SRAD2020 dataset. Higher curves indicate better forecasting results. Our FDNet performs better than baselines in most cases (except for MIM under HSS-50 at time steps 4–9). (a) HSS-30. (b) HSS-40. (c) HSS-50.

5.4 Ablation Studies

We perform four ablation studies to analyze the respective contribution of each module in our model.

1) *Necessity of the Shape Encoder and the Position Encoder.* In order to verify the necessity of separately extracting shape features and position features of input frames, we conduct an ablation experiment in which only one encoder is used to extract features of input frames. The results of group 1 in Table 6 show that without separating the encoder the model performs very poorly.

2) *Sensitivity of Our Model to the Number of ConvLSTM Layers in the Flow Encoder.* We evaluate models with 0/1/2 ConvLSTM layer(s) in the flow encoder in group 2 as shown in Table 6. It shows that

Table 6. Experimental Results Under Ablated Settings

Group	Separating Encoder	no.f	no.d	out.f	out.d	BMSE	HSS-30	HSS-40	HSS-50
1	✘	1	1	✔	✔	2 323	0.49	0.11	0.009
	✔	1	1	✔	✔	1 864	0.55	0.33	0.070
2	✔	0	1	✔	✔	1 898	0.55	0.33	0.070
	✔	1	1	✔	✔	1 864	0.55	0.33	0.070
	✔	2	1	✔	✔	1 863	0.55	0.33	0.060
3	✔	1	0	✔	✔	2 245	0.47	0.17	0.050
	✔	1	1	✔	✔	1 864	0.55	0.33	0.070
	✔	1	2	✔	✔	1 671	0.56	0.34	0.090
	✔	1	3	✔	✔	1 934	0.54	0.33	0.070
4	✔	1	✘	✔	✘	2 587	0.45	0.15	0.010
	✔	1	2	✔	✘	1 690	0.55	0.33	0.060
	✔	1	2	✘	✔	1 903	0.53	0.30	0.150
	✔	1	2	✔	✔	1 671	0.56	0.34	0.090

Note: We use BMSE, HSS-30, HSS-40 and HSS-50 to measure the prediction quality. We conduct four groups of comparisons. In this table, "no.f" means the number of ConvLSTM layers in the flow encoder; "no.d" means the number of ConvLSTM layers in the deformation encoder, and "✘" in "no.d" means that we reduce the deformation modeling branch; "out.f" denotes if the output from the flow modeling branch is delivered to the decoder, and "out.d" denotes if the output from the deformation modeling branch is delivered to the decoder.

the proposed model with one ConvLSTM layer in the flow encoder performs the best.

3) *Effect of the Number of ConvLSTM Layers in the Deformation Encoder.* We also test the effect of the number of ConvLSTM layers in the deformation encoder. We evaluate models with 0/1/2/3 ConvLSTM layer(s) in group 3 as shown in Table 6. The result shows that two ConvLSTM layers get the best score. It is a trade-off: applying too few layers leads to inadequate deformation modeling capability while the excessively deep recurrent model leads to training difficulty.

4) *Contribution of Flow Modeling Branch and Deformation Modeling Branch.* To see how the flow modeling branch and the deformation modeling branch contribute to the final result, we conduct a series of experiments, by 1) removing the deformation modeling branch from the proposed model (omitting the orange branch in Fig.4(b)), 2) only delivering the output from the flow modeling branch ($\boldsymbol{w}_t$ inFig.4(b)) to the decoder, and 3) feeding the concatenation of the output from the deformation modeling branch and the shape encoder to the decoder (concatenation of $\boldsymbol{d}_{t+1}$ and $\boldsymbol{s}_t$ in Fig.4(b)). The difference between settings 1 and 2 is that the latter still retains the deformation modeling information to iterate the prediction of the next step. The corresponding results are shown in group 4 of Table 6. The poor performance of the setting without the deformation encoder indicatesthat the deformation modeling branch is very important to the proposed model. The behavior of the restruction from the flow branch is the most similar to the proposed model, but there still remains a little gap in large rainstorm conditions (> 50 dBZ). The advantage is that the deformation branch tends to forecast heavier rainfall. Therefore the concatenation of these two complementary branches is necessary and achieves better results.

5.5 Results Visualization

We visualize four typical sequences of the predicted radar echo map in Figs.7–10, respectively. The first line shows the original radar echo map sequence. We compare the predicted results of our model and five baseline methods. In order to see how our two proposed pathways work, we also give the visualizations of partial predictions labelled with "flow branch restruction" and "deformation branch restruction". This is done by delivering the output from the flow encoder ($\boldsymbol{w}_t$ in Fig.4(b)) to the decoder and the concatenation of the output from the deformation encoder and the shape encoder to the decoder (concatenation of $\boldsymbol{d}_{t+1}$ and s_t in Fig.4(b)), respectively. All compared models take 21 historical radar echo images as inputs, and predict the next 20 images (radar echo maps for the next two hours).

It can be seen that the flow encoder mainly focuses on the movement of radar echoes as a whole, and can remember the general outline and scope. It helps the model avoid extreme expansions of radar echoes in prediction which appear in all baseline methods (see Fig.7 and Fig.8). The deformation encoder is very sensitive to pixel changes, especially the information lost during the "`warp`" procedure. Therefore it helps the proposed model remember the heavy rainfall information for a long time (see Fig.8 and Fig.9), while the other models all lose this information step by step when modeling motion and transformation together. The deformation branch also equips the model with the capability of predicting new coming rainfall (see Fig.10). Overall, benefiting from these two separate encoding pathways, FDNet's results are not only sharp enough but also more deterministic in future predictions.

6 Conclusions

In this paper, we proposed a flow-deformation network FDNet for precipitation nowcasting. FDNet employs two parallel cross encoding pathways and learns

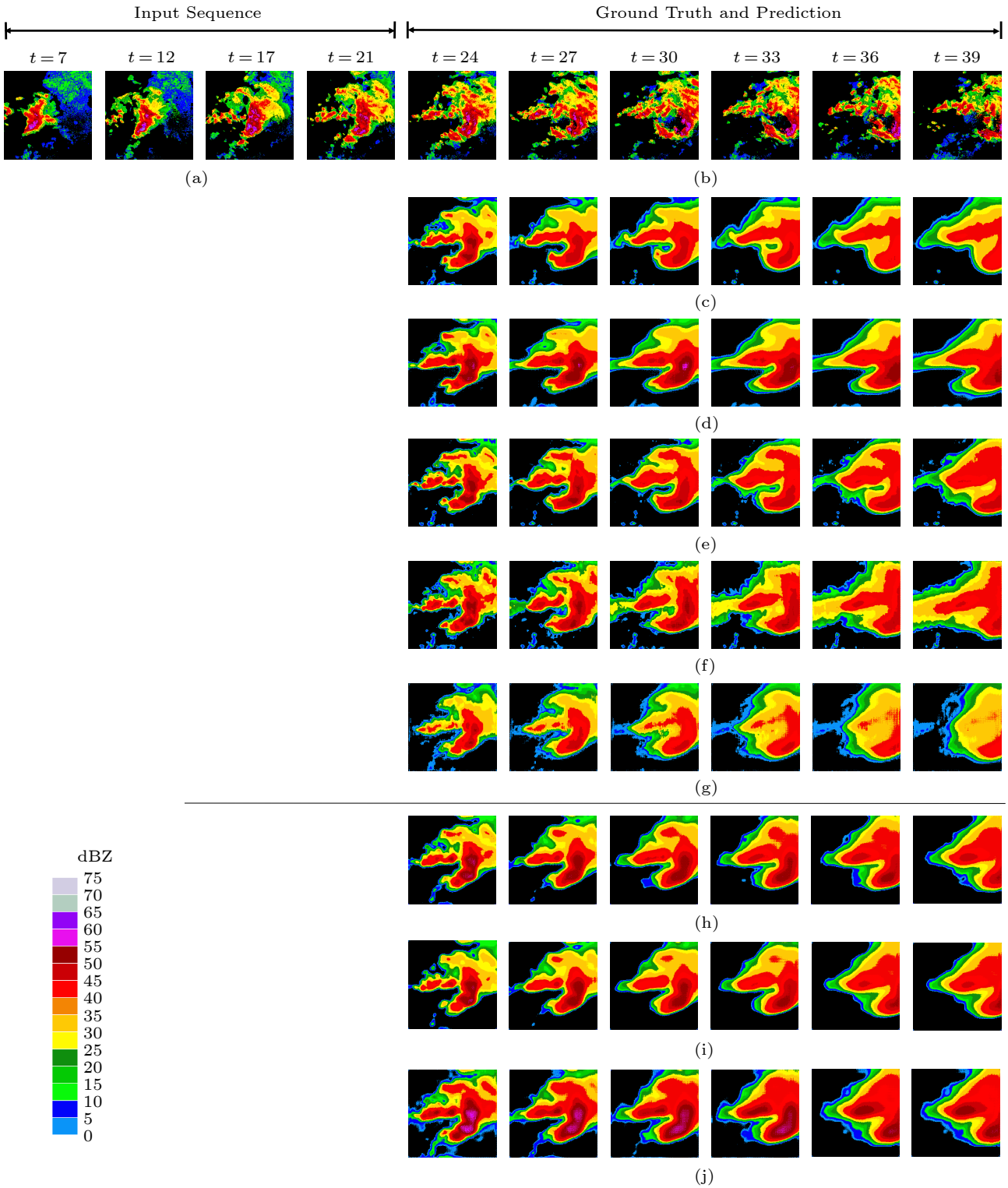


Fig.7. Showcase 1: an example of overall movement of radar echoes. The pixels tend to move from left to right as a whole. Though all these models try to predict the sequence under this movement, the baseline models above the line all suffer from a distinct blur effect with a long tail, especially the frames in MIM. (a) Input sequence. (b) Ground truth. (c) ConvLSTM. (d) TrajGRU. (e) PredRNN. (f) MIM. (g) Conv-TT-LSTM. (h) FDNet. (i) Flow branch restruction. (j) Deformation branch restruction.

to decompose optical flow field motion and morphologic deformation of radar echoes. The model consists of a flow encoder to capture tendencies of the overall displacement of radar echoes, and a deformation encoder to perceive the variations of the deformation. Experimental results suggested that separate model-

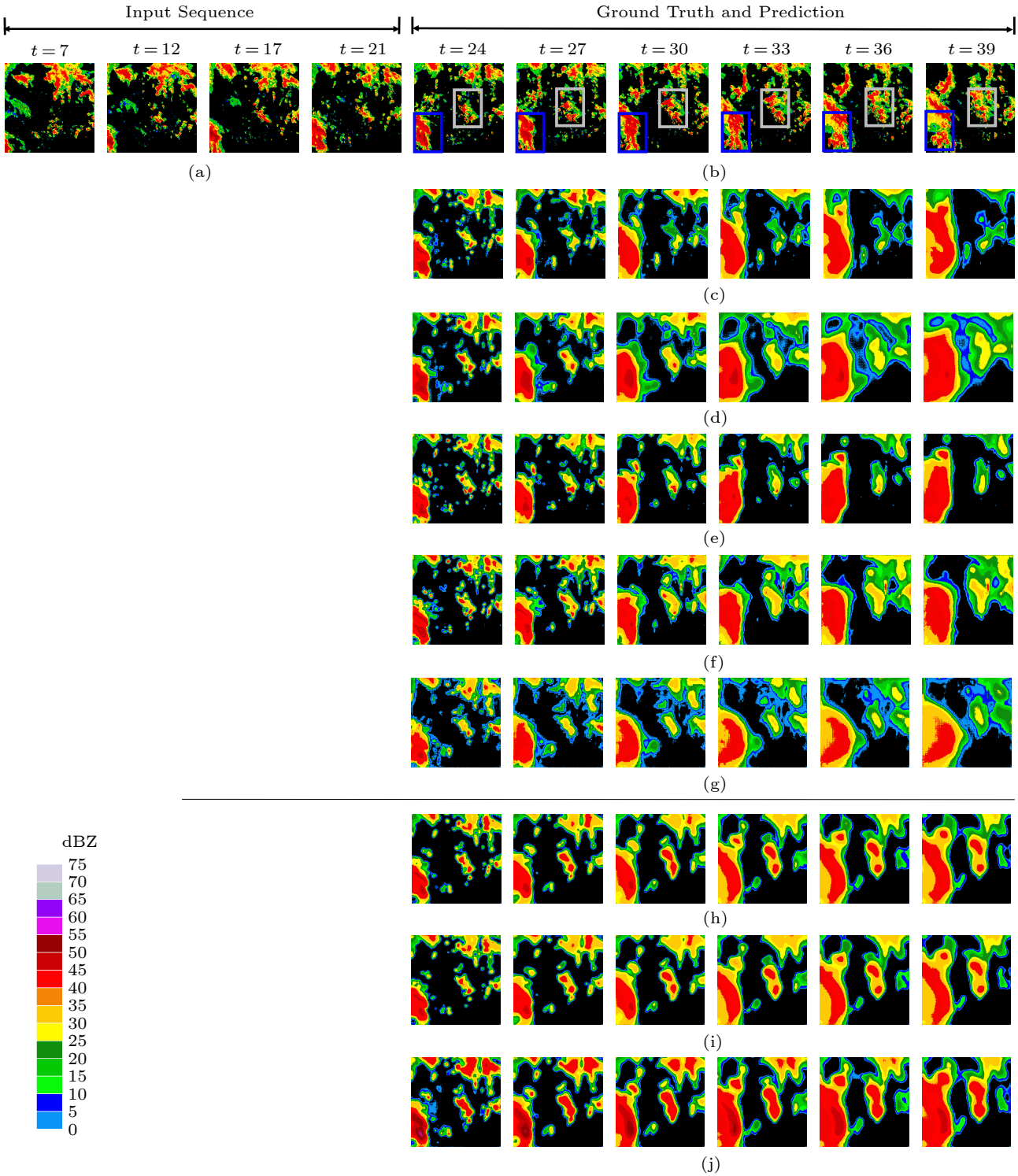


Fig.8. Showcase 2: a challenging example with accumulation and dissipation of radar echoes happening in different regions at the same time. (a) Input sequence. (b) Ground truth. (c) ConvLSTM. (d) TrajGRU. (e) PredRNN. (f) MIM. (g) Conv-TT-LSTM. (h) FDNet. (i) Flow branch restruction. (j) Deformation branch restruction. The pixels tend to move up and right as a whole, but zoom into the detail and the echoes at the left bottom (in the blue boxes of (b)) dissipate, while the rainfall range in the middle is expanding. All baseline models above the line, i.e., (c)–(g), fail to perceive the dissipation at the left bottom and lose the heavy rainfall information in the middle area (in the grey boxes in (b)). Only FDNet captures all these fine-grained evolutions and predicts relatively correct frames, especially for longer future time steps.

ing offlow and deformation reduces the uncertainty of the forecast and slows down the tendency of image blurring. Our model performed favorably compared with the state-of-the-art methods on precipitation

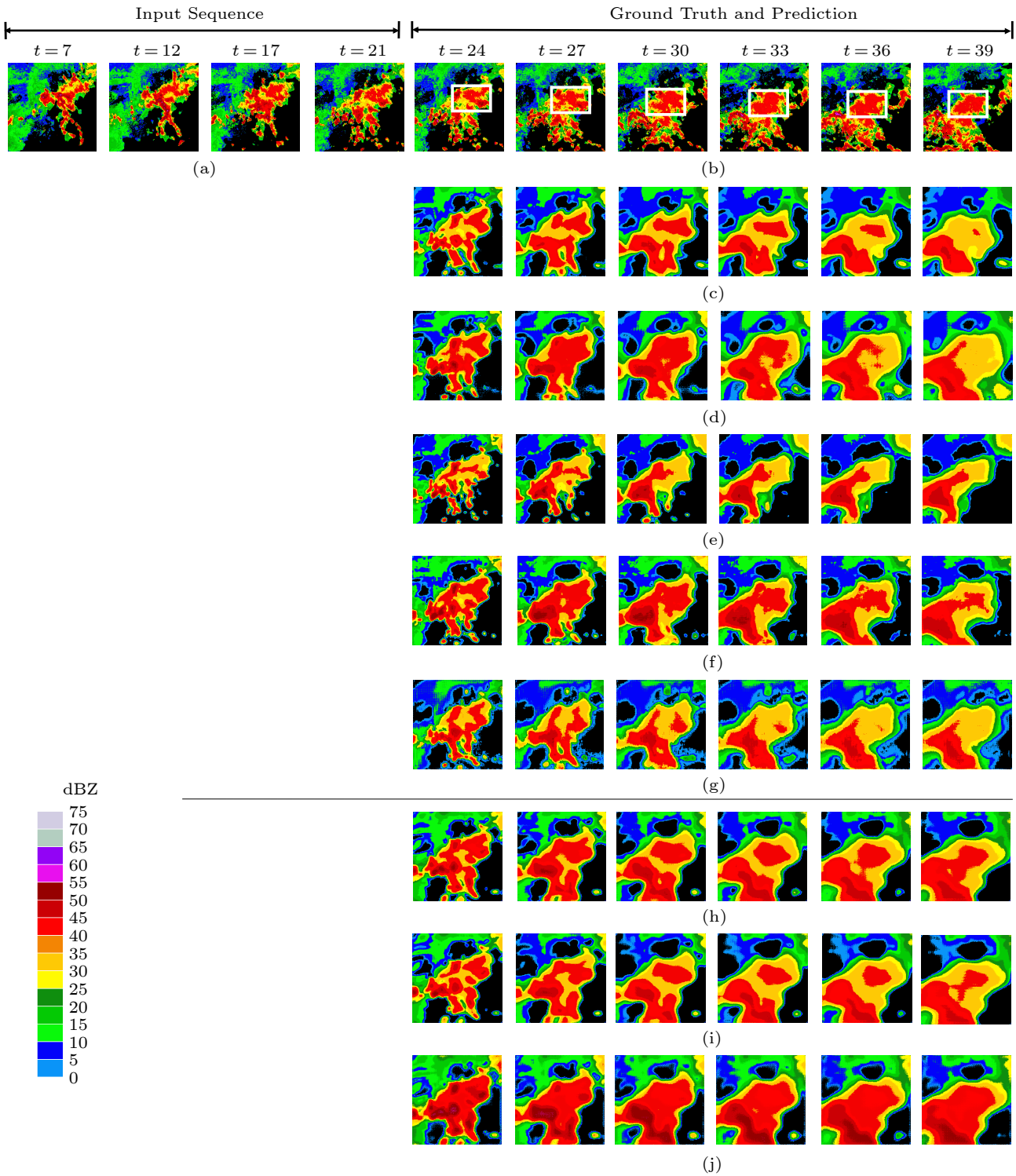


Fig.9. Showcase 3: the intensities of radar echoes remain the same on the whole, with a slight increase. (a) Input sequence. (b) Ground truth. (c) ConvLSTM. (d) TrajGRU. (e) PredRNN. (f) MIM. (g) Conv-TT-LSTM. (h) FDNet. (i) Flow branch restruction. (j) Deformation branch restruction. All baseline methods above the line, i.e., (c)–(g), predict that the rainfall in the middle area (in the white boxes in (b)) would get abating. Only FDNet successfully predicts the long-lasting heavy rain.

nowcasting, especially for relatively longer future predictions. For future work, we plan to train an intelligent model to distinguish different evolution patterns among radar echo sequences, and treat them with the most suitable model individually.

Conflict of Interest The authors declare that they have no conflict of interest.

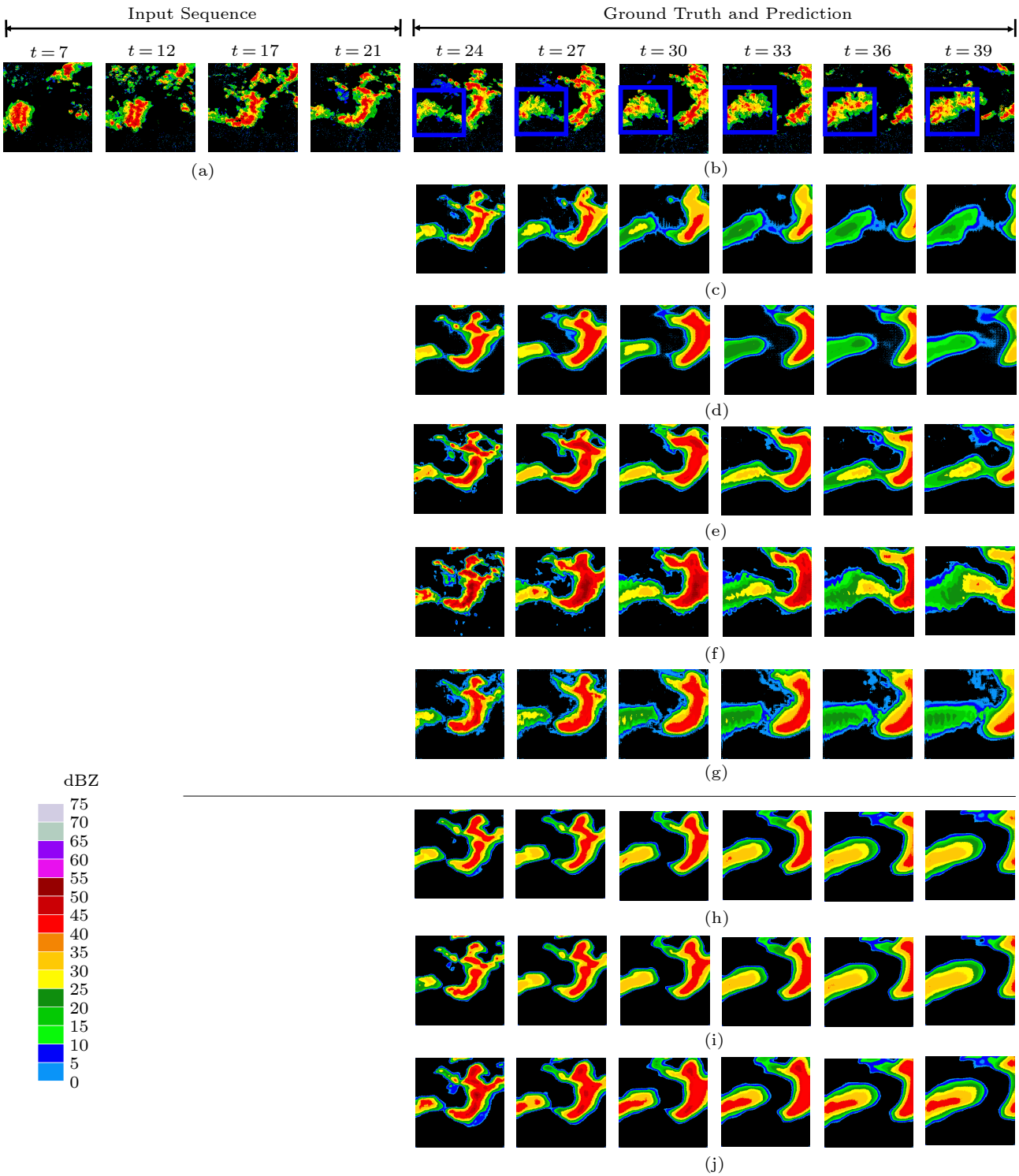


Fig.10. Showcase 4: an example of new coming rainfall. (a) Input sequence. (b) Ground truth. (c) ConvLSTM. (d) TrajGRU. (e) PredRNN. (f) MIM. (g) Conv-TT-LSTM. (h) FDNet. (i) Flow branch restruction. (j) Deformation branch restruction. The deformation modeling is particularly sensitive to newly generated rainfall. For example, the restruction result from the deformation branch successfully predicts the new coming heavy rainfall at the left bottom (in the blue boxes in (b)), which makes FDNet's predictions much closer to the ground truth numerically than the other models.

References

[1] Sun J Z, Xue M, Wilson J W, Zawadzki I, Ballard S P, Onvlee-Hooimeyer J, Joe P, Barker D M, Li P W, Golding B, Xu M, Pinto J. Use of NWP for nowcasting convective precipitation: Recent progress and challenges. *Bulletin of the American Meteorological Society*, 2014, 95(3): 409–426. DOI: 10.1175/BAMS-D-11-00263.1.

[2] Bauer P, Thorpe A, Brunet G. The quiet revolution of numerical weather prediction. *Nature*, 2015, 525(7567): 47–55. DOI: 10.1038/nature14956.

[3] Cheung P, Yeung H Y. Application of optical-flow technique to significant convection nowcast for terminal areas in Hong Kong. In *Proc. the 3rd WMO International Symposium on Nowcasting and Very Short-Range Forecasting* (*WSN12*), Aug. 2012, pp.6–10.

[4] Bowler N E H, Pierce C E, Seed A. Development of a precipitation nowcasting algorithm based upon optical flow techniques. *Journal of Hydrology*, 2004, 288(1/2): 74–91. DOI: 10.1016/j.jhydrol.2003.11.011.

[5] Germann U, Zawadzki I. Scale-dependence of the predictability of precipitation from continental radar images. Part I: Description of the methodology. *Monthly Weather Review*, 2002, 130(12): 2859–2873. DOI: 10.1175/1520-0493(2002)130<2859:SDOTPO>2.0.CO;2.

[6] Sakaino H. Spatio-temporal image pattern prediction method based on a physical model with time-varying optical flow. *IEEE Trans. Geoscience and Remote Sensing*, 2013, 51(5): 3023–3036. DOI: 10.1109/TGRS.2012.2212201.

[7] Shi X J, Chen Z R, Wang H, Yeung D Y, Wong W K, Woo W C. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Proc. the 28th International Conference on Neural Information Processing Systems*, Dec. 2015, pp.802–810. DOI: 10.1007/978-3-319-21233-3_6.

[8] Shi X J, Gao Z H, Lausen L, Wang H, Yeung D Y, Wong W K, Woo W C. Deep learning for precipitation nowcasting: A benchmark and a new model. In *Proc. the 31st International Conference on Neural Information Processing Systems*, Dec. 2017, pp.5617–5627.

[9] Wang Y B, Long M S, Wang J M, Gao Z F, Philip S Y. PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs. In *Proc. the 31st International Conference on Neural Information Processing Systems*, Dec. 2017, pp.879–888.

[10] Villegas R, Yang J M, Hong S, Lin X Y, Lee H. Decomposing motion and content for natural video sequence prediction. In *Proc. the 5th International Conference on Learning Representations*, Apr. 2017.

[11] Wang Y B, Zhang J J, Zhu H Y, Long M S, Wang J M, Yu P S. Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics. In *Proc. the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2019, pp.9146–9154. DOI: 10.1109/CVPR.2019.00937.

[12] Ilg E, Mayer N, Saikia T, Keuper M, Dosovitskiy A, Brox T. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017, pp.1647–1655. DOI: 10.1109/CVPR.2017.179.

[13] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. In *Proc. the 26th International Conference on Neural Information Processing Systems*, Dec. 2012, pp.1097–1105. DOI: 10.1145/3065386.

[14] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks. In *Proc. the 28th International Conference on Neural Information Processing Systems*, Dec. 2014, pp.3104–3112.

[15] Mathieu M, Couprie C, LeCun Y. Deep multi-scale video prediction beyond mean square error. In *Proc. the 4th International Conference on Learning Representations*, May 2016.

[16] Yu B, Yin H T, Zhu Z X. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proc. the 27th International Joint Conference on Artificial Intelligence*, Jul. 2018, pp.3634–3640. DOI: 10.24963/ijcai.2018/505.

[17] Vondrick C, Pirsiavash H, Torralba A. Generating videos with scene dynamics. In *Proc. the 30th International Conference on Neural Information Processing Systems*, Dec. 2016, pp.613–621.

[18] Li Y G, Yu R, Shahabi C, Liu Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *Proc. the 6th International Conference on Learning Representations*, Apr. 30–May 3, 2018.

[19] Ranzato M, Szlam A, Bruna J, Mathieu M, Collobert R, Chopra S. Video (language) modeling: A baseline for generative models of natural videos. arXiv: 1412.6604, 2014. https://arxiv.org/abs/1412.6604, Aug. 2023.

[20] Srivastava N, Mansimov E, Salakhutdinov R. Unsupervised learning of video representations using LSTMs. In *Proc. the 32nd International Conference on International Conference on Machine Learning*, Jul. 2015, pp.843–852.

[21] De Brabandere B, Jia X, Tuytelaars T, Van Gool L. Dynamic filter networks. In *Proc. the 30th International Conference on Neural Information Processing Systems*, Dec. 2016, pp.667–675.

[22] Wang Y B, Jiang L, Yang M H, Li L J, Long M S, Li F F. Eidetic 3D LSTM: A model for video prediction and beyond. In *Proc. the 2019 International Conference on Machine Learning*, May 2019.

[23] Su J H, Byeon W, Kossaifi J, Huang F R, Kautz J, Anandkumar A. Convolutional tensor-train LSTM for spatio-temporal learning. arXiv: 2002.09131, 2020. https://arxiv.org/abs/2002.09131v2, Aug. 2023.

[24] Lin Z H, Li M M, Zheng Z B, Cheng Y Y, Yuan C. Self-attention ConvLSTM for spatiotemporal prediction. In *Proc. the 34th AAAI Conference on Artificial Intelligence*, Feb. 2020, pp.11531–11538. DOI: 10.1609/aaai.v34i07.6819.

[25] Tran Q K, Song S K. Multi-channel weather radar echo extrapolation with convolutional recurrent neural networks. *Remote Sensing*, 2019, 11(19): 2303. DOI: 10.3390/rs11192303.

[26] Dosovitskiy A, Fischer P, Ilg E, Häusser P, Hazirbas C, Golkov V, van der Smagt P, Cremers D, Brox T. FlowNet: Learning optical flow with convolutional networks. In *Proc. the 2015 IEEE International Conference on Computer Vision*, Dec. 2015, pp.2758–2766. DOI: 10.1109/ICCV.2015.316.

[27] Jaderberg M, Simonyan K, Zisserman A, Kavukcuoglu K. Spatial transformer networks. In *Proc. the 29th International Conference on Neural Information Processing Sys-*

tems, Dec. 2015. pp.2017–2025.

[28] Zeiler M D, Taylor G W, Fergus R. Adaptive deconvolutional networks for mid and high level feature learning. In *Proc. the 2011 International Conference on Computer Vision*, Nov. 2011, pp.2018–2025. DOI: 10.1109/ICCV.2011.6126474.

[29] Kingma D P, Ba J. Adam: A method for stochastic optimization. In *Proc. the 3rd International Conference on Learning Representations*, May 2015.

[30] Maas A L, Hannun A Y, Ng A Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. the 30th International Conference on Machine Learning*, June 2013.

[31] Bengio S, Vinyals O, Jaitly N, Shazeer N. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proc. the 29th International Conference on Neural Information Processing Systems*, Dec. 2015, pp.1171–1179.

[32] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In *Proc. the 13th International Conference on Artificial Intelligence and Statistics*, May 2010, pp.249–256.

Bi-Ying Yan is a Ph.D. candidate in University of Chinese Academy of Sciences, Beijing. She received her M.S. degree in computer science and technology from Beihang University, Beijing, in 2013. She is currently a research assistant in the Laboratory of Parallel Software and Computational Science, Institute of Software, Chinese Academy of Sciences, Beijing. Her research interests include machine learning, data mining, big data and parallel computing.

Chao Yang received his B.S. degree in mathematics from the University of Science and Technology of China, Hefei, in 2002, and his Ph.D. degree in computer software and theory from the Institute of Software, Chinese Academy Sciences, Beijing, in 2007. He is currently a full professor with Peking University, Beijing, and a vice director of the Center of Computational Science and Engineering, Peking University, Beijing. His research interest is large-scale parallel computing. He is a recipient of the 2016 ACM Gordon Bell Prize and the 2017 CCF-IEEE CS Young Computer Scientist Award. He is a senior member of CCF and a member of ACM and IEEE.

Feng Chen received his Ph.D. degree in computer science and technology from Beihang University, Beijing, in 2010. He is currently a professor in the Laboratory of Parallel Software and Computational Science, Institute of Software, Chinese Academy of Sciences, Beijing. His research interests include data mining, cloud computing, and large-scale device collaboration.

Kohei Takeda is a senior manager at NTT DATA Corporation, Tokyo. He is leading a research and development team focusing on AI applications and cyber-physical systems. He received his M.Eng. degree in aerospace engineering from the University of Tokyo, Tokyo, in 2002. His research interests include software engineering, machine learning, and information retrieval.

Changjun Wang is a manager at NTT DATA Institute of Management Consulting Inc., Tokyo. He is leading a research and development team focusing on the development of IT application systems in the environmental field. He received his Ph.D. degree in environmental engineering from the Ehime University, Matsuyama, in 1999. His research interests include software engineering and application of geographic information system in the environmental field.