

# PredRNN: A Recurrent Neural Network for Spatiotemporal Predictive Learning

Yunbo Wang, Haixu Wu, Jianjin Zhang, Zhifeng Gao, Jianmin Wang,  
Philip S. Yu, *Fellow, IEEE*, Mingsheng Long, *Member, IEEE*

**Abstract**—The predictive learning of spatiotemporal sequences aims to generate future images by learning from the historical context, where the visual dynamics are believed to have modular structures that can be learned with compositional subsystems. This paper models these structures by presenting PredRNN, a new recurrent network, in which a pair of memory cells are explicitly decoupled, operate in nearly independent transition manners, and finally form unified representations of the complex environment. Concretely, besides the original memory cell of LSTM, this network is featured by a zigzag memory flow that propagates in both bottom-up and top-down directions across all layers, enabling the learned visual dynamics at different levels of RNNs to communicate. It also leverages a memory decoupling loss to keep the memory cells from learning redundant features. We further propose a new curriculum learning strategy to force PredRNN to learn long-term dynamics from context frames, which can be generalized to most sequence-to-sequence models. We provide detailed ablation studies to verify the effectiveness of each component. Our approach is shown to obtain highly competitive results on five datasets for both action-free and action-conditioned predictive learning scenarios.

**Index Terms**—Predictive learning, spatiotemporal modeling, recurrent neural networks

arXiv:2103.09504v4 [cs.LG] 9 Apr 2022

## 1 INTRODUCTION

As a key application of predictive learning, generating future frames from historical consecutive frames has received growing interest in machine learning and computer vision communities. It benefits many practical applications and downstream tasks, such as the precipitation forecasting [1, 2], traffic flow prediction [3, 4], physical scene understanding [5, 6, 7, 8], early activity recognition [9], deep reinforcement learning [10, 11], and the vision-based model predictive control [12, 13]. Many of these existing approaches suggested leveraging RNNs [14, 15] with stacked LSTM units [16] to capture the temporal dependencies of spatiotemporal data. This architecture is mainly inspired by similar ideas from other well-explored tasks of sequential data, such as neural machine translation [17, 18], speech recognition [19], video action recognition [20, 21], and video captioning [21].

For stacked LSTMs, a network structure named *memory cell* plays an important role in alleviating the vanishing gradient problem of RNNs. Strong theoretical and empirical evidence has shown that it can latch the gradients of hidden states inside each LSTM unit in the training process and thereby preserve valuable information of underlying temporal dynamics [16]. However, the state transition pathway of LSTM memory cells may not be optimal for spatiotemporal predictive learning, as this task requires different focuses on the learned representations in many aspects from other tasks of sequential data. First, most predictive networks for language or speech modeling [17, 18, 19] focus on capturing the long-term, non-Markovian properties of sequential data, rather than

spatial deformations of visual appearance. But for future frames prediction, both data structures in space-time are crucial and need to be carefully considered. Second, in other supervised tasks of video data, such as action recognition, high-level semantical features can be informative enough, and the low-level features are less important to final outputs. Due to the absence of complex structures of supervision signals, the stacked LSTMs don't need to preserve fine-grained representations from the bottom up. Although the existing recurrent architecture based on inner-layer memory transitions can be sufficient to capture temporal variations at each level of the network, it may not be the best choice for predictive learning, where low-level details and high-level semantics of spatiotemporal data are both significant to generating future frames.

To jointly model the spatial correlations and temporal dynamics at different levels of RNNs, we propose a new memory-prediction framework named Predictive Recurrent Neural Network (PredRNN), which extends the inner-layer transition function of memory states in LSTMs to *spatiotemporal memory flow*. The spatiotemporal memory flow traverses all nodes of PredRNN in a zigzag path of bi-directional hierarchies: At each timestep, the low-level information is delivered vertically from the input to the output via a newly-designed memory cell, while at the top layer, the spatiotemporal memory flow brings the high-level memory state to the bottom layer at the next timestep. By this means, the top-down expectations interact with the bottom-up signals to both analyze those inputs and generate predictions of subsequently expected inputs, which is different from stacked LSTMs, where the memory state is latched inside each recurrent unit.

Accordingly, we define the central building block of PredRNN as the Spatiotemporal LSTM (ST-LSTM), in which the proposed spatiotemporal memory flow interacts with the original, unidirectional memory state of LSTMs. The intuition is that if we expect a vivid imagination of multiple future images, we need a unified memory mechanism to cope with both short-term deformations of spatial details and long-term dynamics: On one hand, the new

- Y. Wang is with MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China.
- H. Wu, Z. Gao, J. Wang, P. S. Yu, and M. Long are with the School of Software, BNRIst, Tsinghua University, China.
- J. Zhang is with the Microsoft Corporation, China.
- Y. Wang and H. Wu contributed equally to this work.
- Corresponding author: Mingsheng Long, mingsheng@tsinghua.edu.cn.

TABLE 1: The evolution of convolutional recurrent units in video prediction models. The “*Standard*” training scheme is to follow the conventional sequence-to-sequence framework [23] that feeds the historical real observations to the sequence encoder and feeds the previously generated frames to the decoder during the entire training phase. “SS” is short for *scheduled sampling* [24].

Model	Architecture	Recurrent unit	Training (Encoder + Decoder)
ConvLSTM [1]	Seq-to-Seq with stacked layers [23]	Convolutional LSTM	Standard [23] + Standard [23]
TrajGRU [25]	Seq-to-Seq with reversed forecasting layers	Trajectory GRU	Standard + Standard
CDNA [26]	Action-conditioned ConvLSTM network	ConvLSTM [1]	Standard + SS [24]
PredRNN [2] ( <i>Conf. version</i> )	Spatiotemporal memory flow	Spatiotemporal LSTM	Standard + SS
PredRNN++ [27]	Gradient highway	Causal LSTM	Standard + SS
E3D-LSTM [9]	Temporal self-attention on memory cells	Eidetic 3D LSTM	Standard + SS
CrevNet [28]	Two-way autoencoder + Reversible prediction	ConvLSTM [1]; ST-LSTM [2]	Standard + Standard
Conv-TT-LSTM [29]	Seq-to-Seq with stacked layers	Conv. Tensor-Train LSTM	Standard + SS
PredRNN-V2 ( <i>Ours</i> )	Spatiotemporal memory flow	ST-LSTM (decoupled memory)	Reverse scheduled sampling + SS

design of the spatiotemporal memory cell enables the network to learn complex transition functions within short neighborhoods of consecutive frames by increasing the depth of non-linear neurons between time-adjacent RNN states. It thus significantly improves the modeling capability of ST-LSTM for short-term dynamics. On the other hand, ST-LSTM still retains the temporal memory cell of LSTMs and closely combines it with the proposed spatiotemporal memory cell, in pursuit of both long-term coherence of hidden states and their quick response to short-term dynamics.

This journal paper extends our previous work [2] in three technical aspects: First, we propose a decoupling loss to maximize the distance of memory states between the two memory cells in ST-LSTM. It is largely inspired by the theoretical argument that distributed representation has a great potential to match the underlying properties of data distribution [22]. In other words, we implicitly train the two memory cells to focus on different aspects of spatiotemporal variations. Second, we extend PredRNN to support action-conditioned video prediction, such that it models the effects of various actions taken by an agent in high-dimensional pixel observations. Third, another challenge we observed for video prediction is that the widely used sequence-to-sequence training procedure [17] prevents the models from learning long-term dynamics in the context frames. To this end, we introduce a new curriculum learning procedure named Reverse Scheduled Sampling, which forces our model to learn more about *jumpy* frame dependencies by randomly hiding real observations in the input sequence with certain probabilities changing over the training phase. We believe that the proposed training procedure can be easily extended to other video prediction models that also follow the sequence-to-sequence framework [17].

Our approach achieves state-of-the-art performance on five datasets: the Moving MNIST dataset, the KTH action dataset, a radar echo dataset for precipitation forecasting, the Traffic4Cast dataset of high-resolution traffic flows, and the action-conditioned BAIR dataset with robot-object interactions. The last two of them are new in this version of the manuscript. We perform ablation studies to understand the effectiveness of each component of PredRNN, which is complementary to existing methods and further improves them when integrated. We release the code to facilitate future research at <https://github.com/thuml/predrnn-pytorch>.

## 2 RELATED WORK

For spatiotemporal predictive learning, different inductive biases are encoded into neural networks by using different network architectures [30], which, in general, can be roughly divided into three groups: feed-forward models based on CNNs [31, 32, 33], recurrent models [1, 34, 35], and other models including the

combinations of the convolutional and recurrent networks, as well as the Transformer-based and flow-based methods [36, 37].

**Feed-forward models based on CNNs.** The use of convolutional layers has introduced the inductive bias of group invariance over space to spatiotemporal prediction. Oh *et al.* [31] defined a convolutional autoencoder for next-frame prediction in Atari games. Xue *et al.* [38] presented a probabilistic model that encodes motion information as convolutional kernels and learns to produce a probable set of future frames by learning their conditional distribution. Zhang *et al.* [39] used a CNN with residual connections for traffic prediction, which considers the closeness, period, trend, and external factors of the traffic flows. Recent literature proposed to train the predictive CNNs with adversarial learning [40, 41] to reduce the uncertainty of the learning process and improve the sharpness of the generated frames [32, 33, 42, 43, 44, 45, 46]. Compared with the recurrent models, feed-forward models typically show higher computational efficiency on large-scale GPUs. However, these models learn complicated state transitions as compositions of simpler ones by stacking convolutional layers, thus often failing to capture long-term dependency between distant frames.

**Recurrent models.** Recent advances in RNNs typically show a greater ability to model the dynamics of historical observations. Ranzato *et al.* [47] defined an RNN inspired by language modeling, predicting future frames in a discrete space of patch clusters. Srivastava *et al.* [34] proposed a sequence-to-sequence video prediction model borrowed from neural machine translation [17]. Later on, some work improved the modeling of temporal relations and extended the prediction horizon by organizing 2D recurrent states in hierarchical architectures [48, 49, 50]. To model the temporal uncertainty, some work proposed to integrate variational inference with 2D recurrence [35, 51, 52, 53, 54, 55, 56]. Another line of work is to factorize the content and motion of videos. Typical approaches include using optical flows [57, 58], adversarial training schemes [59], reasoning about object-centric content and pose vectors [60, 61], differentiable clustering methods [6], amortized inference [62, 63], and solvers of neural differential equations [64]. However, the above methods mainly used 2D RNNs to model video dynamics in low-dimensional space, which inevitably causes the loss of visual information in real scenarios. In this paper, we are more focused on improving the spatiotemporal modeling capability of the recurrent predictive backbones.

**Convolutional RNNs.** Table 1 summarizes the evolution of recent video prediction models that combine the advantages of convolutional and recurrent architectures. Shi *et al.* [1] proposed the Convolutional LSTM (ConvLSTM), which uses convolutions

to replace the matrix multiplication in the recurrent transitions of the original LSTM. Finn *et al.* [26] designed an action-conditioned ConvLSTM network for visual planning and control. Shi *et al.* [25] combined convolutions with GRUs [18] and extended the receptive fields of state-to-state transitions with non-local neural connections. Wang *et al.* [9] enriched each RNN state by incorporating a time dimension that covers a short snippet, and proposed to model the dynamics using 3D convolutions and temporal self-attention. Su *et al.* [29] improved the efficiency of higher-order ConvLSTMs based on low-rank tensor factorization. In summary, the convolutional RNNs jointly model the visual appearances and temporal dynamics, and is also a foundation for the follow-up approaches [4, 9, 25, 26, 27, 28, 29, 65, 66, 67, 68, 69, 70, 71]. This paper improves upon the existing video prediction models with (i) a zigzag recurrent transition mechanism named *spatiotemporal memory flow*, (ii) a new convolutional recurrent unit with a pair of decoupled memory cells, and (iii) a new training procedure for sequence-to-sequence predictive learning.

### 3 PRELIMINARIES

#### 3.1 Spatiotemporal Predictive Learning

Suppose we are monitoring a dynamical system of  $J$  measurements over time, where each measurement is recorded at all locations in a spatial region represented by an  $M \times N$  grid. From the spatial view, the observation of these  $J$  measurements at any time can be represented by a tensor of  $\mathcal{X} \in \mathbb{R}^{J \times M \times N}$ . From the temporal view, the observations over  $T$  timesteps form a sequence of  $\mathcal{X}_{\text{in}} = \{\mathcal{X}_1, \dots, \mathcal{X}_T\}$ . Given  $\mathcal{X}_{\text{in}}$ , the spatiotemporal predictive learning is to predict the most probable length- $K$  sequence in the future,  $\hat{\mathcal{X}}_{\text{out}} = \{\hat{\mathcal{X}}_{T+1}, \dots, \hat{\mathcal{X}}_{T+K}\}$ . In this paper, we train neural networks parametrized by  $\theta$ . Concretely, we use stochastic gradient descent to find a set of parameters  $\theta^*$  that maximizes the log-likelihood of producing the true target sequence  $\mathcal{X}_{\text{out}}$  given the input data  $\mathcal{X}_{\text{in}}$  for all training pairs  $\{(\mathcal{X}_{\text{in}}^n, \mathcal{X}_{\text{out}}^n)\}_n$ :

$$\theta^* = \arg \max_{\theta} \sum_{(\mathcal{X}_{\text{in}}^n, \mathcal{X}_{\text{out}}^n)} \log P(\mathcal{X}_{\text{out}}^n | \mathcal{X}_{\text{in}}^n; \theta). \quad (1)$$

In this paper, we take video prediction as a typical experimental domain, where the observed data at each timestep is an RGB image and the number of measured channels is 3. Another domain is precipitation nowcasting, where the observed data is a sequence of radar echo maps in a certain geographic region<sup>1</sup>.

#### 3.2 Convolutional LSTM

The Convolutional LSTM (ConvLSTM) [1] models the spatial and temporal data structures simultaneously by explicitly encoding the spatial information into tensors, and applying convolution operators to both the state-to-state and input-to-state recurrent transitions. It overcomes the limitation of vector-variate representations in standard LSTM, where the spatial correlation is not explicitly modeled. The input state  $\mathcal{X}_t$ , memory state  $\mathcal{C}_t$ , and hidden state  $\mathcal{H}_t$  at each timestep are 3D tensors in  $\mathbb{R}^{J \times M \times N}$ . The first dimension is either the number of measurements (for input states) or the number of feature maps (otherwise), and the last two dimensions are the numbers of spatial rows  $M$  and columns  $N$ . Like standard LSTM, the input gate  $i_t$ , forget gate  $f_t$ , output gate  $o_t$ , and input-modulation gate  $g_t$  control the information flow across  $\mathcal{C}_t$ , such

that the gradient will be kept from quickly vanishing by being trapped in the memory cell. ConvLSTM determines the future state of a certain cell in the  $M \times N$  grid based on the input frame and past states of its local neighbors:

$$\begin{aligned} g_t &= \tanh(W_{xg} * \mathcal{X}_t + W_{hg} * \mathcal{H}_{t-1} + b_g) \\ i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \odot \mathcal{C}_{t-1} + b_i) \\ f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \odot \mathcal{C}_{t-1} + b_f) \\ \mathcal{C}_t &= f_t \odot \mathcal{C}_{t-1} + i_t \odot g_t \\ o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \odot \mathcal{C}_t + b_o) \\ \mathcal{H}_t &= o_t \odot \tanh(\mathcal{C}_t), \end{aligned} \quad (2)$$

where  $\sigma$  is the Sigmoid activation function,  $*$  and  $\odot$  denote the convolution operator and the Hadamard product. We observe that the ConvLSTM network can be improved from three aspects.

**Challenge I.** For a stacked ConvLSTM network, the input sequence  $\mathcal{X}_{\text{in}}$  is fed into the bottom layer, and the output sequence  $\mathcal{X}_{\text{out}}$  is generated at the top one. With hidden states  $\mathcal{H}_t$  being delivered from the bottom up, spatial representations are encoded layer by layer. However, the memory states  $\mathcal{C}_t$  are merely updated along the arrow of time within each ConvLSTM layer, being less dependent on the hierarchical visual features at other layers. Thus, the first layer at the current timestep may largely ignore what had been memorized by the top layer at the previous timestep.

**Challenge II.** In ConvLSTM, the output hidden state  $\mathcal{H}_t$  is dependent on the memory state  $\mathcal{C}_t$  and the output gate  $o_t$ , which means that the memory cell is forced to cope with long-term and short-term dynamics simultaneously. Therefore, the modeling capability of  $\mathcal{C}_t$  may greatly limit the overall performance of the model for complex spatiotemporal variations.

**Challenge III.** The ConvLSTM network follows the training procedure of sequence-to-sequence RNNs [23]. During training, it always takes as inputs real observations at encoding timesteps but has to rely more on the long-term information from the previous context frames at the forecasting timesteps. Since both the encoding and forecasting parts of ConvLSTM use the same set of model parameters, the one-step training in the encoding phase may prevent the forecaster from learning the jumpy frame dependencies, thus affecting the performance of long-term prediction.

### 4 METHOD

To tackle the above challenging problems, we propose the Predictive Recurrent Neural Network (PredRNN). For **Challenge I**, we present a new RNN architecture named the spatiotemporal memory flow, which improves the state transition functions of convolutional RNNs. For **Challenge II**, we propose the Spatiotemporal LSTM (ST-LSTM) which serves as the building block of PredRNN. It leverages a pair of memory cells that are jointly learned and explicitly decoupled to cover long- and short-term dynamics of spatiotemporal variations. For **Challenge III**, we improve the training procedure to encourage the predictive network to learn non-Markovian dynamics from longer periods of observation frames. Furthermore, we present an **action-conditioned** PredRNN, which allows simulating the spatiotemporal variations of the environment in response to the actions of the agent in decision-making scenarios.

#### 4.1 Spatiotemporal Memory Flow

PredRNN employs a stack of convolutional recurrent units to learn unified representations of the spatial correlations and temporal

1. We usually visualize radar echoes by mapping them to color images.

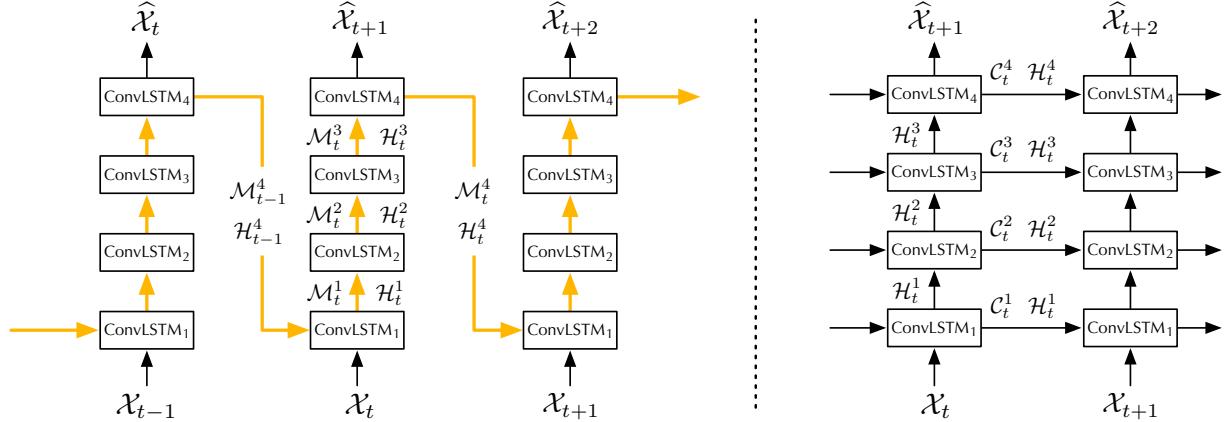


Fig. 1: **Left:** the spatiotemporal memory flow architecture that uses ConvLSTM as the building block. The orange arrows show the deep-in-time path of memory state transitions. **Right:** the original ConvLSTM network proposed by Shi *et al.* [1].

dynamics from the input sequence, and then transforms these features back to the data space to predict future spatiotemporal frames. We initially adopt the original ConvLSTM layer as the basic building block of PredRNN and apply a novel memory state transition method between the recurrent nodes. In the original ConvLSTM network [1] shown in Fig. 1 (right), the memory states  $\mathcal{C}_t^l$  are constrained inside individual recurrent layers and updated along the arrow of time. Only the hidden states  $\mathcal{H}_t^l$  can be delivered from the observation to the final prediction. This temporal memory flow has been widely used in supervised learning tasks such as video classification, where the hidden representations are more and more abstract and class-specific, starting at the bottom. However, in predictive learning, the input frames and the expected outputs share the same data space, *i.e.*, they may have very close data distributions in the spatial domain and very related underlying dynamics in the temporal domain. Therefore, it becomes important to make the predictions more effectively dependent on the memory representations learned at different levels of recurrent layers. If we want to frame the future vividly, we need both high-level understandings of global motions and more detailed memories of the subtle variations in the input sequence.

Considering that the memory cell of ConvLSTM can latch the gradients<sup>2</sup> and thereby store valuable information across recurrent nodes, we improve the above horizontal memory flow by updating the memory state in the zigzag direction, so that it can better deliver knowledge from input to output. We show the key equations of this memory flow using ConvLSTM as the building block:

$$\begin{aligned}
 g_t &= \tanh(W_{xg} * \mathcal{X}_t \mathbb{1}_{\{l=1\}} + W_{hg} * \mathcal{H}_t^{l-1}) \\
 i_t &= \sigma(W_{xi} * \mathcal{X}_t \mathbb{1}_{\{l=1\}} + W_{hi} * \mathcal{H}_t^{l-1} + W_{ci} * \mathcal{M}_t^{l-1}) \\
 f_t &= \sigma(W_{xf} * \mathcal{X}_t \mathbb{1}_{\{l=1\}} + W_{hf} * \mathcal{H}_t^{l-1} + W_{cf} * \mathcal{M}_t^{l-1}) \\
 \mathcal{M}_t^l &= f_t \odot \mathcal{M}_t^{l-1} + i_t \odot g_t \\
 o_t &= \sigma(W_{xo} * \mathcal{X}_t \mathbb{1}_{\{l=1\}} + W_{ho} * \mathcal{H}_t^{l-1} + W_{co} * \mathcal{M}_t^l) \\
 \mathcal{H}_t^l &= o_t \odot \tanh(\mathcal{M}_t^l).
 \end{aligned} \tag{3}$$

We name this new memory state transition method the *spatiotemporal memory flow*, and show its transition direction by the orange arrows in Fig. 1 (left). In this way, the memory states in different ConvLSTM layers are no longer independent, and all nodes in the entire recurrent network jointly maintain a memory

2. Like the standard LSTM, the memory cell of ConvLSTM was originally designed to alleviate the gradient vanishing problem during training.

bank denoted by  $\mathcal{M}$ . The input gate, input modulation gate, forget gate, and output gate are no longer dependent on the hidden state and the temporal memory state from the previous timestep at the same layer. Instead, they rely on the hidden state  $\mathcal{H}_t^{l-1}$  and the spatiotemporal memory state  $\mathcal{M}_t^{l-1}$  ( $l \in \{1, \dots, L\}$ ) supplied by the previous layer at current timestep (see Fig. 1 (left)). In particular, the bottom recurrent unit ( $l = 1$ ) receives state values from the top layer at the previous timestep:  $\mathcal{H}_t^{l-1} = \mathcal{H}_{t-1}^L$ ,  $\mathcal{M}_t^{l-1} = \mathcal{M}_{t-1}^L$ . The four layers in this figure have different sets of convolutional parameters regarding the input-to-state and state-to-state transitions. They thereby read and update the values of the memory state based on their individual understandings of the spatiotemporal dynamics as the information flows through the current node. Note that we replace the notation for memory state from  $\mathcal{C}$  to  $\mathcal{M}$  to emphasize that it flows in the zigzag direction in PredRNN, instead of the horizontal direction in standard recurrent networks. Different from ConvLSTM which uses Hadamard product  $\odot$  for the computation of the gates, we adopt convolution operators  $*$  for finer-grained memory transitions. An additional benefit of this change is that the learned PredRNN can be deployed directly on the input sequence of different spatial resolutions.

The spatiotemporal memory flow provides a recurrent highway for hierarchical visual representations that can reduce the loss of information from the bottom layers to the top of the network. Besides, by introducing more nonlinear gated neurons within temporal neighborhoods, it expands the deep transition path of hidden states and enhances the modeling capability of the network for short-term dynamics. In contrast, the original ConvLSTM network requires larger convolution kernels for input-to-state and state-to-state transitions in order to capture faster motion, resulting in an unnecessary increase in model parameters.

Alternatively, we can understand the spatiotemporal memory flow from the perspective of *memory networks* [72, 73, 74]. Here, the proposed spatiotemporal memory state  $\mathcal{M}$  can be viewed as a simple version of the so-called *external memory* with continuous memory representations, and the stacked recurrent layers can be viewed as multiple computational steps. The layer-wise forget gates, input gates, and input modulation gates respectively determine the read and write functions, as well as the content to be written. One advantage of the classic *memory networks* is to capture long-term structure (even with multiple temporal hops) within sequences. Our spatiotemporal memory flow is analogous to their mechanism, as it enables our model to consider different levels of video representations before making a prediction.

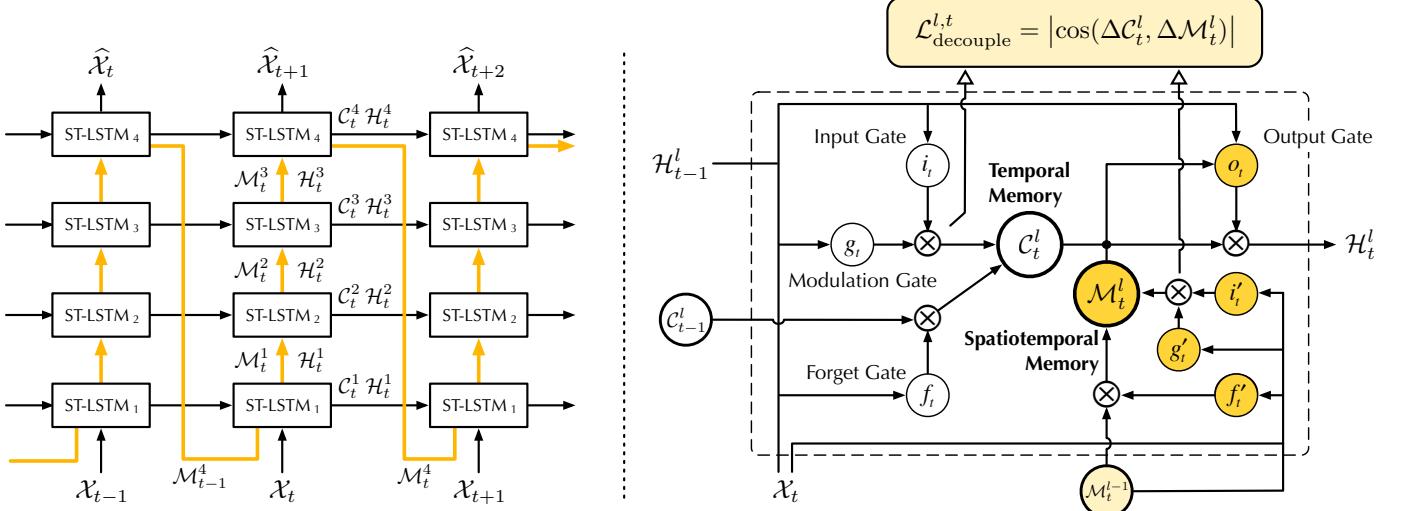


Fig. 2: **Left:** the main architecture of PredRNN, in which the orange arrows denote the state transition paths of  $\mathcal{M}_t^l$ , namely the spatiotemporal memory flow. **Right:** the ST-LSTM unit with twisted memory states that serves as the building block of the proposed PredRNN, where the orange circles denote the unique structures compared with ConvLSTM.

#### 4.2 Spatiotemporal LSTM with Memory Decoupling

As described previously, the spatiotemporal memory state is updated first upwards across layers then forwards to the next timestep. It stretches the state transition path across time and adds extra neurons between horizontally adjacent nodes at the same level of the network. It thus enables the network to learn complex non-linear transition functions of short-term motions. However, this deep-in-time architecture may also bring in the gradient vanishing problem. The roundabout memory transition path may make it difficult to capture long-term dependencies. For both short-term recurrence depth and long-term coherence, we introduce a double-flow memory transition mechanism that combines the original memory cell  $\mathcal{C}$  and the new memory cell  $\mathcal{M}$ , which derives a recurrent unit named Spatiotemporal LSTM (ST-LSTM):

$$\begin{aligned}
g_t &= \tanh(W_{xg} * \mathcal{X}_t + W_{hg} * \mathcal{H}_{t-1}^l) \\
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1}^l) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1}^l) \\
\mathcal{C}_t^l &= f_t \odot \mathcal{C}_{t-1}^l + i_t \odot g_t \\
g'_t &= \tanh(W'_{xg} * \mathcal{X}_t + W_{mg} * \mathcal{M}_t^{l-1}) \\
i'_t &= \sigma(W'_{xi} * \mathcal{X}_t + W_{mi} * \mathcal{M}_t^{l-1}) \\
f'_t &= \sigma(W'_{xf} * \mathcal{X}_t + W_{mf} * \mathcal{M}_t^{l-1}) \\
\mathcal{M}_t^l &= f'_t \odot \mathcal{M}_t^{l-1} + i'_t \odot g'_t \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1}^l + W_{co} * \mathcal{C}_t^l + W_{mo} * \mathcal{M}_t^l) \\
\mathcal{H}_t^l &= o_t \odot \tanh(W_{1 \times 1} * [\mathcal{C}_t^l, \mathcal{M}_t^l]). \tag{4}
\end{aligned}$$

In Fig. 2, we present the final PredRNN model by taking ST-LSTM as the building block in place of ConvLSTM. There are two memory states: First,  $\mathcal{C}_t^l$  is the temporal memory that transits within each ST-LSTM unit from the previous node at  $t-1$  to the current timestep. We adopt the original gates for  $\mathcal{C}_t^l$  from the standard LSTM, but remove the Hadamard terms  $W_{\bullet\bullet} \odot \mathcal{C}_{t-1}^l$  from the computation of  $i_t$  and  $f_t$ , due to empirical performance and model efficiency. Second,  $\mathcal{M}_t^l$  is the spatiotemporal memory, which transits vertically to the current node from the lower  $l-1$  ST-LSTM unit at the same timestep. In particular, we assign  $\mathcal{M}_{t-1}^l$  to

$\mathcal{M}_t^0$  for the bottom ST-LSTM where  $l=1$ . We construct another set of input gate  $i'_t$ , forget gate  $f'_t$ , and input modulation gate  $g'_t$  for  $\mathcal{M}_t^l$ , because the memory transition functions in distinct directions are supposed to be controlled by different signals.

The final hidden states  $\mathcal{H}_t^l$  of each node are dependent on a combination of the horizontal and zigzag memory states: we concatenate the  $\mathcal{C}_t^l$  and  $\mathcal{M}_t^l$ , and then apply the  $1 \times 1$  convolutional layer for dimensionality reduction, which makes the hidden state  $\mathcal{H}_t^l$  have the same dimensions as the memory states. In addition to simple concatenation, pairs of memory states are finally twisted and unified by output gates with bidirectional control signals (horizontal and vertical), resulting in comprehensive modeling of long-term and short-term dynamics. This dual-memory mechanism benefits from the compositional advantage with distributed representations [75, 76]. Intuitively, due to different mechanisms of state transitions, the pair of memory cells in ST-LSTM are expected to deal with different aspects of spatiotemporal variations:

- $\mathcal{M}$  introduces a deeper transition path that zigzags across ST-LSTM units. With the forget gate  $f'_t$  and the input-related modules  $i'_t \odot g'_t$ , it improves the ability to model complex short-term dynamics from one timestep to the next, and allows  $\mathcal{H}$  to transit adaptively at different rates.
- $\mathcal{C}$  operates on a slower timescale. It provides a shorter gradient path between distant hidden states, thus facilitating the learning process of long-term dependencies.

However, as shown in Fig. 3, we visualized the increments of memory states at each timestep (*i.e.*,  $\Delta \mathcal{C}_t^l$  and  $\Delta \mathcal{M}_t^l$ ) using t-SNE [77], and found that they were not automatically separated as expected. In fact, the two memory states are often so intertwined that they are difficult to decouple spontaneously through their respective network architectures. To some extent, it results in the inefficient utilization of network parameters.

Building upon the first version of ST-LSTM [2], we present a new decoupling loss, as shown in Fig. 2 (right), that keeps  $\mathcal{C}$  and  $\mathcal{M}$  from learning redundant features. In each ST-LSTM unit, we first add the convolutional layer upon the increments of  $\mathcal{C}_t^l$  and  $\mathcal{M}_t^l$  at every timestep, and leverage a new decoupling loss to explicitly extend the distance between them in latent space. By this means, different memory states are trained to focus on different aspects of

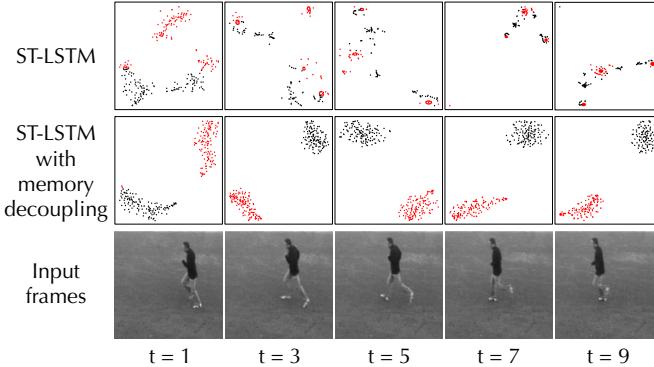


Fig. 3: Visualization of  $\Delta\mathcal{C}_t^l$  (red points) and  $\Delta\mathcal{M}_t^l$  (black points) using t-SNE [77] on the KTH action dataset. Models are respectively trained without or with memory decoupling.

spatiotemporal variations. The overall memory decoupling method can be formulated as follows:

$$\begin{aligned} \Delta\mathcal{C}_t^l &= W_{\text{decouple}} * (i_t \odot g_t) \\ \Delta\mathcal{M}_t^l &= W_{\text{decouple}} * (i'_t \odot g'_t) \\ \mathcal{L}_{\text{decouple}} &= \sum_t \sum_l \sum_c \frac{|\langle \Delta\mathcal{C}_t^l, \Delta\mathcal{M}_t^l \rangle_c|}{\|\Delta\mathcal{C}_t^l\|_c \cdot \|\Delta\mathcal{M}_t^l\|_c}, \end{aligned} \quad (5)$$

where  $W_{\text{decouple}}$  denotes  $1 \times 1$  convolutions shared by all ST-LSTM units;  $\langle \cdot, \cdot \rangle_c$  and  $\|\cdot\|_c$  are respectively dot product and  $\ell_2$  norm of flattened feature maps, which are calculated for each channel  $c$ . At training time, the increments of memory states,  $i_t \odot g_t$  and  $i'_t \odot g'_t$ , are derived from Eq. (4). Notably, the new parameters are only used at training time and are removed from the entire model at inference. That is, there is no increase in model size compared to the previous version of ST-LSTM [2]. By defining the decoupling loss with the cosine similarity, our approach encourages the increments of the two memory states to be orthogonal at any timestep. It unleashes the respective strengths of  $\mathcal{C}$  and  $\mathcal{M}$  for long-term and short-term dynamic modeling. As shown by t-SNE visualization in Fig. 3, at test time,  $\Delta\mathcal{C}_t^l$  and  $\Delta\mathcal{M}_t^l$  can be easily separated.

The decoupling loss is largely inspired by the theoretical evidence that using reasonable inductive bias to construct distributed representations can bring a great performance boost if it matches properties of the underlying data distribution [22]. There is a similar idea in ensemble learning that generally, to form a good ensemble, the base learners should be as more accurate as possible, and as more diverse as possible [78]. The diversity of base learners can be enhanced in different ways, such as sub-sampling the training examples, manipulating the input attributes, and employing randomness into training procedures [79]. It is worth noting that the proposed memory decoupling method has not been used by existing ensemble learning algorithms, though it is inspired by the general idea of enhancing the diversity of base learners. We use it to diversify the pairs of memory states, intuitively in pursuit of a more disentangled representation of long- and short-term dynamics in predictive learning.

The final PredRNN model is trained end-to-end in a fully unsupervised manner. The overall training objective is:

$$\mathcal{L}_{\text{final}} = \sum_{t=2}^{T+K} \|\hat{\mathcal{X}}_t - \mathcal{X}_t\|_2^2 + \mathcal{L}_{\text{decouple}}, \quad (6)$$

where the first term is the frame reconstruction loss that works on the network output at each timestep, the second term is the memory decoupling regularization from Eq. (5).

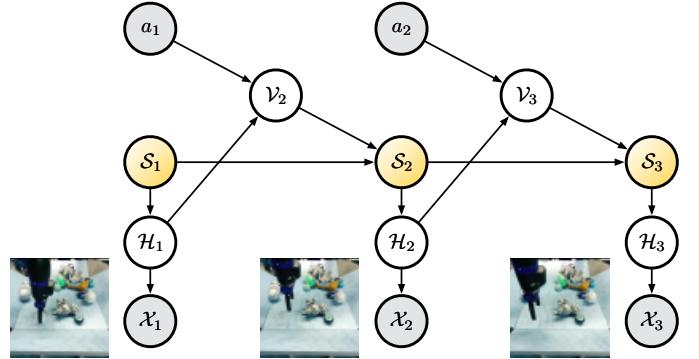


Fig. 4: Graphical model of the action-conditioned PredRNN. For simplicity, we display only one ST-LSTM layer and use  $\mathcal{S}_t$  as the combination of the memory states  $\mathcal{C}_t$  and  $\mathcal{M}_t$  in Eq. (7).

### 4.3 Action-Conditioned ST-LSTM

Action-conditioned video prediction models allow for flexibility in learning the policies of agent-based decision making systems [11, 13, 26, 31, 35, 80, 81]. Inspired by the work of Recurrent Environment Simulator [80], we extend ST-LSTMs, which can be denoted by ST-LSTM( $\mathcal{X}_t, \mathcal{H}_{t-1}^l, \mathcal{C}_{t-1}^l, \mathcal{M}_{t-1}^{l-1}$ ), to action-conditioned video prediction with

$$\begin{aligned} \text{Action fusion: } \mathcal{V}_t^l &= (W_{hv} * \mathcal{H}_{t-1}^l) \odot (W_{av} * \mathcal{A}_{t-1}) \\ \mathcal{H}_t^l, \mathcal{C}_t^l, \mathcal{M}_t^l &= \text{ST-LSTM}(\mathcal{X}_t, \mathcal{V}_t^l, \mathcal{C}_{t-1}^l, \mathcal{M}_{t-1}^{l-1}), \end{aligned} \quad (7)$$

where  $\mathcal{V}_t^l$  encodes previous hidden state  $\mathcal{H}_{t-1}^l$  and the action taken at the previous timestep  $\mathcal{A}_{t-1}$ . In our experiments,  $\mathcal{A}_t$  has continuous values and has been transformed to the same resolution as the hidden state. By incorporating  $\mathcal{V}_t^l$  in each ST-LSTM unit, at both encoding and forecasting timesteps, PredRNN learns to simulate the consequences of future action sequences over long time periods. Different from the original Recurrent Environment Simulator, we validate the effectiveness of action-conditioned ST-LSTM on a dataset collected with a real robot [13].

### 4.4 Training with Reverse Scheduled Sampling

As shown in Fig. 5, we propose a new curriculum learning strategy for PredRNN, which consists of two components:

- **Reverse scheduled sampling (RSS):** It is used at the encoding timesteps and forces the model to learn more about long-term dynamics by randomly hiding real observations with decreasing probabilities as the training phase progresses.
- **Scheduled sampling (SS) [24]:** It is used at the forecasting timesteps to alleviate the inconsistency of data flow between the training and inference phases.

**Difficulty in learning long-term dynamics.** We follow the common practice [26, 51] that uses a unified predictive model  $f_\theta(\cdot)$  for sequence encoding and forecasting. We use  $\mathcal{X}_{\text{in}} = \{\mathcal{X}_1, \dots, \mathcal{X}_T\}$  to denote the context frames and  $\mathcal{X}_{\text{out}} = \{\mathcal{X}_{T+1}, \dots, \mathcal{X}_{T+K}\}$  for ground-truth output frames. We use  $\hat{\mathcal{X}}_{t+1}, t \in \{1, \dots, T+K-1\}$ , to denote the prediction at each timestep. During inference, as well as in the typical training scheme, we have

$$\hat{\mathcal{X}}_{t+1} = \begin{cases} f_\theta(\mathcal{X}_t, \mathcal{H}_{t-1}, \mathcal{S}_{t-1}) & \text{if } t \leq T, \\ f_\theta(\hat{\mathcal{X}}_t, \mathcal{H}_{t-1}, \mathcal{S}_{t-1}) & \text{otherwise,} \end{cases} \quad (8)$$

where  $\mathcal{S}_{t-1} = \{\mathcal{C}_{t-1}, \mathcal{M}_{t-1}\}$  represents the temporal dynamics as a combination of previous memory states. The main difference

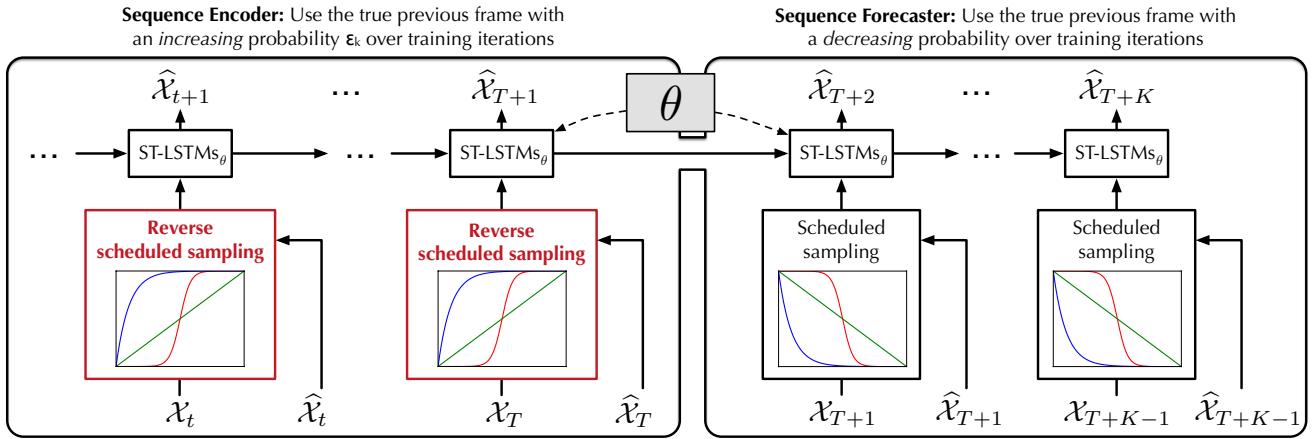


Fig. 5: In the training phase, we apply the *reverse scheduled sampling* at the encoding timesteps to force the model to learn long-term dynamics from context frames. This training scheme can be easily combined with the original scheduled sampling strategy, which is used to close the training-inference gap at forecasting timesteps. Both the encoder and the forecaster are parametrized by  $\theta$ .

between the encoding part ( $t \leq T$ ) and the forecasting part ( $t > T$ ) is whether to use the true frame  $\mathcal{X}_t$  or the previous prediction  $\hat{\mathcal{X}}_t$ . In the encoder, the model is mainly learned to make a one-step prediction, because  $\mathcal{X}_t$  tends to be a more informative signal than  $\mathcal{H}_{t-1}$  and  $\mathcal{S}_{t-1}$ , keeping PredRNN from digging deeper into the non-Markovian properties of historical observations. But for the forecaster, since  $\hat{\mathcal{X}}_t$  does not contain new observations, the model has to learn long-term dynamics from the latter. Such a training inconsistency between the encoder and the forecaster may lead to an ineffective optimization of  $\theta$  and hamper the model from learning long-term dynamics from  $\mathcal{X}_{in}$ .

**Reverse scheduled sampling.** To force the recurrent model to learn long-term dynamics from historical observations, we propose the reverse scheduled sampling, a curriculum learning strategy applied to the input frames at encoding timesteps. At forecasting timesteps, we follow the previous literature [26, 51] to apply the original schedule sampling [24] to the inputs. Finally, we have

$$\hat{\mathcal{X}}_{t+1} = \begin{cases} f_\theta(\hat{\mathcal{X}}_t \xrightarrow{RSS} \mathcal{X}_t, \mathcal{H}_{t-1}, \mathcal{S}_{t-1}) & \text{if } t \leq T, \\ f_\theta(\mathcal{X}_t \xrightarrow{SS} \hat{\mathcal{X}}_t, \mathcal{H}_{t-1}, \mathcal{S}_{t-1}) & \text{otherwise,} \end{cases} \quad (9)$$

where  $\xrightarrow{RSS}$  indicates a gradual change throughout the training phase from taking the previous prediction  $\hat{\mathcal{X}}_t$  as the input to taking the true frame  $\mathcal{X}_t$ . At encoding timesteps, the changing schedule is in the reverse order of the scheduled sampling strategy denoted by  $\xrightarrow{SS}$  at forecasting timesteps. Concretely, for RSS, there is a probability ( $\epsilon_k \in [0, 1]$ ) of sampling the true frame  $\mathcal{X}_t \in \mathcal{X}_{in}$  or a corresponding probability ( $1 - \epsilon_k$ ) of sampling  $\hat{\mathcal{X}}_t$ , such that over the entire encoder, a sequence of sampling outcomes can be seen as a Bernoulli process with  $T$  independent trials.  $\epsilon_k$  is an *increasing* function of the number of training iterations  $k$ , starting from  $\epsilon_s$  and increasing to  $\epsilon_e$ , which has the following forms:

- Linear increase:  $\epsilon_k = \min\{\epsilon_s + \alpha_l \times k, \epsilon_e\}$ ;
- Exponential increase:  $\epsilon_k = \epsilon_e - (\epsilon_e - \epsilon_s) \times \exp(-\frac{k}{\alpha_e})$ ;
- Sigmoid increase:  $\epsilon_k = \epsilon_s + (\epsilon_e - \epsilon_s) \times \frac{1}{1 + \exp(-\frac{\beta_s - k}{\alpha_s})}$ ,

where  $\alpha_l, \alpha_e, \alpha_s > 0$  denote the increasing factors and  $\beta_s > 0$  denotes the starting point of the sigmoid function. These hyperparameters jointly decide the increasing curve of  $\epsilon_k$ . Examples of such schemes are shown in Fig. 6 in the red curves. The encoder is trained with a progressively simplified curriculum. It gradually changes from generating multi-step future frames, which

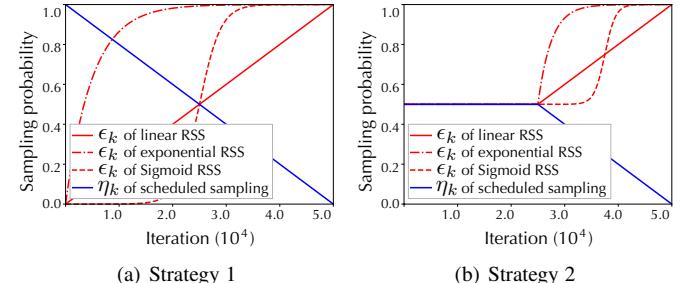


Fig. 6: Two feasible strategies to combine the reverse scheduled sampling (in red) and the original scheduled sampling (in blue).

is challenging due to the absence of some historical observations, to making one-step predictions, just as the encoder does at test time. Such a training scheme encourages the model to extract long-term, non-Markovian dynamics from the input sequence.

**Entire training scheme.** Fig. 6 shows two feasible strategies to jointly use RSS and the original scheduled sampling method, where  $\eta_k$  denotes the probability of the original scheduled sampling at forecasting timesteps. For simplicity, we make  $\eta_k$  decay linearly, although other scheduled sampling schemes could be employed (such as an exponential decay). The biggest difference between the two strategies lies in whether the variation ranges of the sampling probabilities of encoder and forecaster are close at the early stage of training. Empirically, the second strategy performs slightly better (as shown in Table 6 in Section 5.2). This suggests that in the early training stages, it is helpful to sample true frames with similar probabilities in the encoding and forecasting parts of PredRNN.

Like scheduled sampling, the RSS training strategy can be widely used in most sequence-to-sequence models beyond PredRNN to enhance the long-term modeling capability.

## 5 EXPERIMENTS

In this section, we evaluate our approach on five spatiotemporal prediction datasets. We strongly encourage readers to view <https://github.com/thuml/predrnn-pytorch> for the source code.

### 5.1 Experimental Setups

**Datasets.** We evaluate PredRNN on the following datasets for both synthetic and real-world scenarios:

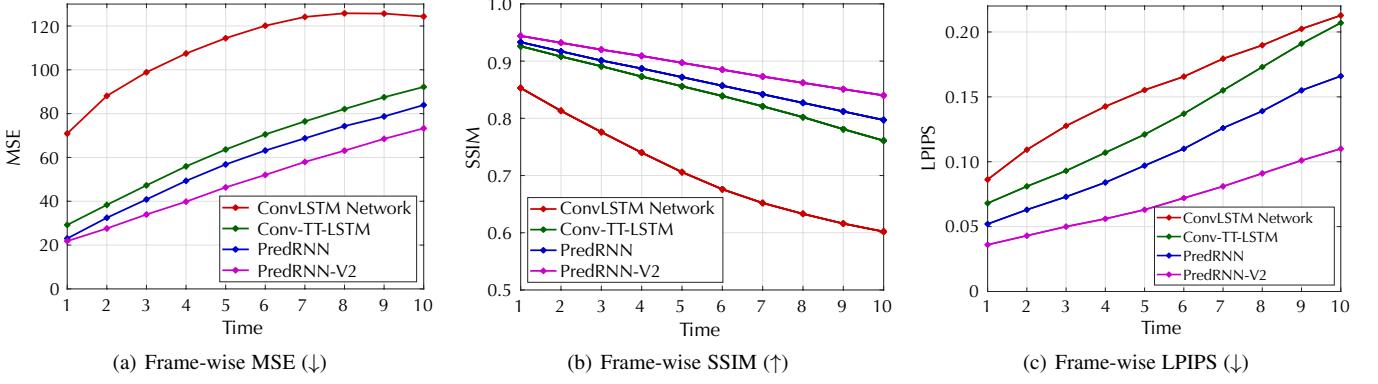
(a) Frame-wise MSE ( $\downarrow$ )(b) Frame-wise SSIM ( $\uparrow$ )(c) Frame-wise LPIPS ( $\downarrow$ )

Fig. 7: Frame-wise results on the Moving MNIST test set produced by models trained on the fixed training set.

- **Moving MNIST:** This dataset contains handwritten digits that are sampled from the static MNIST, placed at random locations, and initialized with a random speed. They bounce off the edges of the image at a certain angle. For a fair comparison, we follow two training setup from ConvLSTM [1] and CrevNet [28]. In the first setup, we use a fixed training set of 10,000 samples; while in the second one, we generate training data on the fly, which provides a larger number of training samples. We use a test set of 5,000 sequences where the digits are sampled from a different subset of static MNIST.
- **KTH [82]:** This dataset contains 6 types of human actions, *i.e.*, walking, jogging, running, boxing, hand-waving, and hand-clapping, performed by 25 persons in 4 different scenes. The videos last 4 seconds on average and have a frame rate of 25 FPS. We resize the frames to a resolution of  $128 \times 128$ . We adopt the protocol from [57], *i.e.*, persons 1-16 for training; persons 17-25 for testing, and obtain a training set of 108,717 sequences and a test set of 4,086 sequences.
- **Radar echo dataset:** This dataset contains 10,000 consecutive radar maps recorded every 6 minutes at Guangzhou, China. We transform the radar maps to pixel values and represent them as  $128 \times 128$  gray-scale images. We divide the dataset into 7,800 training sequences and 1,800 test sequences.
- **Traffic4Cast [83]:** This dataset records the GPS trajectories of consecutive traffic flows in Berlin, Moscow, and Istanbul in the form of video frames in 2019. The size of each frame is  $495 \times 436 \times 3$ . The value of each pixel corresponds to the traffic information in an area of  $100\text{m} \times 100\text{m}$  in 5 minutes, including mean speed, volume, and major traffic direction.
- **BAIR [13]:** This dataset contains the action-conditioned videos collected by a Sawyer robotic arm pushing a variety of objects. At each timestep, we have a  $64 \times 64$  RGB image and the action vector of the commanded gripper pose. We have a training set of 822,016 sequences and a test set of 4,864 sequences.

**Compared models.** We use the ConvLSTM network [1] as the primary baseline model, and compare PredRNN with the current states of the art for each dataset, including:

- CrevNet [28] for Moving MNIST and Traffic4Cast.
- Conv-TT-LSTM [29] for the KTH action dataset.
- TrajGRU [25] for precipitation forecasting.
- SV2P [35] for action-conditioned video prediction.

Notably, some existing approaches have extended PredRNN in different aspects [9, 27] or use ST-LSTM as the network backbone [28]. The differences between them are summarized in Table 1. Particularly for these approaches, the proposed contributions of

TABLE 2: Performance on the Moving MNIST test set, averaged over 10 prediction timesteps. All models are trained with a fixed training set of 10,000 sequences. For a fair comparison, we also report the model size and computation efficiency. ConvLSTM\* denotes a 4-layer network with a 256-channel hidden state each.

Model	MSE ( $\downarrow$ )	SSIM ( $\uparrow$ )	LPIPS ( $\downarrow$ )	#Params. (MB)	FLOPS (G)	Mem. (GB)
ConvLSTM [1]	103.3	0.707	0.156	16.60	80.7	2.58
ConvLSTM* [1]	62.8	0.846	0.126	65.96	320.8	5.62
CDNA [26]	97.4	0.721	-	-	-	-
VPN Baseline [67]	64.1	0.870	-	-	-	-
MIM [4]	52.0	0.874	0.079	37.37	181.7	4.22
Conv-TT-LSTM [29]	64.3	0.846	0.133	23.91	116.3	6.17
PredRNN- $\mathcal{M}$ -Only	74.0	0.851	0.109	18.30	89.02	2.37
PredRNN ( <i>Conf.</i> )	56.8	0.867	0.107	23.85	115.9	3.52
PredRNN-V2	<b>48.4</b>	<b>0.891</b>	<b>0.071</b>	23.86	116.6	3.97

memory decoupling and reverse scheduled sampling are shown to be easily combined with them and enable them to approach state-of-the-art performance. To facilitate the discussion of ablation study, we refer to different versions of PredRNN as:

- **PredRNN- $\mathcal{M}$ -Only:** This model improves the ConvLSTM network with the spatiotemporal memory flow ( $\mathcal{M}$ ). The architecture is shown in Fig. 1 (left).
- **PredRNN:** This model uses ST-LSTMs as the building blocks. It was proposed in our conference paper [2].
- **PredRNN-V2:** This is the final proposed model that improves the training process of the original PredRNN with memory decoupling and reverse scheduled sampling.

**Implementation details.** We use the ADAM optimizer [84] to train the models with a mini-batch of 2 to 16 sequences according to different datasets. Unless otherwise specified, we set the learning rate to  $10^{-4}$  and stop the training process after 80,000 iterations. We typically use four ST-LSTM layers in PredRNN to strike a balance between prediction quality and training efficiency. We set the number of channels of each hidden state to 128 and the size of convolutional kernels inside the ST-LSTM unit to  $5 \times 5$ .

## 5.2 Moving MNIST Dataset

In this dataset, future trajectories of moving digits are predictable based on enough historical observations, where part of the challenge is to infer the underlying dynamics with random initial velocities. Moreover, the frequent occlusion of multiple digits leads to complex and short-term variations of local spatial information, which brings more difficulties to spatiotemporal prediction. We train the models to predict the next ten frames given the first ten.

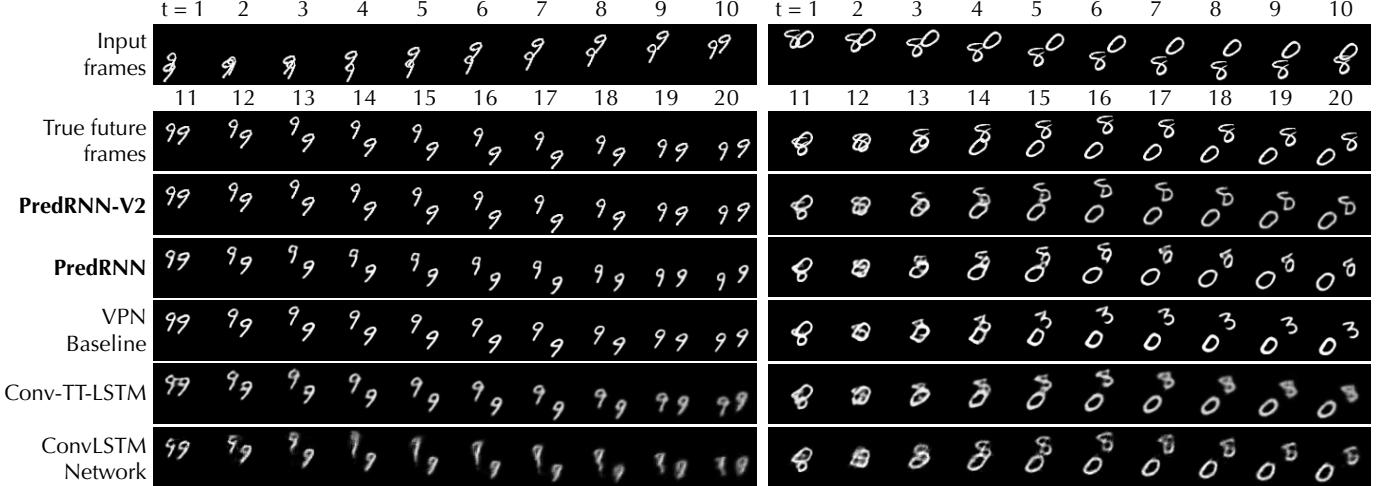


Fig. 8: Prediction examples on the Moving MNIST test set.

TABLE 3: We follow the setup of CrevNet to train the predictive models on dynamically generated Moving MNIST. The results of CrevNet are taken directly from the paper [28]. Models marked by \* have three 64-channel ST-LSTM layers with fewer parameters.

Model	MSE ( $\downarrow$ )		SSIM ( $\uparrow$ )		#Params. (MB)
	2-digit	3-digit	2-digit	3-digit	
CrevNet w/ ConvLSTM	38.5	57.2	0.928	0.886	2.77
CrevNet w/ ST-LSTM	22.3	40.6	0.949	0.916	5.00
PredRNN*	21.5	39.6	0.931	0.884	4.40
PredRNN-V2*	19.9	34.8	0.939	0.900	4.41
PredRNN	15.7	26.7	0.951	0.918	23.85
PredRNN-V2	13.4	24.7	0.958	0.928	23.86

**Quantitative results with a fixed training set.** Table 2 shows the results of all compared models averaged per frame. We adopt evaluation metrics that were widely used by previous methods: the Mean Squared Error (MSE), the Structural Similarity Index Measure (SSIM) [85], and the Learned Perceptual Image Patch Similarity (LPIPS) [86]. The difference between these metrics is that MSE estimates the absolute pixel-wise errors, SSIM measures the similarity of structural information within the spatial neighborhoods, while LPIPS is based on deep features and aligns better to human perceptions. Fig. 7 provides the corresponding frame-wise comparisons. The final PredRNN model significantly outperforms all previous approaches, and all the proposed techniques have their contributions. First, with the proposed spatiotemporal memory flow, the PredRNN- $\mathcal{M}$ -only model reduces the per-frame MSE of the ConvLSTM baseline from 103.3 down to 74.0. Second, by using the ST-LSTM in place of the ConvLSTM unit, our model further reduces the MSE down to 56.8. Finally, the employment of the memory decoupling and the reverse scheduled sampling techniques brings another 14.8% improvement in MSE (from 56.8 to 48.4). In Table 2, we also show the model size, computational efficiency, and memory usage for the sake of fair comparisons. Note that PredRNN performs better and is more efficient than a large version of the ConvLSTM network (denoted by ConvLSTM\*) with a doubled number of channels in the hidden states.

**Quantitative results with a dynamic training set.** To further compare the performance of PredRNN with the state of the art on the Moving MNIST dataset, we follow the experimental setups of recent approaches [28, 67] where models are trained on dynamically generated data with two or three flying digits. The training process

TABLE 4: An ablation study on the zigzag memory flow. We compare the baseline PredRNN with two alternatives that truncate the memory flow at specific positions in the transition path of  $\mathcal{M}_t^L$ .

Model	MSE ( $\downarrow$ )
PredRNN (w/o RSS or memory decoupling)	56.8
- Truncated at $\mathcal{M}_t^1 \rightarrow \dots \rightarrow \mathcal{M}_t^L$ (bottom-up)	57.6
- Truncated at $\mathcal{M}_t^L \rightarrow \mathcal{M}_{t+1}^1$ (top-down)	59.7

is stopped after 100,000 iterations. As shown in Table 3, PredRNN and PredRNN-V2 yield better results than those in Table 2, as they can have a better understanding of the spatiotemporal dynamics of the dataset with the growth of training steps. Notably, CrevNet also uses ST-LSTM as the recurrent unit. According to the results from its original text [28], it is superior to another ConvLSTM-based variant. More importantly, it is shown to be further improved by memory decoupling and reverse scheduled sampling.

**Qualitative comparison.** Fig. 8 shows two examples randomly sampled from the test set, where most of the frames produced by the compared models are severely blurred, especially for long-term predictions. In contrast, PredRNN produces clearer images, which means it can be more certain of future variations due to its stronger long-term modeling capabilities. When we look closely, we can see that the most challenging issues on this dataset are to make accurate predictions of the future trajectories of moving digits and to maintain the correct shape of each digit after occlusions (in the second example, the compared VPN model incorrectly predicts the digit 8 as 3 after the occlusion of 8 and 0). In both cases, the original PredRNN and the newly proposed PredRNN-V2 progressively improve the quality of the prediction results.

**Ablation studies on the memory flow.** To understand the benefit of the spatiotemporal memory flow, we compare the performance of (i) truncating the bottom-up state transition path of PredRNN from  $\mathcal{M}_t^1$  to  $\mathcal{M}_t^L$  and (ii) truncating the top-down transition path from  $\mathcal{M}_t^L$  to  $\mathcal{M}_{t+1}^1$ . Fig. 9(a) shows the normalized gradient values of  $\|\nabla_{\mathcal{H}_t^1} \mathcal{L}_{T+K}\|$  for  $1 \leq t < 20$ , which demonstrates the benefit of the zigzag memory flow in learning long-term data trends due to the ability of alleviating the problem of vanishing gradients. Table 4 gives the quantitative results in MSE, indicating that the top-down transition path contributes more to the final performance. Furthermore, in Fig. 10, we study the effect of the memory flow in long-term modeling by analyzing the saturation ratio among all

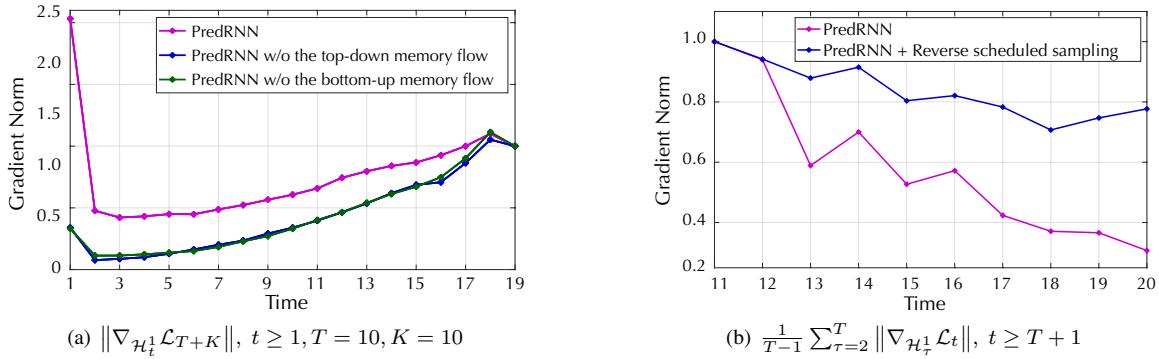


Fig. 9: Gradient analyses on the Moving MNIST test set. We use well-trained models and average the results over 100 sequences randomly sampled from the test set. **(a)** It presents the gradients of previous states concerning the loss at the last forecasting timestep, showing that the proposed memory flow benefits long-term modeling by alleviating the vanishing gradients. **(b)** It presents the accumulated gradients of the encoding states concerning the losses over the forecasting timesteps, showing the importance of RSS in encoding long-term dynamics of the inputs. We here use the RSS training strategy with an exponentially increasing  $\epsilon_k$  (from Fig. 6(b)). Both *PredRNN* and *PredRNN+RSS* take as input the historical real observations at encoding timesteps during testing.

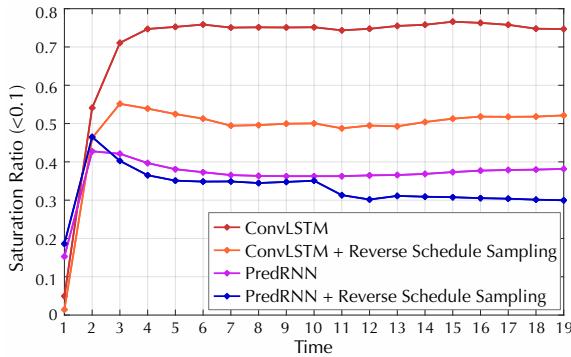


Fig. 10: An analysis of the ratios of the saturated forget gates ( $f_t < 0.1$ ) in all transitions of temporal memory states  $C_t^l$  ( $l \in \{1, 2, 3, 4\}$ ). We obtain the results by evaluating 100 samples from the Moving MNIST test set. The results demonstrate the effect of the RSS and the zigzag memory flow in long-term modeling.

elements in the forget gates ( $f_t < 0.1$ ) for the temporal memory cell  $C_t^l$ . A high saturation ratio indicates that the model tends to block the information flow of long-term trends. Compared with ConvLSTM, the spatiotemporal memory flow releases the modeling capability of  $C_t^l$  for long-term dynamics.

**Ablation study on the memory decoupling.** To show that memory decoupling facilitates both long-term and short-term dependencies, as shown in Fig. 11, we make the pixel intensity of the images change regularly or irregularly over time. Thanks to the decoupled memory cells of ST-LSTMs, our approach can respond to sudden changes more rapidly and adapt to video dynamics at different timescales. As shown by the purple curves, we also include the results of a larger ConvLSTM network with a model size significantly larger than that of *PredRNN+Memory Decoupling* (65.96 MB vs. 23.86 MB). Nonetheless, our model responds more quickly to the rapidly changing pixel intensity. In Table 5, we use the MIM model [4], which is also based on the ST-LSTM unit, to specifically show the generality of the memory decoupling loss to different network backbones.

**Ablation study on the reverse scheduled sampling.** We compare different combinations of the original and the reverse scheduled sampling techniques. From Table 6, the second strategy in Fig. 6 with an exponentially increased  $\epsilon_k$  performs best. It is because

TABLE 5: Effect of memory decoupling and reverse scheduled sampling on different network backbones (sorted by publish date).

Backbone	Decoupling	RSS	MSE ( $\downarrow$ )	Perf. gain
ConvLSTM [1]	×	×	103.3	37.8%
	○	○	64.2	
PredRNN	×	×	56.8	14.8%
	○	○	51.1	
PredRNN++ [27]	×	×	46.5	3.7%
	○	○	44.8	
E3D-LSTM [9]	×	×	44.2	4.8%
	○	○	42.1	
MIM [4]	×	×	52.0	11.9%
	○	○	47.9	
Conv-TT-LSTM [29]	×	×	64.3	8.2%
	○	○	59.0	
MotionRNN [71]	×	×	52.4	5.7%
	○	○	49.4	

TABLE 6: Ablation study on the training schemes. For RSS,  $\epsilon_k$  changes from  $\epsilon_s$  to  $\epsilon_e$ . Models are trained w/o the decoupling loss.

Method	$\epsilon_s$	$\epsilon_e$	RSS mode	MSE ( $\downarrow$ )
PredRNN	-	-	-	57.3
+ Scheduled sampling [24]	-	-	-	56.8
+ 1st strategy in Fig. 6(a)	0.0	1.0	Linear	51.8
			Sigmoid	53.9
			Exponential	51.8
+ 2nd strategy in Fig. 6(b)	0.5	1.0	Linear	51.9
			Sigmoid	50.9
			Exponential	<b>50.6</b>

the encoder-forecaster discrepancy can be effectively reduced by keeping their probabilities of sampling the true context frames close to each other in the early stage of training. To further demonstrate that the reverse scheduled sampling can contribute to learning long-term dynamics, we perform empirical analyses on the long-term gradients in Fig. 9(b) and the saturation ratio of forget gates in Fig. 10. For the gradient analysis, we evaluate the gradients of the encoder's hidden states with respect to the loss functions at different output timesteps, and average the results over the entire

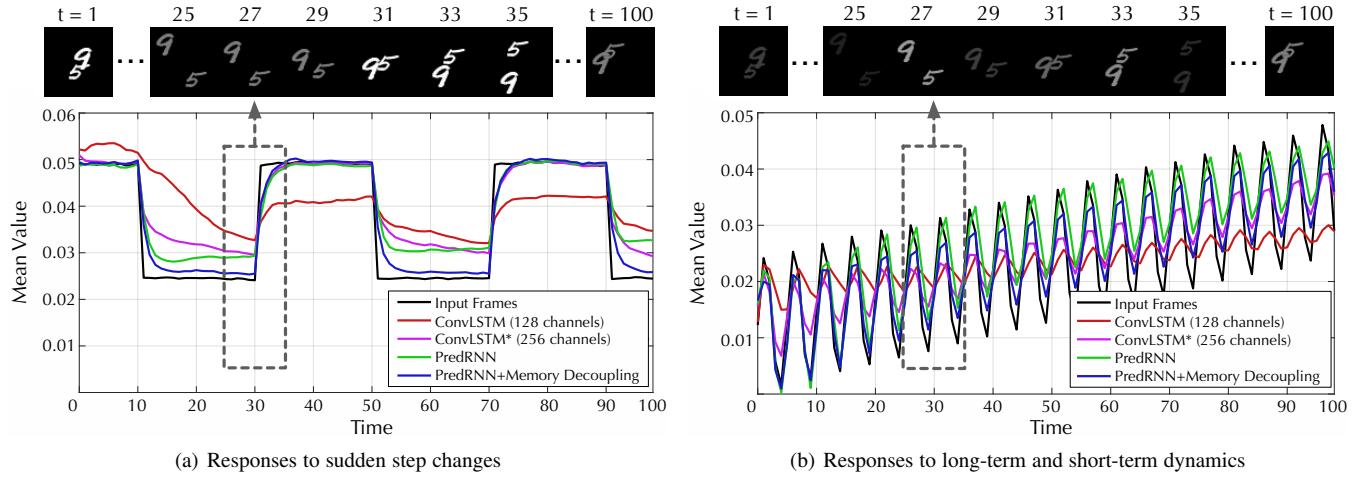


Fig. 11: Model performance under different types of temporal dynamics, where the models take as inputs the real observations at all timesteps, and the Y axis represents the mean value of predictions averaged over 100 test samples. We show two ground-truth sequences as examples at the top of the figure for a better intuitive understanding of the two types of pixel intensity changes. For each type, we apply the same rate of changes to all test samples. With memory-decoupled ST-LSTMs, our model is shown to (a) respond more rapidly to unexpected, sudden variations, and (b) simultaneously capture temporal dynamics at different timescales.

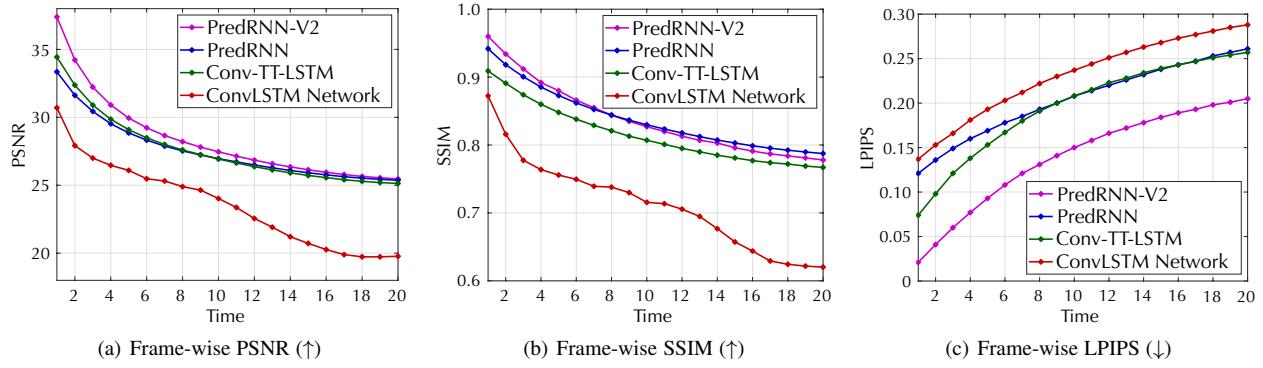


Fig. 12: Frame-wise results on KTH. The prediction horizon is 10 timesteps at training time and 20 timesteps at test time.

input sequence:  $\frac{1}{T-1} \sum_{\tau=2}^T \|\nabla_{\mathcal{H}_\tau} \mathcal{L}_t\|$ ,  $t \in [T+1, T+K]$ . The normalized gradient curves show that the context information can be encoded more effectively by using reverse scheduled sampling. In Fig. 10, we find that with RSS, both the ConvLSTM network and PredRNN maintain lower saturation ratios for the forget gates in the temporal memory cell  $\mathcal{C}_t^l$ , which indicates that RSS improves the learning process of long-term dynamics.

**Generality of the proposed techniques.** Table 5 shows the results of applying the proposed techniques to some existing models that are also based on convolutional RNNs. It is worth noting that, first, our original ST-LSTM has been the foundation of some recent architectures, thus allowing further improvements introduced by memory decoupling. Besides, the proposed reverse scheduled sampling method is shown to be a general training approach for recurrent models, enabling them to approach the state of the art. In particular, although PredRNN++ is also based on ST-LSTMs, it only achieves a 3% performance gain. The reason is that, on one hand, like RSS, the *gradient highway unit* of PredRNN++ is also to improve the long-term modeling capability; On the other hand, it models the state transitions from  $\mathcal{C}_t^l$  to  $\mathcal{M}_t^l$ , which implicitly allows  $\mathcal{M}$  to learn new information beyond what has been learned by  $\mathcal{C}$ . However, the two components mentioned above in PredRNN++ introduce additional parameters and increase the computational burden. In contrast, memory decoupling and RSS increase the number of model parameters by a negligible amount but have

TABLE 7: Results on KTH averaged over 20 future timesteps.

Model	PSNR ( $\uparrow$ )	SSIM ( $\uparrow$ )	LPIPS ( $\downarrow$ )
ConvLSTM [1]	23.58	0.712	0.231
MCnet + Residual [57]	26.29	0.806	-
TrajGRU [25]	26.97	0.790	-
DFN [87]	27.26	0.794	-
Conv-TT-LSTM [29]	27.62	0.815	0.196
PredRNN	27.55	<b>0.839</b>	0.204
PredRNN-V2	<b>28.37</b>	0.838	<b>0.139</b>

greater benefits for the final performance.

### 5.3 KTH Action Dataset

In this dataset, all compared models are trained across the 6 action categories by predicting 10 future frames from 10 observations. At test time, the prediction horizon is expanded to 20 timesteps.

We adopt the Peak Signal to Noise Ratio (PSNR) from the previous literature as the third evaluation metric, in addition to SSIM and LPIPS. Like MSE, PSNR also estimates the pixel-level similarity of two images (higher is better). The evaluation results of different methods are shown in Table 7, and the corresponding frame-wise comparisons are shown in Fig. 12, from which we have two observations: First, our models show significant improvements in both short-term and long-term predictions over the ConvLSTM network. Second, with the newly proposed memory decoupling and

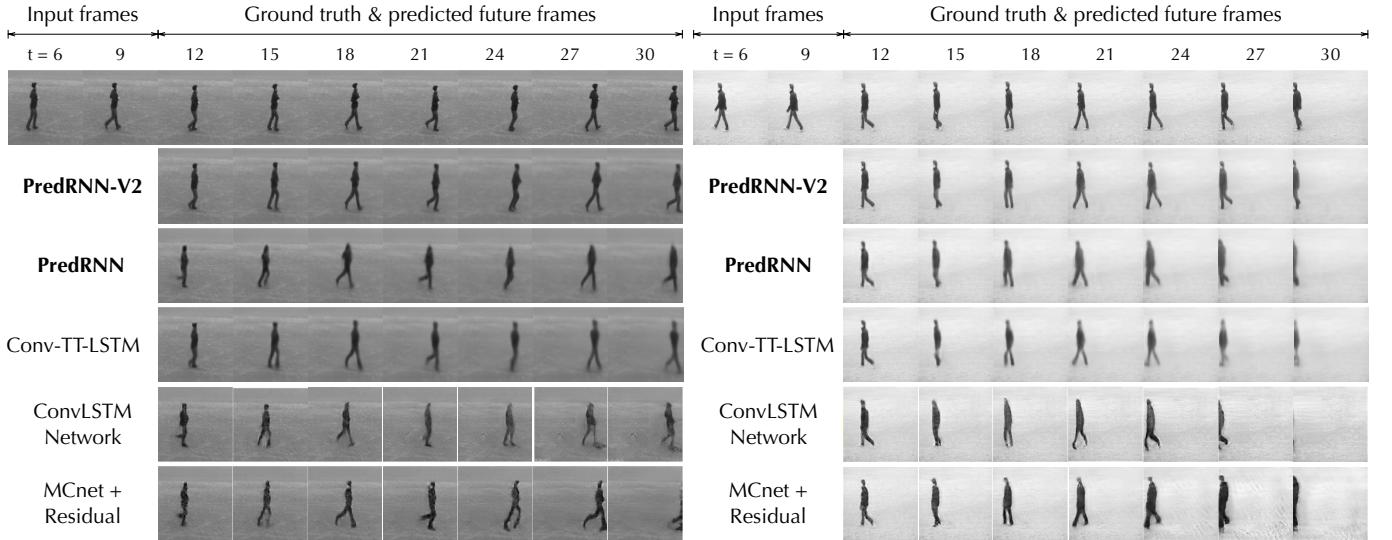


Fig. 13: Prediction examples on the KTH test set, where we predict 20 frames into the future based on the past 10 frames.

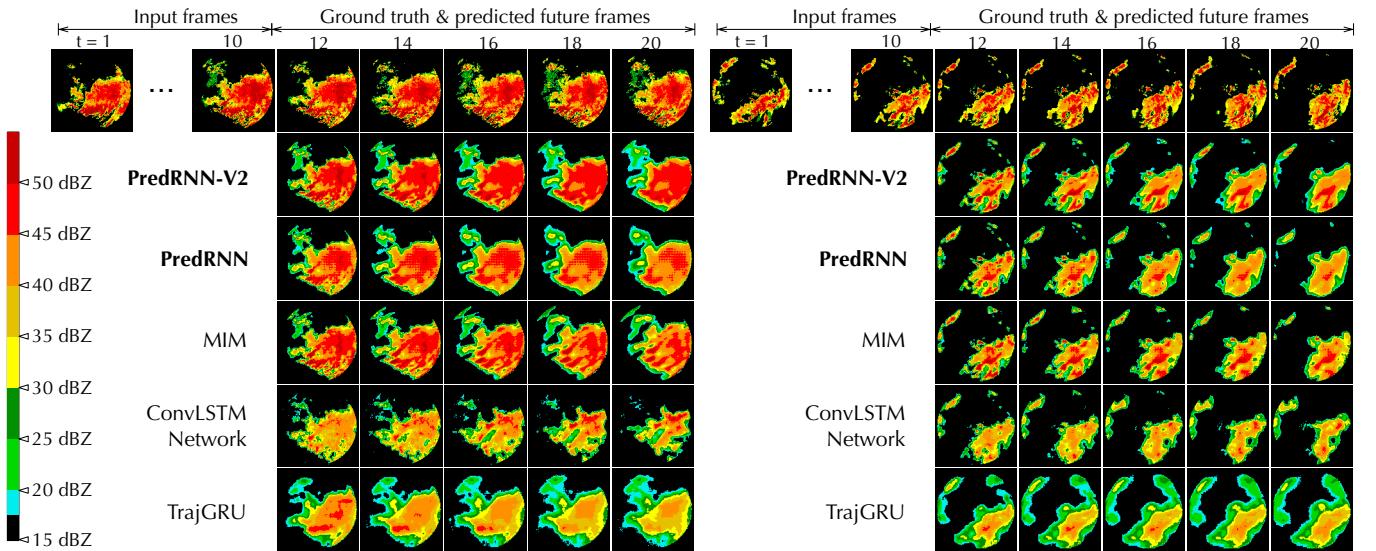


Fig. 14: Prediction examples on the radar echo test set, in which 10 future frames are generated from the past 10 observations.

reverse scheduled sampling, PredRNN-V2 improves the conference version by a large margin in LPIPS (from 0.204 to 0.139). As mentioned above, LPIPS is more sensitive to perceptual human judgments, indicating that PredRNN-V2 has a stronger ability to generate high-fidelity images. In accordance with these results, we can see from the visual examples in Fig. 13 that our approaches (especially PredRNN-V2) obtain more accurate predictions of future movement and body details. By decoupling memory states, PredRNN-V2 learns to capture the complex spatiotemporal variations from different timescales.

#### 5.4 Precipitation Forecasting from Radar Echoes

The accurate prediction of the movement of radar echoes in the next 0-2 hours is the foundation of precipitation forecasting. In this dataset, each sequence contains 10 input frames and 10 output frames, covering the historical data for the past hour and that for the future hour. It is challenging because the echoes tend to have non-rigid shapes and may move, accumulate or dissipate rapidly due to complex atmospheric physics, which makes it important to learn the dynamics in a unified spatiotemporal feature space.

TABLE 8: Quantitative results on the radar echo dataset.

Model	MSE ( $\downarrow$ )	CSI-30 ( $\uparrow$ )	CSI-40 ( $\uparrow$ )	CSI-50 ( $\uparrow$ )
TrajGRU [25]	68.3	0.309	0.266	0.211
ConvLSTM [1]	63.7	0.381	0.340	0.286
MIM [4]	39.3	0.451	0.418	0.372
PredRNN	39.1	0.455	0.417	0.358
PredRNN-V2	<b>36.4</b>	<b>0.462</b>	<b>0.425</b>	<b>0.378</b>

TABLE 9: Quantitative results on the Traffic4cast dataset.

Backbone	Recurrent unit	Decoupl.	RSS	MSE ( $10^{-3}$ )
U-Net [88]	None	$\times$	$\times$	6.992
U-Net + PredRNN	ST-LSTM	$\times$	$\times$	7.035
CrevNet [28]	ST-LSTM	$\circ$	$\circ$	<b>5.135</b>
	ConvLSTM	$\times$	$\times$	6.789
	ST-LSTM	$\times$	$\times$	6.613
	ST-LSTM	$\circ$	$\circ$	6.506

In Table 8, we compare PredRNN with three existing methods that have been shown effective for precipitation forecasting. Follow-

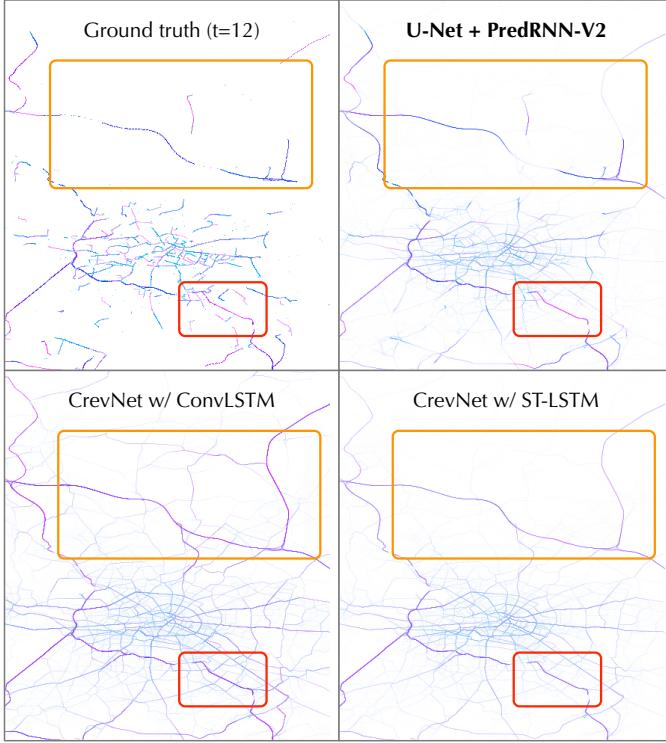


Fig. 15: Showcases at the last forecasting timestep on Traffic4cast. Please note the differences in areas in the orange and red boxes.

ing a common practice, we evaluate the predicted radar maps with the Critical Success Index (CSI). Concretely, we first transform the pixel values back to echo intensities in dBZ, and then take 30, 40 and 50 dBZ as thresholds to compute:  $\text{CSI} = \frac{\text{hits}}{\text{hits} + \text{misses} + \text{false\_alarms}}$ , where *hits* indicates the true positive, *misses* indicates the false positive, and *false\_alarms* is the false negative. From Table 8, PredRNN consistently achieves the best performance over all CSI thresholds. In Fig. 14, we visualize the predicted radar frames by mapping them into RGB space, where areas with echo values over 40 dBZ tend to have severe weather.

### 5.5 Traffic4Cast Dataset

We evaluate the proposed methods on Traffic4Cast [83], a real-world dataset with complex and drastically changing information. We follow the experimental setups from CrevNet [28], which is a competitive model on this dataset and use the data of Berlin to construct a training set of 82,080 frames and a test set of 2,160 frames. The models are trained to predict 3 future frames from 9 context frames. All pixel values are normalized to  $[0, 1]$ .

To cope with high-dimensional input frames, we apply the autoencoder architecture of U-Net [88] to the network backbone of PredRNN. Specifically, the decoder of U-Net contains four ST-LSTM layers, and the CNN encoder takes both traffic flow maps and spatiotemporal memory states as inputs. We also apply the proposed methods to CrevNet, which uses either ConvLSTM or ST-LSTM as the alternatives of recurrent units in the autoencoder architecture. Therefore, it can be further improved by memory decoupling and reverse scheduled sampling.

In Table 9, we follow the standard evaluation metric of MSE from the competition and have the following observations. Initially, the use of ST-LSTM improves the performance of the predictive models based on U-Net and CrevNet. Besides, the proposed

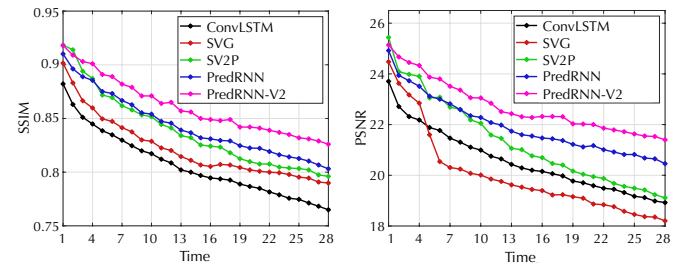


Fig. 16: Frame-wise SSIM and PSNR on the action-conditioned BAIR dataset. Note that the prediction time horizon is 10 during the training phase and extended to 28 during testing phase.

TABLE 10: Quantitative results on the BAIR robot pushing dataset averaged over 28 future timesteps in both action-free and action-conditioned experimental settings.

Inputs	Model	SSIM ( $\uparrow$ )	PSNR ( $\uparrow$ )
Action-Free	PredRNN	0.536	16.21
	PredRNN-V2	0.577	16.80
Action-Conditioned	ConvLSTM [4]	0.807	20.52
	SVG [51]	0.821	19.97
	SV2P [35]	0.837	21.32
	PredRNN	0.843	21.93
	PredRNN-V2	<b>0.861</b>	<b>22.74</b>

memory decoupling and reverse scheduled sampling achieve further improvements over the models based on vanilla ST-LSTMs. Fig. 15 provides the qualitative comparisons of different variants of our methods based on the architecture of CrevNet. As shown in the red box at  $t = 11$ , the final model with memory-decoupled ST-LSTM and reverse scheduled sampling predicts the rapid changes of traffic flows most accurately.

### 5.6 Action-Conditioned BAIR Robot Pushing Dataset

We evaluate the action-conditioned PredRNN on the BAIR dataset that is widely used by previous methods, e.g., SVG [51] and SV2P [35]. We follow the experimental setups in SV2P to train the models to predict 10 frames into the future based on the first 2 visual observations, as well as the entire action sequence covering both encoding and forecasting timesteps. During the testing phase, instead, we extend the prediction time horizon to 28 timesteps for long-term evaluation. We mainly compare the performance of our approach with that of SVG and SV2P, which are competitive probabilistic models for stochastic video prediction. Specifically for these models, we draw 100 prediction samples from the prior distribution given a single testing sequence and report the results of the sample with the best SSIM score.

From Table 10, by comparing the proposed models with the action-free PredRNN, we find that the action information plays a significant role in future image generation. The performance of action-free PredRNN is mainly affected by the lack of observable dynamics due to limited input sequence, as well as the unpredictable spatiotemporal changes when future actions are unknown. It can be concluded that Eq. (7) provides an effective method to fuse the action information into the predictive network.

More importantly, in Fig. 16, we have the following observations. First, the action-conditioned PredRNN and PredRNN-V2 outperform SVG and SV2P significantly, especially for long-term



Fig. 17: Prediction examples on the BAIR dataset, in which 28 frames are predicted given the first 2 frames and the entire action sequence at both encoding and forecasting timesteps. We zoom in the areas bounded by red boxes for clear comparison.

future prediction. Although probabilistic video prediction is useful for many downstream tasks such as epistemic POMDPs with implicit partial observability. The results indicate that in our cases in the BAIR dataset, the visual observations and action sequences provide rich information for future prediction. Second, the proposed models significantly outperform the action-conditioned ConvLSTM network, where we also apply the action fusion operation in Eq. (7) to each ConvLSTM unit. Again, the results validate the effectiveness of ST-LSTM over ConvLSTM. Third, with memory decoupling and RSS, PredRNN-V2 presents a consistent improvement over PredRNN and achieves the state of the art. Fig. 17 shows the qualitative results, in which our model predicts the future position of commanded gripper more precisely and largely enriches the details of objects.

### 5.7 Model Limitations

To explore on what conditions the proposed model contributes the most, we visualize the examples that have comparable quantitative results in SSIM and MSE with the second-best models on the KTH, radar, and BAIR datasets. As shown in Fig. 18, on KTH and BAIR, *i.e.*, datasets with clear visual structures, PredRNN-V2 significantly outperforms the other models in the visual quality of the generated future images. It makes more accurate predictions about future positions of the character (for KTH) and the robot arm (for BAIR). However, as for the radar echo dataset, Fig. 18 reveals the limitation of our approach in dealing with the special cases of the long-tail distribution of the dataset. In these cases, the radar observations

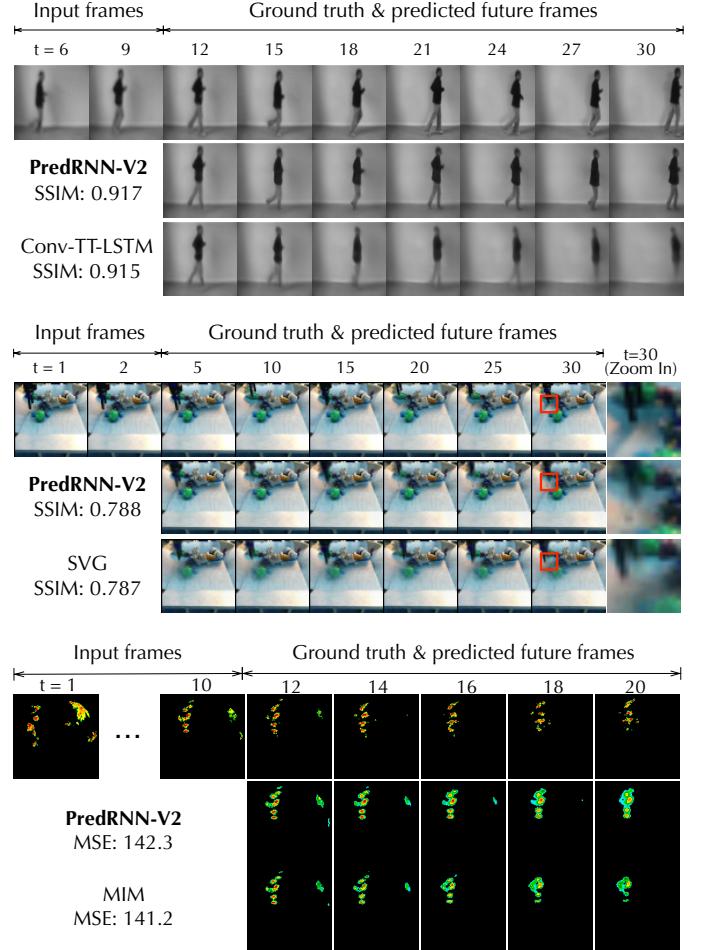


Fig. 18: Examples that PredRNN-V2 yields comparable quantitative results with the previous art.

show very sparse but still significant areas of high intensity echoes. Although PredRNN-V2 still achieves a better performance than the compared model in the forecast of future positions of the high-intensity areas, it cannot perfectly model the dynamics of the sparse echoes. Precipitation nowcasting is an extremely difficult research problem for existing spatiotemporal predictive learning approaches. A future research direction for mitigating the long-tail effect is to explicitly consider the complex physical properties and combine them with deep learning models.

## 6 CONCLUSION

In this paper, we propose a recurrent network named PredRNN for spatiotemporal predictive learning. With a new Spatiotemporal LSTM unit, PredRNN models the short-term deformations in spatial appearance and the long-term dynamics over multiple frames simultaneously. The core of the Spatiotemporal LSTM is a zigzag memory flow that propagates across stacked recurrent layers vertically and through all temporal states horizontally, which enables the interactions of the hierarchical memory representations at different levels of PredRNN. Building upon the conference version of this paper, we introduce a new method to decouple the twisted memory states along the horizontal and the zigzag pathway of recurrent state transitions. It enables the model to benefit from learning distributed representations that could cover different aspects of spatiotemporal variations. Furthermore, we

also propose a new curriculum learning strategy named reverse scheduled sampling, which forces the encoding part of PredRNN to learn temporal dynamics from longer periods of the context frames. Our approach achieves state-of-the-art performance on synthetic and natural spatiotemporal datasets, including both action-free and action-conditioned predictive learning scenarios.

## ACKNOWLEDGMENTS

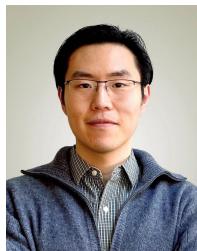
This work was supported by the National Natural Science Foundation of China (62022050, 62106144 and 62021002). M. Long was supported by Beijing Nova Program (Z201100006820041) and BNRist Innovation Fund (BNR2021RC01002). Y. Wang was supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and Shanghai Sailing Program (21Z510202133). The work was in part done when Y. Wang was a student at Tsinghua University. Y. Wang and H. Wu contributed equally to this work.

## REFERENCES

- [1] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, “Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” in *NeurIPS*, 2015, pp. 802–810.
- [2] Y. Wang, M. Long, J. Wang, Z. Gao, and S. Y. Philip, “PredRNN: Recurrent neural networks for predictive learning using spatiotemporal lstms,” in *NeurIPS*, 2017, pp. 879–888.
- [3] Z. Xu, Y. Wang, M. Long, and J. Wang, “PredCNN: Predictive learning with cascade convolutions,” in *IJCAI*, 2018, pp. 2940–2947.
- [4] Y. Wang, J. Zhang, H. Zhu, M. Long, J. Wang, and P. S. Yu, “Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics,” in *CVPR*, 2019, pp. 9154–9162.
- [5] J. Wu, E. Lu, P. Kohli, B. Freeman, and J. Tenenbaum, “Learning to see physics via visual de-animation,” in *NeurIPS*, 2017, pp. 153–164.
- [6] S. Van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber, “Relational neural expectation maximization: Unsupervised discovery of objects and their interactions,” in *ICLR*, 2018.
- [7] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, “Neural relational inference for interacting systems,” in *ICML*, 2018, pp. 2688–2697.
- [8] Z. Xu, Z. Liu, C. Sun, K. Murphy, W. T. Freeman, J. B. Tenenbaum, and J. Wu, “Unsupervised discovery of parts, structure, and dynamics,” in *ICLR*, 2019.
- [9] Y. Wang, L. Jiang, M.-H. Yang, L.-J. Li, M. Long, and L. Fei-Fei, “Eidetic 3D LSTM: A model for video prediction and beyond,” in *ICLR*, 2019.
- [10] D. Ha and J. Schmidhuber, “Recurrent world models facilitate policy evolution,” in *NeurIPS*, 2018, pp. 2450–2462.
- [11] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, “Learning latent dynamics for planning from pixels,” in *ICML*, 2019, pp. 2555–2565.
- [12] C. Finn and S. Levine, “Deep visual foresight for planning robot motion,” in *ICRA*, 2017, pp. 2786–2793.
- [13] F. Ebert, C. Finn, A. X. Lee, and S. Levine, “Self-supervised visual planning with temporal skip connections,” in *CoRL*, 2017.
- [14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.
- [15] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
- [16] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [17] I. Sutskever, J. Martens, and G. E. Hinton, “Generating text with recurrent neural networks,” in *ICML*, 2011.
- [18] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” *arXiv preprint arXiv:1409.1259*, 2014.
- [19] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *ICML*, 2014.
- [20] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *CVPR*, 2015, pp. 4694–4702.
- [21] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” in *CVPR*, 2015, pp. 2625–2634.
- [22] R. Pascanu, G. Montufar, and Y. Bengio, “On the number of response regions of deep feed forward networks with piecewise linear activations,” *arXiv preprint arXiv:1312.6098*, 2013.
- [23] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *NeurIPS*, 2014, pp. 3104–3112.
- [24] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, “Scheduled sampling for sequence prediction with recurrent neural networks,” in *NeurIPS*, 2015, pp. 1171–1179.
- [25] X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, “Deep learning for precipitation nowcasting: A benchmark and a new model,” in *NeurIPS*, 2017, pp. 5617–5627.
- [26] C. Finn, I. Goodfellow, and S. Levine, “Unsupervised learning for physical interaction through video prediction,” in *NeurIPS*, 2016, pp. 64–72.
- [27] Y. Wang, Z. Gao, M. Long, J. Wang, and P. S. Yu, “PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning,” in *ICML*, 2018, pp. 5123–5132.
- [28] W. Yu, Y. Lu, S. Easterbrook, and S. Fidler, “Efficient and information-preserving future frame prediction and beyond,” in *ICLR*, 2020.
- [29] J. Su, W. Byeon, F. Huang, J. Kautz, and A. Anandkumar, “Convolutional tensor-train LSTM for spatio-temporal learning,” in *NeurIPS*, 2020.
- [30] S. Oprea, P. Martinez-Gonzalez, A. Garcia-Garcia, J. A. Castro-Vargas, S. Orts-Escalano, J. Garcia-Rodriguez, and A. Argyros, “A review on deep learning techniques for video prediction,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [31] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, “Action-conditional video prediction using deep networks in atari games,” in *NeurIPS*, 2015, pp. 2863–2871.
- [32] M. Mathieu, C. Couprie, and Y. LeCun, “Deep multi-scale video prediction beyond mean square error,” in *ICLR*, 2016.
- [33] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, “MoCoGAN:

- Decomposing motion and content for video generation,” in *CVPR*, 2018, pp. 1526–1535.
- [34] N. Srivastava, E. Mansimov, and R. Salakhudinov, “Unsupervised learning of video representations using LSTMs,” in *ICML*, 2015, pp. 843–852.
- [35] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine, “Stochastic variational video prediction,” in *ICLR*, 2018.
- [36] D. Weissenborn, O. Täckström, and J. Uszkoreit, “Scaling autoregressive video models,” in *ICLR*, 2019.
- [37] M. Kumar, M. Babaeizadeh, D. Erhan, C. Finn, S. Levine, L. Dinh, and D. Kingma, “Videoflow: A flow-based generative model for video,” in *ICLR*, 2020.
- [38] T. Xue, J. Wu, K. Bouman, and B. Freeman, “Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks,” in *NeurIPS*, 2016.
- [39] J. Zhang, Y. Zheng, and D. Qi, “Deep spatio-temporal residual networks for citywide crowd flows prediction,” in *AAAI*, 2017, pp. 1655–1661.
- [40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *NeurIPS*, 2014, pp. 2672–2680.
- [41] E. Denton, S. Chintala, R. Fergus *et al.*, “Deep generative image models using a Laplacian pyramid of adversarial networks,” in *NeurIPS*, 2015, pp. 1486–1494.
- [42] P. Bhattacharjee and S. Das, “Temporal coherency based criteria for predicting video frames using deep multi-stage generative adversarial networks,” in *NeurIPS*, 2017, pp. 4271–4280.
- [43] X. Liang, L. Lee, W. Dai, and E. P. Xing, “Dual motion GAN for future-flow embedded video prediction,” in *ICCV*, 2017, pp. 1744–1752.
- [44] Y. Wu, R. Gao, J. Park, and Q. Chen, “Future video synthesis with object motion prediction,” in *CVPR*, 2020, pp. 5539–5548.
- [45] S. Gur, S. Benaim, and L. Wolf, “Hierarchical patch vae-gan: Generating diverse videos from a single sample,” in *NeurIPS*, 2020.
- [46] B. Liu, Y. Chen, S. Liu, and H.-S. Kim, “Deep learning in latent space for video prediction and compression,” in *CVPR*, 2021, pp. 701–710.
- [47] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra, “Video (language) modeling: a baseline for generative models of natural videos,” *arXiv preprint arXiv:1412.6604*, 2014.
- [48] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee, “Learning to generate long-term future via hierarchical prediction,” in *ICML*, 2017, pp. 3560–3569.
- [49] N. Wichters, R. Villegas, D. Erhan, and H. Lee, “Hierarchical long-term video prediction without supervision,” in *ICML*, 2018, pp. 6038–6046.
- [50] T. Kim, S. Ahn, and Y. Bengio, “Variational temporal abstraction,” *NeurIPS*, vol. 32, pp. 11570–11579, 2019.
- [51] E. Denton and R. Fergus, “Stochastic video generation with a learned prior,” in *ICML*, 2018, pp. 1174–1183.
- [52] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, “Stochastic adversarial video prediction,” *arXiv preprint arXiv:1804.01523*, 2018.
- [53] R. Villegas, A. Pathak, H. Kannan, D. Erhan, Q. V. Le, and H. Lee, “High fidelity video prediction with large stochastic recurrent neural networks,” in *NeurIPS*, 2019, pp. 81–91.
- [54] L. Castrejon, N. Ballas, and A. Courville, “Improved conditional vrnn for video prediction,” in *ICCV*, 2019, pp. 7608–7617.
- [55] J.-Y. Franceschi, E. Delasalles, M. Chen, S. Lamprier, and P. Gallinari, “Stochastic latent residual video prediction,” in *ICML*, 2020, pp. 3233–3246.
- [56] B. Wu, S. Nair, R. Martin-Martin, L. Fei-Fei, and C. Finn, “Greedy hierarchical variational autoencoders for large-scale video prediction,” in *CVPR*, 2021, pp. 2318–2328.
- [57] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee, “Decomposing motion and content for natural video sequence prediction,” in *ICLR*, 2017.
- [58] X. Bei, Y. Yang, and S. Soatto, “Learning semantic-aware dynamics for video prediction,” in *CVPR*, 2021, pp. 902–912.
- [59] E. Denton and V. Birodkar, “Unsupervised learning of disentangled representations from video,” in *NeurIPS*, 2017, pp. 4417–4426.
- [60] J.-T. Hsieh, B. Liu, D.-A. Huang, L. F. Fei-Fei, and J. C. Niebles, “Learning to decompose and disentangle representations for video prediction,” in *NeurIPS*, 2018, pp. 517–526.
- [61] N. Bodla, G. Shrivastava, R. Chellappa, and A. Shrivastava, “Hierarchical video prediction using relational layouts for human-object interactions,” in *CVPR*, 2021.
- [62] P. Zablotskaia, E. A. Dominici, L. Sigal, and A. M. Lehrmann, “Unsupervised video decomposition using spatio-temporal iterative inference,” *arXiv preprint arXiv:2006.14727*, 2020.
- [63] K. Greff, R. L. Kaufman, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner, “Multi-object representation learning with iterative variational inference,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 2424–2433.
- [64] V. L. Guen and N. Thome, “Disentangling physical dynamics from unknown factors for unsupervised video prediction,” in *CVPR*, 2020.
- [65] V. Patraucean, A. Handa, and R. Cipolla, “Spatio-temporal video autoencoder with differentiable memory,” in *ICLR Workshop*, 2016.
- [66] W. Lotter, G. Kreiman, and D. Cox, “Deep predictive coding networks for video prediction and unsupervised learning,” in *ICLR*, 2017.
- [67] N. Kalchbrenner, A. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu, “Video pixel networks,” in *ICML*, 2017, pp. 1771–1779.
- [68] W. Byeon, Q. Wang, R. Kumar Srivastava, and P. Koumoutsakos, “Contextvp: Fully context-aware video prediction,” in *ECCV*, 2018, pp. 753–769.
- [69] M. Oliu, J. Selva, and S. Escalera, “Folded recurrent neural networks for future video prediction,” in *ECCV*, 2018, pp. 716–731.
- [70] J. Xu, B. Ni, Z. Li, S. Cheng, and X. Yang, “Structure preserving video prediction,” in *CVPR*, 2018, pp. 1460–1469.
- [71] H. Wu, Z. Yao, J. Wang, and M. Long, “Motionrnn: A flexible model for video prediction with spacetime-varying motions,” in *CVPR*, 2021.
- [72] A. Graves, G. Wayne, and I. Danihelka, “Neural turing machines,” *arXiv preprint arXiv:1410.5401*, 2014.
- [73] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, “End-to-end memory networks,” in *NeurIPS*, 2015, pp. 2440–2448.
- [74] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, “Hybrid computing using a

- neural network with dynamic external memory,” *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.
- [75] G. E. Hinton, “Distributed representations,” 1984.
- [76] S. Bengio and Y. Bengio, “Taking on the curse of dimensionality in joint distributions using neural networks,” *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 550–557, 2000.
- [77] L. v. d. Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [78] A. Krogh and J. Vedelsby, “Neural network ensembles, cross validation, and active learning,” in *NeurIPS*, 1995, pp. 231–238.
- [79] Z.-H. Zhou, “Ensemble learning.” *Encyclopedia of biometrics*, vol. 1, pp. 270–273, 2009.
- [80] S. Chiappa, S. Racaniere, D. Wierstra, and S. Mohamed, “Recurrent environment simulators,” in *ICLR*, 2017.
- [81] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, “Dream to control: Learning behaviors by latent imagination,” in *ICLR*, 2020.
- [82] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: a local svm approach,” in *ICPR*, 2004.
- [83] IARAI, “Traffic4cast 2019: Traffic map movie forecasting.” <https://www.iarai.ac.at/traffic4cast/2019-competition/>, 2019.
- [84] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.
- [85] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [86] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *CVPR*, 2018, pp. 586–595.
- [87] B. De Brabandere, X. Jia, T. Tuytelaars, and L. Van Gool, “Dynamic filter networks,” in *NeurIPS*, 2016, pp. 667–675.
- [88] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *MICCAI*, 2015.



**Yunbo Wang** received the BE degree from Xi'an Jiaotong University in 2012, and the ME and PhD degrees from Tsinghua University in 2015 and 2020. He received the CCF Outstanding Doctoral Dissertation Award in 2020, advised by Philip S. Yu and Mingsheng Long. He is now an assistant professor at the AI Institute and the Department of Computer Science at Shanghai Jiao Tong University. He does research in deep learning, especially predictive learning, spatiotemporal modeling, and model-based decision making.



**Haixu Wu** received the BE degree in computer software from Tsinghua University in 2020. He is working towards the ME degree in computer software at Tsinghua University. His research interests include machine learning and computer vision.



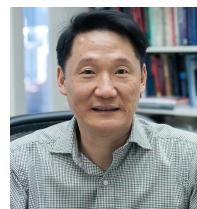
**Jianjin Zhang** received the ME degree in computer software from Tsinghua University in 2019. His research interests lie on the intersection of machine learning, time series analysis, and natural language processing.



**Zhifeng Gao** received the ME degree in computer software from Tsinghua University in 2019. His research interests include machine learning and big data systems.



**Jianmin Wang** received the BE degree from Peking University, China, in 1990, and the ME and PhD degrees in computer software from Tsinghua University, China, in 1992 and 1995, respectively. He is a full professor with the School of Software, Tsinghua University. His research interests include big data management systems and large-scale data analytics. He led to develop a product data & lifecycle management system, which has been deployed in hundreds of enterprises in China. He is leading to develop a big data platform in the National Engineering Lab for Big Data Software.



**Philip S. Yu** received his BS in electrical engineering from the National Taiwan University, and his MS and PhD also in electrical engineering from Stanford University in 1978. He is a Distinguished Professor at the University of Illinois at Chicago and Tsinghua University. Yu holds over 300 US patents, is ACM Fellow and IEEE Fellow, is Editor-in-Chief of ACM Transactions on Knowledge Discovery from Data, and has been awarded several awards by IBM and the IEEE. Yu's research interests are in the fields of data mining, social network, privacy preserving data publishing, data stream, database systems, and Internet applications and technologies.



**Mingsheng Long** received the BE degree in electrical engineering and the PhD degree in computer science from Tsinghua University in 2008 and 2014 respectively. He was a visiting researcher in computer science, UC Berkeley from 2014 to 2015. He is an associate professor with the School of Software, Tsinghua University. He serves as Area Chairs of major machine learning conferences (ICML, NeurIPS, and ICLR). His research is persistently dedicated to theories and algorithms of machine learning, with special interests in transfer, adaptation, and data-efficient learning, foundation deep models, learning with spatiotemporal and scientific knowledge.