MS-LSTM: Exploring Spatiotemporal Multiscale Representations in Video Prediction Domain

Zhifeng Ma[a], Hao Zhang[a,*] and Jie Liu[b]

[a]Faculty of Computing, Harbin Institute of Technology, Harbin, China
[b]International Research Institute for Artificial Intelligence, Harbin Institute of Technology, Shenzhen, China

ARTICLE INFO

Keywords:
Video prediction
deep learning
LSTM

ABSTRACT

The drastic variation of motion in spatial and temporal dimensions makes the video prediction task extremely challenging. Existing RNN models obtain higher performance by deepening or widening the model. They obtain the multi-scale features of the video only by stacking layers, which is inefficient and brings unbearable training costs (such as memory, FLOPs, and training time). Different from them, this paper proposes a spatiotemporal multi-scale model called MS-LSTM wholly from a multi-scale perspective. On the basis of stacked layers, MS-LSTM incorporates two additional efficient multi-scale designs to fully capture spatiotemporal context information. Concretely, we employ LSTMs with mirrored pyramid structures to construct spatial multi-scale representations and LSTMs with different convolution kernels to construct temporal multi-scale representations. Detailed comparison experiments with eight baseline models on four video datasets show that MS-LSTM has better performance but lower training costs.

1. Introduction

As a fundamental yet essential research task in predictive learning, video prediction or called spatiotemporal predictive learning has aroused widespread research interest in the computer vision community. It benefits many aspects in real life, such as traffic flow prediction [9, 22, 52], human motion trajectory prediction[74, 7, 5], robot control [81, 78], and precipitation nowcasting [77, 83, 43, 13, 46]. Video prediction is a special case of self-supervision, where the model needs to estimate upcoming future frames from given historical frames at the pixel level. Training such models does not require any annotated data, yet models need to capture a notion of the complex dynamics of real-world phenomena (such as physical interactions) to generate coherent sequences [3].

Due to the inherent uncertainty of natural videos, predictions from models rapidly degrade over time as uncertainty grows, converging to an average of the possible future outcomes, visually represented as blurriness [48]. To address this issue, prior approaches choose to deepen or widen the pioneer model. Since ConvLSTM [58] incorporates convolution into Long Short-Term Memory (LSTM) [27] to simultaneously model spatiotemporal dynamics, a series of subsequent models use it as the cornerstone to continuously refresh the prediction performance, and their evolution can be roughly expressed as ConvLSTM: → TrajGRU [59] and PredRNN [72] → PredRNN++ [70] → MIM [75] → MotionRNN [79] → PrecipLSTM [45]. These practices of increasing the depth and width of ConvLSTM lead to the increase of model parameters and the expansion of model capacity, which is beneficial to deal with complex nonlinear transformations. Nevertheless, the side effect is the skyrocketing training cost, such as memory (most intractable), FLOPs, and training time, which is unbearable in high-resolution and resource-constrained scenes.

Besides depth and width, there are two other essential factors that can improve model performance in the computer vision domain, they are cardinality [80] and scale [17]. Cardinality reduces the number of parameters by grouping convolutions and then increases the number of groups to improve the performance while ensuring the same parameters as the original convolution. Unfortunately, cardinality also aggravates the memory footprint like width and depth [44]. Obtaining multi-scale representations in vision tasks requires feature extractors to use a large range of receptive fields to describe objects at different scales [17], which can usually be achieved by stacking convolutional layers (VGGNet [60], ResNet [26]), or using larger convolution kernels (GoogLeNet [64], SKNet [37], Res2Net [17]), or using pooling (downsampling, LeNet [33], AlexNet [32]). Among them, pooling is the most efficient way. In detail, the former two increase the depth and width of the network respectively, while the latter just adds some layers without parameters and allows the model to perform at low resolutions (low overhead). In the past few years, the advances in backbone CNN (Convolutional Neural Network) architectures have demonstrated a trend toward more effective and efficient multi-scale representations. Analogously, how to design a more efficient network architecture is also the key to further improving the performance of convolutional RNN (Recurrent Neural Network).

Based on the following two facts: (1) The boom of CNN has witnessed the rapid development of multi-scale technology. (2) A basic problem of video prediction is how to efficiently learn a good spatiotemporal representation of video speculation or inference. In this work, we absorb the essence of the multi-scale design of CNN, combine it with the spatiotemporal characteristics of video, and propose a simple

*Corresponding author
zhh1000@hit.edu.cn (H. Zhang)
ORCID(s): 0000-0002-6769-2115 (H. Zhang)

yet efficient multi-scale approach called Multi-Scale LSTM (MS-LSTM). Unlike most existing methods that enhance the layer-wise multi-scale representation strength of RNN, MS-LSTM improves the multi-scale representation ability at a more granular level. MS-LSTM incorporates two additional efficient multi-scale designs to fully capture spatiotemporal contextual information. From a spatial perspective, we stack RNN layers like ordinary RNNs and use downsampling to further obtain multi-scale representations. Whereas, video prediction is a pixel reconstruction task. Hence, we add a symmetric decoder to restore the output of the encoder to the original pixel space, resulting in a mirrored pyramid structure like UNet [55]. From a temporal perspective, we use LSTMs with different sizes of kernels instead of a single kernel to build multi-scale representations, which we call Multi-Kernel LSTM (MK-LSTM). MK-LSTM follows the Markov assumption like ConvLSTM, using different scales of cellular memory to describe the deformation of objects at different scales over time. In addition, we analyze the training cost (params, memory, FLOPs, time) of MS-LSTM and its components, and analyze the reason (increase of receptive field) for the performance improvement brought by the multi-scale architecture. It should be noted that the utilization of the big kernel gains rewards but it brings extra training burden, which can partly be resolved by the introduction of downsampling, which can ensure that our model sufficiently learns multi-scale features while bringing modest training cost. Our contributions can be summarized as follows:

- We propose a new spatiotemporal multiscale model named MS-LSTM, which incorporates three orthogonal multiscale designs.
- MS-LSTM employs stacked LSTMs with multiple convolution kernels (MK-LSTM) to form a mirrored pyramid structure, which can fully acquire the spatiotemporal context representation and generates video from coarse to fine.
- We theoretically analyze the training cost and performance of MS-LSTM. Experiments on four datasets with eight competing models have proved that MS-LSTM has lower training occupation but higher performance.

2. Related Work

2.1. Video Prediction Models

The mainstream models can be divided into four categories: Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Generative Network, and Attention-based Network. They use different network structures to encode different inductive biases (prior assumptions) into deep networks. The CNN models are dominated by UNet [55] family [2, 49, 25, 57]. However, CNN implicitly assumes complex changes in spatial appearance and may therefore fall short in learning long-term dependencies [73]. The RNN models are dominated by ConvLSTM [58] family, which is a large and thriving family. Their family members have TrajGRU [59], MCNet [67], PredRNN [72], E3D-LSTM [71], PredRNN++ [70], CubicLSTM [16], MIM [75], SA-ConvLSTM [38], PhyDNet [22], TMU [84], [35], MAU [6], MotionRNN [79], STRPM [5], PredRNN-V2 [73], PrecipLSTM [45], and so on. In order to alleviate the fuzzy prediction problem caused by L_2 loss, which prefers to obtain the average states of the future, generative models explore the prediction problem by adding regularity constraints to the loss function. These models are either based on adversarial training (GAN) [47, 81, 65, 69, 53, 28, 36] or variational autoencoders (VAEs) [10, 1, 39] or both [34, 74, 14]. Recently, with the popularity of the new neural network architecture ViT [12], some works, like VideoGPT [82], TATS [20], Earthformer [19], and MaskViT [24], try to apply various Transformer [66] variants to video prediction domain. Compared with the other three types of networks, the design of RNN models introduces appropriate prior knowledge, neither requiring complex tricks to maintain training nor relying on oceans of samples to learn potential laws. Consequently, this paper mainly explores RNN models for spatiotemporal predictive learning.

2.2. Multi-scale Models

Visual patterns occur at multi-scales in natural scenes. It is of critical importance to design good features for multi-scale stimuli for visual tasks [17]. Unsurprisingly, multi-scale structures have been widely adopted in the computer vision territory, like image classification [64, 26, 17, 40], image segmentation [55, 68], object detection [54, 23], image deblurring [8], video recognition [15, 41], video segmentation [63], video deblurring [86], and so on. However, due to the complexity of the video prediction task, few works apply multiscale structures to RNN models efficiently.

Recently, CMS-LSTM [4] devotes to acquiring contextual representation and spatiotemporal multi-scale representation for video prediction. It acquires contextual representations by interacting the current input with the previous output state multiple times and uses multi-patch attention to design multi-scale structures. Unfortunately, the multiscale design of CMS-LSTM is inefficient, which brings quadratic complexity and can only be used in low-resolution scenes (Section 5.3).

3. Preliminaries

3.1. Problem Formulation

For a dynamical system (e.g., a video clip), from the spatial view, if we need to record c measurements of a certain local area ($h \times w$ grid points) at any time, we can express it in the form of a tensor $X_t \in \mathbb{R}^{c \times h \times w}$; from the temporal view, we can express it as a sequence of tensors $\{X_0, ..., X_{m-1}, X_m, ..., X_{m+n-1}\}$. Let $X = \{X_0, ..., X_{m-1}\}$ be the model input and $Y = \{X_m, ..., X_{m+n-1}\}$ be the true target. The video prediction problem is to infer the most probable future sequence $\tilde{Y}$ with length-n given the historical sequence X with length-m. In this paper, we train a neural network parametrized by θ to solve such a task. We use stochastic gradient descent to find a set of parameters θ^* that maximizes the likelihood of producing the true target sequence Y given the input data X:

$$\theta^* = \arg\max_{\theta} \; P(Y|X;\theta). \tag{1}$$

3.2. ConvLSTM

ConvLSTM [58] is the pioneer recurrent model to deal with this issue. It incorporates convolution into both the input-to-state and the state-to-state transitions of LSTM [27], which can model the spatial variation and temporal dynamics simultaneously. Since ConvLSTM introduces correct prior knowledge, it is well-suited for spatiotemporal inference tasks. The ConvLSTM unit is formulated as

$$\begin{aligned}
i_t &= \sigma(W_{ix} * X_t^l + W_{ih} * H_{t-1}^l),\\
f_t &= \sigma(W_{fx} * X_t^l + W_{fh} * H_{t-1}^l),\\
g_t &= \tanh(W_{gx} * X_t^l + W_{gh} * H_{t-1}^l),\\
C_t^l &= f_t \odot C_{t-1}^l + i_t \odot g_t,\\
o_t &= \sigma(W_{ox} * X_t^l + W_{oh} * H_{t-1}^l),\\
H_t^l &= o_t \odot \tanh(C_t^l),
\end{aligned} \tag{2}$$

where W_{**} are the parameters to be optimized; X_t^l, H_t^l, and C_t^l represent input, hidden state, and cell state of layer l at time t respectively; i_t, f_t, g_t, and o_t stand for input gate, forget gate, input modulation gate, and output gate at time t respectively; '$*$' is the convolution operation; '$\odot$' is the Hadamard product; 'σ' and 'tanh' denote the sigmoid and tanh activation function respectively. A common practice is to stack more ConvLSTM units to obtain higher performance. Fig. 1 exhibits the structure of a 6-layer ConvLSTM.

Eq. (2) shows that convolutions are important parts of ConvLSTM. Thus, we can extend the training cost of the convolution to ConvLSTM. Suppose b, c, h, w, and k represent the batch size, input or output channel, height, width, and kernel size, respectively. The parameter complexity of a convolution layer is c^2k^2, and the parameter complexity of a ConvLSTM unit is Uc^2k^2 ($U = 8$). The FLOPs (computation complexity) of a convolution layer is $2bc^2hwk^2$, and the FLOPs of a ConvLSTM unit is $2Ubc^2hwk^2$, which is based on the assumption that we only consider the FLOPs of convolution, ignoring the FLOPs of the activation function, Hadamard product, and addition (they're relatively small [51]). Suppose we use the output of the convolution layer or the gradient of the output to represent the space complexity of the convolution layer, which is used for computing the gradients of the error with respect to the parameters by the chain rule (Section 4.3.1). The space complexity of a convolution layer is $bchw$, and the space complexity of a ConvLSTM unit is $\tilde{U}bchw$. We believe that $\tilde{U} > U$, and $\tilde{U}$ increases when U increases. Because there are multiple activation layers and Hadamard product layers in addition to convolutional layers in the ConvLSTM unit, and their output sizes are the same as that of convolution.

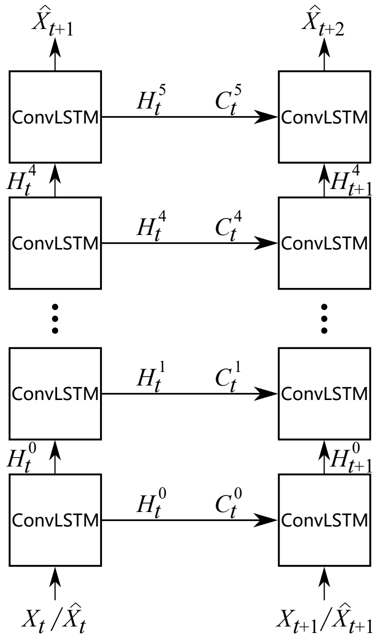


Fig. 1: The architecture of ConvLSTM. It only uses depth to obtain the multi-scale representation.

3.3. ConvLSTM Variants

Since Shi et al. [58] invent the predecessor model, many variants of ConvLSTM have been proposed. This paper mainly conducts a comparative study with several classic and representative RNN models, which all adopt a similar architecture to ConvLSTM (Fig. 1). TrajGRU [59] brings optical flow into the hidden state, in which the recurrent connections are dynamically determined. PredRNN [72] introduces an additional spatiotemporal memory cell to propagate information across both horizontal and vertical directions with the highway connection. PredRNN++ [70] increases the transition depth and reorganizes the memories of PredRNN in a cascade fashion. To enhance the ability of PredRNN on modeling high-order dynamics, MIM [75] introduces more memory cells to process non-stationary and stationary information. MotionRNN [79] inserts the MotionGRU between the layers of MIM, which is based on the principle that physical world motions can be decomposed into transient variation and motion trend. PredRNN-V2 [73] decouples the cell state and the spatiotemporal memory cell of PredRNN and introduces the reverse scheduled sampling strategy. PrecipLSTM [45] adds meteorological spatial memory and meteorological temporal memory on the basis of PredRNN for precipitation nowcasting. These networks are getting wider (TrajGRU, PredRNN, PredRNN-V2, MIM, PrecipLSTM) and deeper (PredRNN++, MotionRNN). As a result, they bring limited improvement in model performance but introduce significant growth in training cost. In high-resolution and resource-constrained scenes, they will not be up to the generation task. Models with low GPU memory usage, fast training speed, and high prediction accuracy may be more popular.

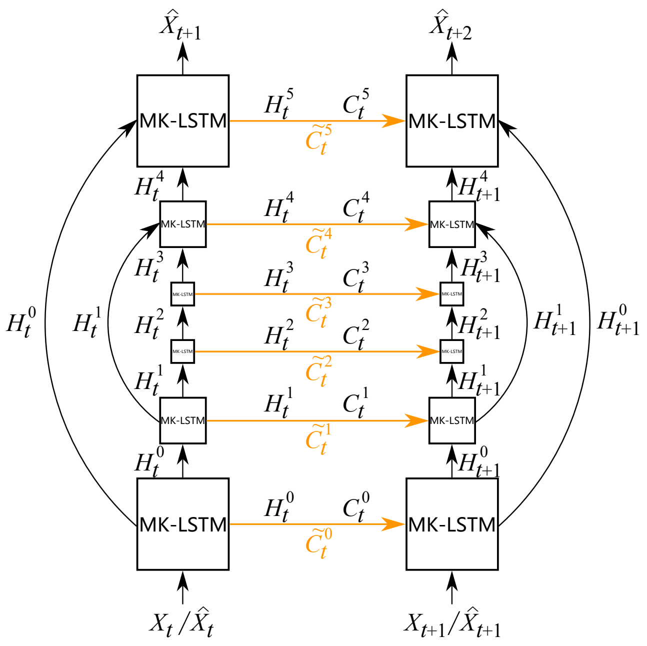

$\hat{X}_{t+1}$
$\hat{X}_{t+2}$
MK-LSTM
H_t^5 C_t^5 $\tilde{C}_t^5$
H_t^4 C_t^4 $\tilde{C}_t^4$
H_t^3 C_t^3 $\tilde{C}_t^3$
H_t^2 C_t^2 $\tilde{C}_t^2$
H_t^1 C_t^1 $\tilde{C}_t^1$
H_t^0 C_t^0 $\tilde{C}_t^0$
H_{t+1}^4 H_{t+1}^3 H_{t+1}^2 H_{t+1}^1 H_{t+1}^0
$X_t/\hat{X}_t$
$X_{t+1}/\hat{X}_{t+1}$

Fig. 2: The architecture of MS-LSTM. It uses depth, downsampling, and multiple kernels to obtain multi-scale representations. The model performs one-step predictions along the spatial axis (vertical direction) while passing hidden states between layers. The model extrapolates future frames along the time axis (horizontal direction) while passing hidden and cell states over time. Skip connections ("+") are represented by curves to combine the features of the encoder and decoder at the same scale, enabling the model to generate static features easily [10]. The yellow symbols and lines represent the newly added multi-scale cell memory and tensor flow.

4. Multi-Scale LSTM

Due to the high dimensionality and uncertainty of natural videos, extracting a robust representation from raw pixel values is an overly complicated task. The per-pixel variability between consecutive frames causes exponential growth in the prediction error on the long-term horizon [48]. Common models struggle with blurriness, and pixel-level video prediction is still challenging.

A straightforward idea is that if we hope to foretell the future, we need to memorize as many historical events as possible. When we recall something that happened before, we do not just recall object movements, but also recollect visual appearances from coarse to fine [72]. Motivated by this, we present a new recurrent architecture called Multi-Scale LSTM (MS-LSTM), which combines the multi-scale design of traditional CNNs with the spatiotemporal motion property of videos. MS-LSTM (6 layers, Fig. 2) integrates three orthogonal multi-scale designs, which are depth, downsampling, and multiple convolution kernels, where the first two constitute Spatial Multi-Scale LSTM (SMS-LSTM), and the first and last constitute Temporal Multi-Scale LSTM (TMS-LSTM). The use of multi-scale features makes it easy to capture the motion of objects at different scales, where the small scale focuses on the contour and shape transformation of the object, while the large scale focuses on detailed features, such as the evolution of precipitation or the trajectories of human limbs. We experimentally verify that MS-LSTM has strong spatiotemporal modeling capabilities but consumes fewer training resources (Section 5). We'll cover the specifics below.

4.1. Spatial Multi-Scale LSTM

Limited by the scale of the convolution kernel, convolution can only capture the short-distance spatial dependence in the image [47]. One way is to use stacked convolution layers. The other way is to use the pooling operation. Both have brought about an increase in the spatial receptive field, making the model see wider, which laid the foundation for today's CNN. The convolutional RNN models (like PredRNN [72], PredRNN++ [70], MIM [75], MotionRNN [79]) in the field of video prediction have absorbed the essence of "deep" learning, and most of them adopt the method of stacking more modules to obtain a wider spatiotemporal receptive field. However, unlike convolution operations, recursive operations cannot be parallelized, resulting in a surge in memory usage and training time. Therefore, blindly increasing the depth will no longer be the most effective way. The use of pooling will allow the model to better sense contextual information and reduce training requirements by working at low resolutions. Regrettably, the use of downsampling to capture spatial dependencies has been poorly explored among convolutional RNN models.

In this paper, we improve the existing RNN models by using the familiar approaches in CNN, stacking RNN layers but interspersing downsampling. Yet using downsampling will bring about a loss of image resolution, which is unacceptable for the pixel-level prediction task that requires the output to have the same resolution as the input. Inspired by UNet [55], we introduce a symmetric decoder to restore the output of the encoder to the original pixel space, resulting in a mirrored pyramid structure. Specifically, supposing we stack 6 layers of RNN (Fig. 2), then the bottom three layers make up the encoder and the top three layers make up the decoder. We gradually decrease the resolution of the encoder but gradually increase the resolution of the decoder. The downsampling and upsampling operations are implemented by max pooling and bilinear interpolation, respectively. Besides, we add skip connections ("+") between the encoder and decoder at the same scale to enable the model to generate static features easily [10]. Ultimately, we obtain a spatial multi-scale RNN/LSTM.

It should be emphasized that the RNN unit can be replaced by any existing model block, such as ConvLSTM, TrajGRU, PredRNN, PredRNN++, MIM, MotionRNN, PrecipLSTM, etc. Given that their basic units are either too complex (purely for novelty) or do not introduce multi-scale design, we redesign a new RNN unit (MK-LSTM) that is relatively simple and introduces the temporal multi-scale design, which will be presented in the next section. The SMS-LSTM in the subsequent experimental part of this paper refers to the one that uses the ConvLSTM unit.

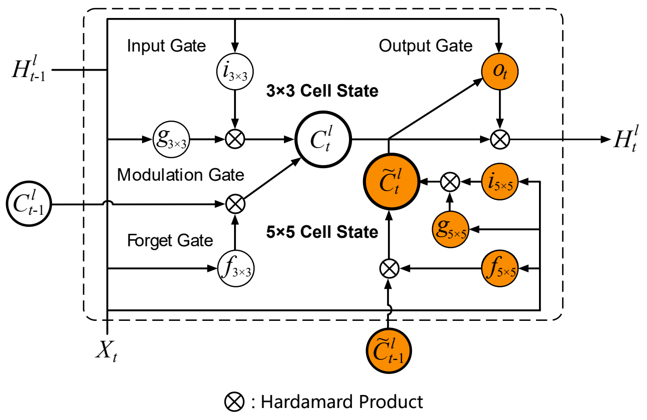


Fig. 3: The architecture of MK-LSTM. Orange-filled circles denote the differences between MK-LSTM and ConvLSTM (White-filled circles).

4.2. Temporal Multi-Scale LSTM

In addition to stacking layers and using pooling operations, GoogLeNet also utilizes the Inception composed of parallel filters of different sizes to enhance the multiscale representation capability. It also motivates some follow-up work. In theory, superimposing two convolutional layers with 3×3 kernels is equivalent to a convolutional layer with 5×5 kernel, Res2Net [17] forms an equivalent multi-kernel layer like GoogLeNet by adding shortcuts between parallel group convolutions. SKNet [37] propose a dynamic selection mechanism that can select 3×3 or 5×5 kernel in a soft-attention manner. Inspired by these works, we design a multi-scale convolutional LSTM to explore fine-grained temporal multiscale representations, which we call Multi-Kernel LSTM (MK-LSTM). The Temporal Multi-Scale LSTM (TMS-LSTM) will be obtained by stacking multiple MK-LSTM units. The intuitive view of TMS-LSTM is to replace ConvLSTM units in Fig. 1 with MK-LSTM units.

MK-LSTM (Fig. 3 and Eq. 3) follows the Markov assumption like ConvLSTM, using different scales of cellular memory to describe the deformation of objects at different scales over time. We only employ two types of convolution kernels: 3×3 and 5×5. There are two reasons: (1) For 1×1 kernel, the receptive field of states will not grow as layers stack. (2) For larger kernels, later states have larger receptive fields and are related to a wider range of the input. However, using an excessively large convolution kernel will lead to a rapid increase in training consumption. Based on the above reasons, we construct MK-LSTM using two ConvLSTMs with 3×3 and 5×5 kernel respectively, a compromise between performance and consumption. The equation of the MK-LSTM unit is shown as follows:

$$\begin{aligned}
i_{3\times3} &= \sigma(W_{ix} * X_t^l + W_{ih} * H_{t-1}^l),\\
f_{3\times3} &= \sigma(W_{fx} * X_t^l + W_{fh} * H_{t-1}^l),\\
g_{3\times3} &= \tanh(W_{gx} * X_t^l + W_{gh} * H_{t-1}^l),\\
C_t^l &= f_{3\times3} \odot C_{t-1}^l + i_{3\times3} \odot g_{3\times3},\\
i_{5\times5} &= \sigma(W'_{ix} * X_t^l + W'_{ih} * H_{t-1}^l),\\
f_{5\times5} &= \sigma(W'_{fx} * X_t^l + W'_{fh} * H_{t-1}^l),\\
g_{5\times5} &= \tanh(W'_{gx} * X_t^l + W'_{gh} * H_{t-1}^l),\\
\tilde{C}_t^l &= f_{5\times5} \odot \tilde{C}_{t-1}^l + i_{5\times5} \odot g_{5\times5},\\
o_t &= \sigma(W_{ox} * X_t^l + W_{oh} * H_{t-1}^l + W_{oc} * C_t^l\\
&\quad + W'_{ox} * X_t^l + W'_{oh} * H_{t-1}^l + W'_{oc} * \tilde{C}_t^l),\\
H_t^l &= o_t \odot \tanh(W_{1\times1} * [C_t^l, \tilde{C}_t^l]),
\end{aligned} \tag{3}$$

where W_{**}, W'_{**}, and $W_{1\times1}$ are the parameters to be optimized; X_t^l and H_t^l represent input and hidden state of layer l at time t respectively; C_t^l and $\tilde{C}_t^l$ represent cell states of different kernel size; $i_{*\times*}$, $f_{*\times*}$, and $g_{*\times*}$ stand for input gate, forget gate, and input modulation gate of different kernel size respectively; o_t stands for output gate at time t; '$*$' is the convolution operation; '$\odot$' is the Hadamard product; 'σ' and 'tanh' denote the sigmoid and tanh activation function respectively.

MK-LSTM adds additional multi-scale cellular memory on top of ConvLSTM, which makes it simple for the model to recall past object changes at different scales and perform long-term predictions. Furthermore, using different convolution kernels results in different sizes of receptive fields, which makes it easy to capture motion at different speeds, where the larger convolution kernel can capture faster motion while the smaller convolution kernel can capture slower motion [58]. This characteristic has been confirmed in experiments (Section 5). From the production of the Moving MNIST [62] and KTH [56] datasets, it can be intuitively seen that there are different rates of movement. The TaxiBJ [85] and Germany [2] datasets implicitly contain different rates of movement, because the speed of the movement of traffic and precipitation should also be random. It should be noted that the utilization of multiple kernels, likes depth, gains rewards but brings extra training burden, which can partly be resolved by the introduction of downsampling in SMS-LSTM, which ensures that MS-LSTM can thoroughly learn multi-scale representations without causing excessive training cost.

4.3. Analysis of Training Cost and Performance

4.3.1. Analysis of Training Cost

The main conclusion in this section is: Compared with ConvLSTM (TMS-LSTM), SMS-LSTM (MS-LSTM) has the same parameters and training time but less memory and FLOPs. Compared with ConvLSTM (SMS-LSTM), TMS-LSTM (MS-LSTM) has larger parameters, training time, memory, and FLOPs. Compared with ConvLSTM, MS-

LSTM has larger parameters, moderate training time, moderate memory, and moderate FLOPs because of the neutralizing effect of SMS-LSTM. These conclusions are verified in Section 5.2.2. Below we will introduce the analysis details.

From Section 3.2, we know that the parameter complexity of a ConvLSTM unit is Uc^2k^2, the computation complexity (FLOPs) of a ConvLSTM unit is $2Ubc^2hwk^2$, the space complexity (layer output) of a ConvLSTM unit is $\tilde{U}bchw$. Since the MK-LSTM unit can be viewed as two ConvLSTM units using different convolution kernels, we can extend these training cost representations of ConvLSTM units to MK-LSTM units.

Params Analysis. Since the parameters of the ConvLSTM (MK-LSTM) unit are independent of the scale (hw) and the max pooling layer, the bilinear interpolation layer, and skip connections ("+") will not increase the parameters of the model, SMS-LSTM (MS-LSTM) has the same parameters as ConvLSTM (TMS-LSTM). In addition, TMS-LSTM (MS-LSTM) has more parameters than ConvLSTM (SMS-LSTM), which is because it uses more convolutions (U) and larger convolution kernels (k).

Memory Analysis. During the model training process, there are mainly three parts of memory usage, model memory (M_{par}), optimizer memory ($3M_{par}$ for Adam [31]), and output memory ($2M_{out}$) [61]. The model memory is used to store model parameters, the optimizer memory is used to store the gradient and momentum buffer of the parameter, and the output memory consists of two equal parts, which are the forward output memory (M_{out}) to store the layer output and the backward output memory to store the output gradient [18]. In summary, the total memory footprint of the model during training $M_{all} = 4M_{par} + 2M_{out}$.

Since SMS-LSTM (MS-LSTM) has the same parameters as ConvLSTM (TMS-LSTM), SMS-LSTM (MS-LSTM) has the same model memory (M_{par}) as ConvLSTM (TMS-LSTM). However, the output memory of a ConvLSTM (MK-LSTM) unit is proportional to the scale (hw). SMS-LSTM (MS-LSTM) uses small-scale ConvLSTM (MK-LSTM) units, so it has fewer forward output memory (M_{out}) than ConvLSTM (TMS-LSTM), that is, fewer total memory usage M_{all}. In addition, TMS-LSTM (MS-LSTM) has more total memory footprint than ConvLSTM (SMS-LSTM), which is because it uses more convolutions (more U, more $\tilde{U}$, more M_{out}) and more parameters (more M_{par}).

FLOPs Analysis. Since the FLOPs of a ConvLSTM (MK-LSTM) unit is proportional to the scale (hw) and SMS-LSTM (MS-LSTM) uses small-scale ConvLSTM (MK-LSTM) units, SMS-LSTM (MS-LSTM) has fewer FLOPs than ConvLSTM (TMS-LSTM). In addition, TMS-LSTM (MS-LSTM) has more FLOPs than ConvLSTM (SMS-LSTM), which is because it uses more convolutions (U) and larger convolution kernels (k).

Time Analysis. Theoretically, the smaller the feature size, the fewer times to calculate the convolution, and the lower the time complexity. However, in practical applications, convolution operations are generally converted to matrix operations (img2col in Caffe [30] or unfold in PyTorch [50]) to speed up calculations, and the current optimization technology for matrix operations is very mature, which eventually leads to the time complexity of ConvLSTM being insensitive to scale (hw). Therefore, the time complexity of SMS-LSTM (MS-LSTM) is not much different from ConvLSTM (TMS-LSTM). But the time complexity of TMS-LSTM (MS-LSTM) is larger than ConvLSTM (SMS-LSTM) due to the fact that it uses more convolution layers (U) with larger convolution kernels (k), which cannot be parallelized.

4.3.2. Analysis of Performance

We find no work on the interpretability of spatiotemporal predictive learning, except for [29]. [29] considers video generation as a coarse-to-fine synthesis process (decoding part), relying on extending the present and erasing the past mechanism to capture spatiotemporal dynamics (encoding part). In view of the fact that there are relatively many studies on the interpretability of CNN-based models, most of them analyze the CNN (encoding part, usually consists of convolution and pooling layers) from the perspective of receptive fields, such as [42], [11], and [21]. We also try to understand the working principle of the convolutional RNN encoder from the perspective of the receptive field. For the decoding generation part of convolutional RNN, we agree with [29] that decoding should be a process from coarse to fine.

We analyze the efficiency of our model by taking RNN with 6 layers as an example, where the first 3 layers can be regarded as an encoder for encoding the current frame, and the last 3 layers can be regarded as a decoder for decoding the next frame. Assuming that the activation function (σ and tanh) will not change the receptive field, nor will element-wise addition and element-wise multiplication (Hadamard product), then from Eq. (2) we can see that the receptive field of the output (H_t^l) of a ConvLSTM unit should be the same as the receptive field of the output ($W_{*x} * X_t^l$ or $W_{*h} * H_{t-1}^l$) of a convolution layer. That is, in theory, using a ConvLSTM layer and using a convolution layer will get the same receptive field. Theoretically, stacking two convolution layers with 3×3 kernels has a receptive field equivalent to one convolution layer with a 5×5 kernel, then the theoretical receptive field of the ConvLSTM encoder is 7×7. However, the theoretical receptive field of the SMS-LSTM encoder will reach approximately 15×15, because sub-sampling increases the receptive field size multiplicatively [42]. On the basis of SMS-LSTM, we replace the ConvLSTM unit in it with the MK-LSTM unit using a large convolution kernel to continually increase the receptive field. This leads to the receptive field of the MS-LSTM encoder being much larger than 15×15. Obviously, the MS-LSTM encoder can capture longer-range spatiotemporal context than the ConvLSTM encoder, which will be of great help when modeling complex spatiotemporal changes. Further, the poor performance of the ConvLSTM encoder also indirectly affects the effect of decoding prediction, while the MS-LSTM decoder can obtain multi-scale context informa-

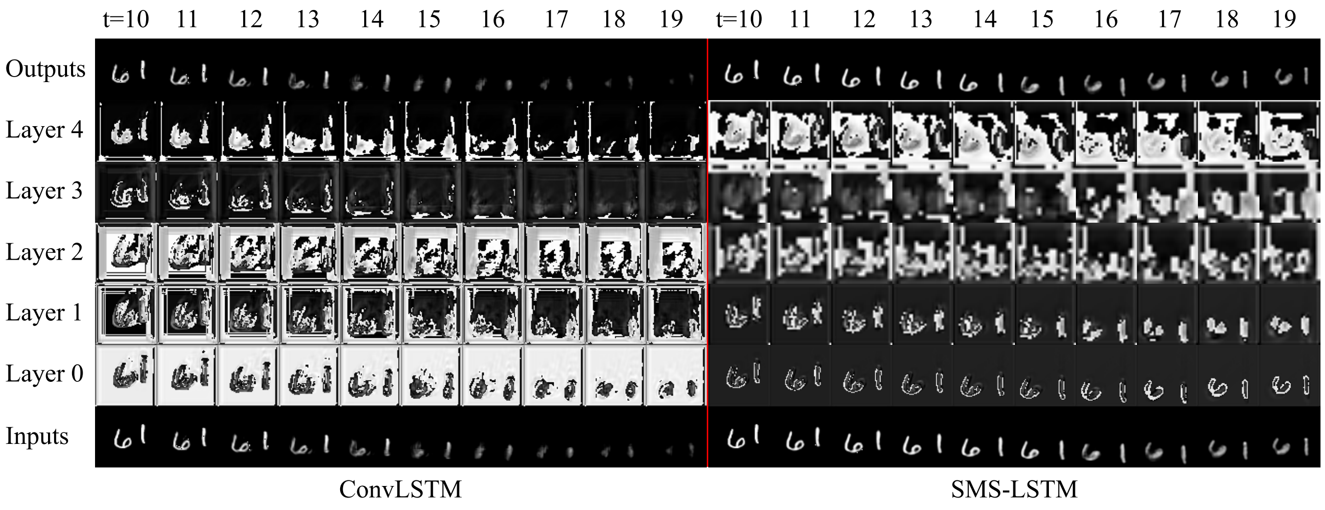


Fig. 4: The layer outputs of ConvLSTM and SMS-LSTM on the Moving MNIST dataset.

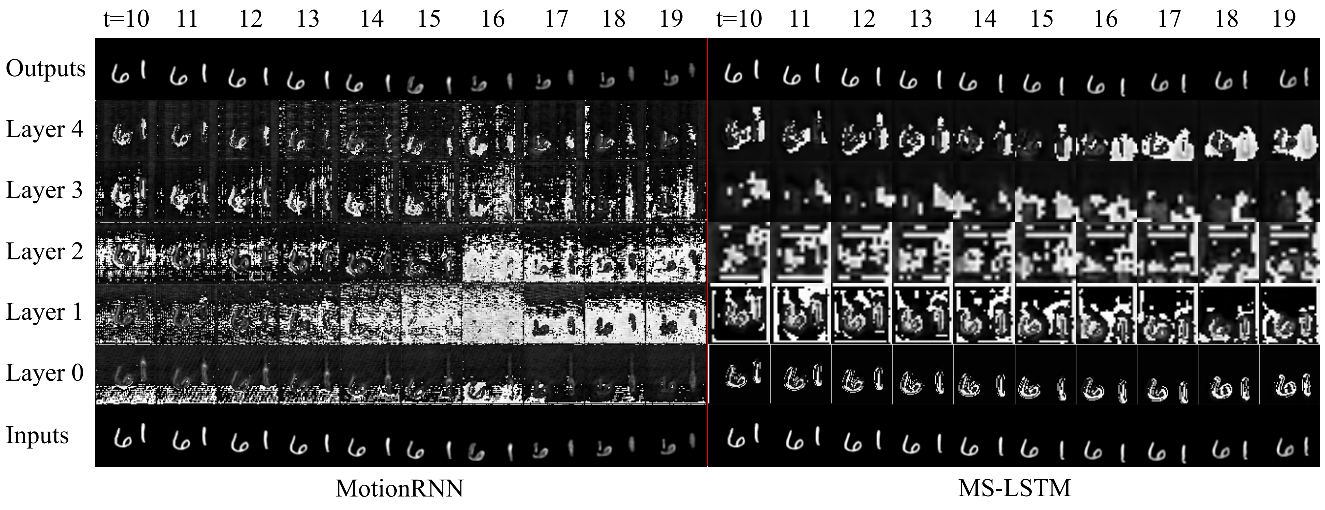


Fig. 5: The layer outputs of MotionRNN and MS-LSTM on the Moving MNIST dataset.

tion from skip connections. Furthermore, the use of multiscale layers makes it easy to model the motion of objects at different scales, where the small scale focuses on the contour and shape changes of the object, while the large scale focuses on some detailed features, such as the changes of clouds in a local area or the movement trajectories of hands and feet.

We print the output of each layer of the MS-LSTM, SMS-LSTM, and ConvLSTM encoders in the experimental part (Section 5.2.1). These all verify the above analysis: the encoder of MS-LSTM has a larger receptive field while the decoder of MS-LSTM follows the law and decodes from coarse to fine, which leads to better performance of MS-LSTM.

5. Experiments

We conduct experiments on a synthetic dataset (Moving MNIST [62]) and three real-world spatiotemporal sequence datasets (TaxiBJ [85], KTH [56], Germany [2]).

5.1. Implementation Details

We apply the same experimental settings for all models to make fair comparisons. All models use a similar structure that is stacked with 6 layers of RNNs. We use Adam [31] as the optimizer and set the initial learning rate to 0.0003. We optimize the model with the $L_1 + L_2$ loss. In the training phase, the mini-batch is set to 4 for all datasets, and the training process is stopped after 40, 15, 20, and 20 epochs for Moving MNIST, TaxiBJ, KTH, and Germany, respectively. The kernel size of RNN is 3×3 (3×3 and 5×5 for MS-LSTM) and the channel of the hidden state is 32. All experiments are tested on a machine equipped with 4 NVIDIA Tesla A100 GPUs, each of them having almost 40GB of memory. We run all experiments three times and use the average results for quantitative evaluation.

5.2. Moving MNIST

We generate Moving MNIST in the same way as [62]. The dataset contains a total of 15,000 sequences, and the sample ratio of the training set and test set is 7:3. Each sequence contains 20 frames, 10 frames are used for training, and 10 frames are used for prediction. Each image in the sequence contains 2 numbers and the spatial size is 64 × 64. The numbers in the training set and the test set are randomly obtained from the training and test set of the static MNIST dataset [33] respectively, such that the spatial information of the test data would not be overly exposed during the training process. These numbers bounce off the border at random positions, random speeds, and random directions. Although the extrapolation method of each sequence is fixed, the extrapolation method between different sequences is different.

5.2.1. *Visualization of Layer Outputs*

Fig. 4 shows the inputs, outputs of the intermediate layers, and outputs of ConvLSTM and SMS-LSTM in one sequence prediction process in the later stage of training. (1) Analysis of the encoder part: First, the numbers in the feature maps of the two RNNs are gradually getting fatter, which indicates that the receptive field is increasing and motion is captured. However, the numbers in SMS-LSTM are fatter than ConvLSTM, i.e. larger spatiotemporal receptive fields. Secondly, the feature map of SMS-LSTM is becoming more and more abstract, and we can no longer distinguish the numbers in layer 2, only the rough outline. In contrast, we can consistently discern the digits in the feature map of the ConvLSTM (limited to the earlier moments, the later moments recursively predict poorly, the same below). (2) Analysis of the decoder part: the decoder of SMS-LSTM recovers the resolution from coarse to fine, layer 3 predicts the outline, and layer 4 predicts the details. But the decoder of ConvLSTM does not have this reasonable process, and the numbers in it can always be distinguished. (3) Overall analysis: SMS-LSTM captures motion by rapidly expanding the receptive field, and completes the prediction task from coarse to fine. ConvLSTM expands the receptive field slowly, does not learn abstract advanced features, and lacks a stepwise image reconstruction process, resulting in poor performance.

Fig. 5 shows the inputs, outputs of the intermediate layers, and outputs of MotionRNN [79] and MS-LSTM in one sequence prediction process in the later stage of training. The encoding and decoding process of MS-LSTM is not much different from SMS-LSTM. However, from the comparison of Fig. 4 and Fig. 5, it can be found that the numbers in layer 0 of MS-LSTM are fatter (larger receptive field) than those of SMS-LSTM, which is more obvious in the comparison of layer 1. The reason for this phenomenon is that we introduce TMS-LSTM (MK-LSTM with large convolution kernel) on the basis of SMS-LSTM (stacked layer and downsampling), which enables MS-LSTM to capture richer spatiotemporal context and faster motion than MS-LSTM. The encoding and decoding process of MotionRNN is not much different from ConvLSTM. The spatiotemporal receptive fields of both encoders are slowly expanding, and there is no coarse-to-fine decoding process in their decoders. Furthermore, neither learns abstract high-level features, since we can always discern the digits in the feature map. Compared with ConvLSTM, MotionRNN introduces numerous temporal memory cells, which is beneficial to long-term prediction. However, its encoder cannot obtain a rich spatial receptive field, which makes it difficult to cope with complex and fast nonlinear transformations, which ultimately leads to its performance being far inferior to MS-LSTM.

Table 1
Ablation study on the Moving MNIST dataset.

Models	Params	Memory	FLOPs	Time	MSE↓
ConvLSTM	0.4M	3.6G	34.4G	1.9H	90.36
SMS-LSTM	0.4M	2.7G	15.1G	2.0H	63.90
TMS-LSTM	1.9M	9.5G	147.3G	5.0H	67.70
MS-LSTM	1.9M	5.2G	64.5G	5.0H	52.63

Table 2
Training cost comparison on the Moving MNIST dataset. We employ a single machine with four graphics cards for training and use Distributed Data Parallel (DDP) of PyTorch [50] to accelerate the training process. In this case, each card trains one batch, and the memory and FLOPs of the four cards is the same. The table below shows the memory footprint and FLOPs of a single card, which also applies to experiments on other datasets.

Model	Params	Memory	FLOPs	Time
ConvLSTM [58]	0.4M	3.6G	34.4G	1.9H
PredRNN [72]	0.9M	6.3G	69.8G	4.9H
PredRNN++ [70]	1.4M	8.8G	94.7G	4.5H
PredRNN-V2 [73]	0.9M	7.3G	70.8G	7.9H
MIM [75]	1.8M	11.0G	143.0G	9.5H
MotionRNN [79]	1.9M	11.8G	146.2G	32.7H
MS-LSTM	1.9M	5.2G	64.5G	5.0H

Table 3
Quantitative comparison on the Moving MNIST dataset.

Model	SSIM↑	MSE↓	MAE↓
ConvLSTM [58]	0.810	90.36	142.5
PredRNN [72]	0.845	66.17	118.0
PredRNN++ [70]	0.846	65.63	119.0
PredRNN-V2 [73]	0.855	61.57	112.9
MIM [75]	0.855	63.76	113.4
MotionRNN [79]	0.862	59.62	108.9
MS-LSTM	**0.879**	**52.63**	**97.9**

5.2.2. *Ablation Study*

As shown in Table 1, both Spatial Multi-Scale LSTM (SMS-LSTM) and Temporal Multi-Scale LSTM (TMS-LSTM) bring performance improvements. Besides, as analyzed in Section 4.3.1, the use of pooling does not change the parameters and training time but reduces memory and FLOPs, while the use of large convolution kernels brings additional parameters, memory, FLOPs, and training time. Since SMS-LSTM brings down the training cost (memory (most tricky) and FLOPs) while TMS-LSTM brings up the training cost (all), we combine SMS-LSTM and TMS-

LSTM to construct MS-LSTM to obtain optimal performance without causing excessive training cost.

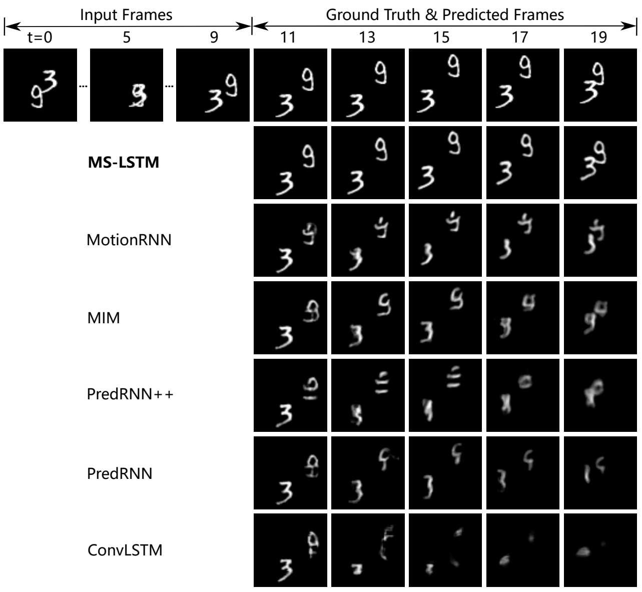


Fig. 6: Qualitative comparison on the Moving MNIST dataset.

Table 4
Training cost comparison on the TaxiBJ dataset.

Model	Params	Memory	FLOPs	Time
ConvLSTM [58]	0.4M	2.1G	3.2G	0.8H
TrajGRU [59]	0.5M	2.3G	3.6G	8.1H
PredRNN [72]	0.9M	2.4G	6.4G	1.9H
PredRNN++ [70]	1.4M	2.7G	8.7G	1.8H
MIM [75]	1.8M	2.8G	13.2G	3.7H
MotionRNN [79]	1.9M	2.9G	13.5G	11.8H
CMS-LSTM [4]	1.1M	5.6G	11.5G	9.4H
MS-LSTM	1.9M	2.3G	5.9G	2.0H

Table 5
Framewise MSE comparison on the TaxiBJ dataset.

Model	Frame 1↓	Frame 2↓	Frame 3↓	Frame 4↓
ConvLSTM [58]	0.199	0.268	0.315	0.356
TrajGRU [59]	0.193	0.264	0.305	0.352
PredRNN [72]	0.170	0.222	0.257	0.287
PredRNN++ [70]	0.162	0.200	0.234	0.268
MIM [75]	0.164	0.208	0.251	0.288
MotionRNN [79]	0.147	0.188	0.228	0.263
CMS-LSTM [4]	0.169	0.210	0.249	0.289
MS-LSTM	**0.141**	**0.173**	**0.199**	**0.230**

5.2.3. *Comparison of the Training Cost and Performance*

From Table 2, we can see that MS-LSTM has the same amount of parameters as MotionRNN, but the training resources required to train MS-LSTM are much lower than MotionRNN. Specifically, the memory occupation, FLOPs, and training time of MS-LSTM is about $\frac{1}{2}$, $\frac{1}{2}$, and $\frac{1}{6}$ of MotionRNN respectively, which is almost less than those of PredRNN. In addition, the quantitative experiments of MS-LSTM also surpassed MotionRNN (Table 3), which proves the significance of the introduction of spatiotemporal multiscale structure. For example, compared to MotionRNN, the MSE of MS-LSTM is reduced by 11.7% from 59.62 to 52.63.

The example in Fig. 6 is challenging because there is a severe occlusion in the input sequence. Occlusion can be regarded as an information bottleneck, where the mean and variance of the spatiotemporal process change drastically, indicating the existence of high-order non-stationarity [75]. The image generated by MS-LSTM is more satisfactory than other models, and the ambiguity is not obvious. In fact, we can't even tell the numbers in the last frame generated by other models. We can conclude that MS-LSTM shows more ability in capturing non-stationary changes of complex time-space sequences.

5.3. TaxiBJ Traffic Flow

The TaxiBJ dataset [85] contains taxicab trajectory data and meteorology data in four time periods in Beijing, which are collected by the sensors inside the cars. The trajectory data have two types of crowd flows, that is a $8 \times 32 \times 32 \times 2$ heat map, and the last dimension denotes the entering and leaving traffic flow intensities in the same area. We choose the entering traffic flow data for training. In detail, we use 4 known frames to predict the next 4 frames (traffic conditions for the next two hours). Then, we follow the experimental settings in MIM [75] and split the data into the training set and test set, in which the training set contains 19,560 samples while the test set contains 1,344 samples.

5.3.1. *Comparison of the Training Cost and Performance*

The quantitative experimental result is shown in Table 4 and Table 5. We can get the same conclusions as above. MS-LSTM requires less training cost but has higher accuracy. The qualitative experimental result is shown in Fig. 7, which compares the absolute difference between true and predicted images. Obviously, the prediction of MS-LSTM is the most accurate among all compared models.

Recently, there is a competing multiscale RNN model published in 2022, which is CMS-LSTM [4]. CMS-LSTM uses multi-patch attention to design multi-scale structures. Unfortunately, as shown in Table 4 and Table 5, its multi-scale design is expensive and inefficient. Although its parameters do not exceed MS-LSTM, its memory usage is more than twice that of MS-LSTM, which is because the attention in CMS-LSTM brings quadratic space complexity. This also causes CMS-LSTM to only be used in low-resolution scenes, and we can't even run it on other datasets. The thing reflected in Table 5 is that CMS-LSTM performs poorly, it does not even surpass PredRNN++, which shows that the multi-scale design of CMS-LSTM is inefficient.

5.4. KTH Human Action

The KTH dataset [56] contains six human actions (walking, jogging, running, boxing, hand waving, and hand clap-

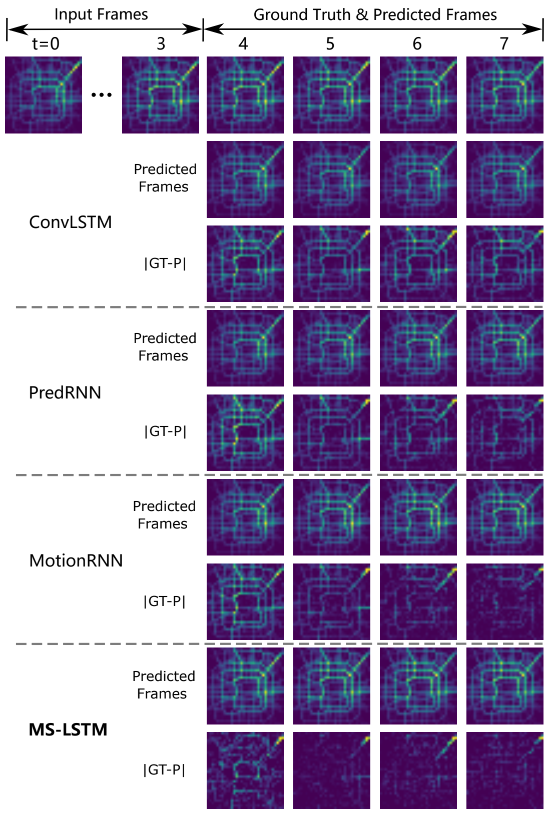
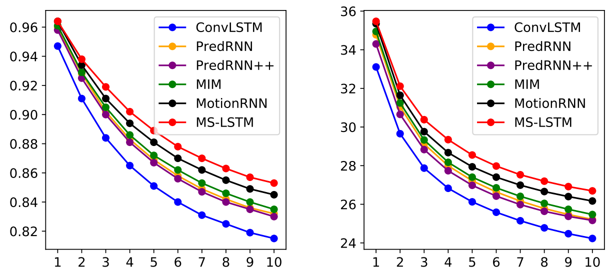


Fig. 7: Qualitative comparison on the TaxiBJ dataset. |GT-P| denotes the absolute difference frame between ground truth and predicted frames. Prediction is the most accurate when there is nothing in the difference frame.

Table 6
Training cost comparison on the KTH dataset.

Model	Params	Memory	FLOPs	Time
ConvLSTM [58]	0.4M	8.3G	137.7G	1.0H
TrajGRU [59]	0.5M	18.1G	154.2G	9.4H
PredRNN [72]	0.9M	18.4G	279.3G	2.2H
PredRNN++ [70]	1.4M	27.7G	378.7G	2.8H
MIM [75]	1.8M	35.7G	571.9G	4.3H
MotionRNN [79]	1.9M	38.7G	584.8G	12.1H
MS-LSTM	1.9M	14.5G	257.9G	2.1H

ping), performed by 25 subjects in four different scenarios: outdoors, outdoors with scale variations, outdoors with different clothes, and indoors. All sequences are shot on a homogeneous background using a static camera with a frame rate of 25fps. The raw sequence is downsampled to a spatial resolution of 160×120 pixels, with an average length of 4 seconds. In this paper, we choose the categories of walking, jogging, and running, which mainly represent lower body movements at different rates. We resize video frames into 128×128 pixels and crop the frame based on the text provided in the dataset to ensure that humans always appear in the image. We use persons 1-16 for training and 17-25 for testing. During the training and testing phase, the sliding window for all actions is 20, and we use 10 frames to predict



(a) SSIM (b) PSNR

Fig. 8: Framewise SSIM and PSNR comparison on the KTH dataset. The bigger SSIM and PSNR, the better.

the next 10 frames, and the stride for walking and jogging is 10 while running is 3.

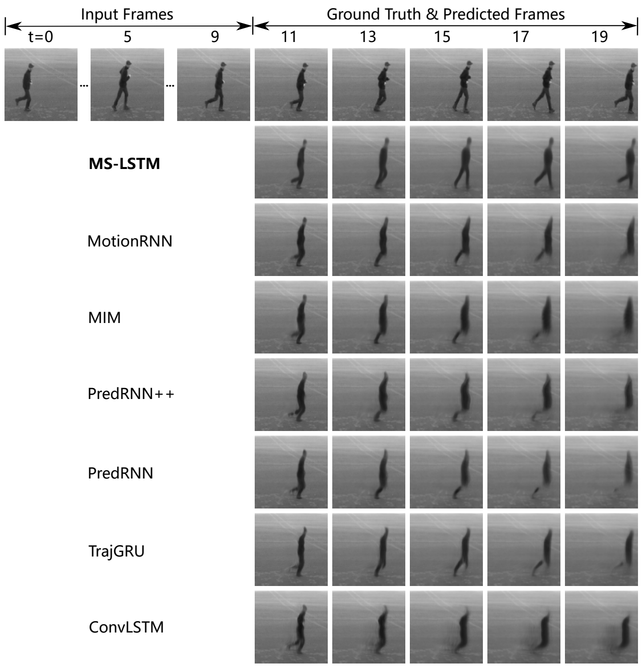
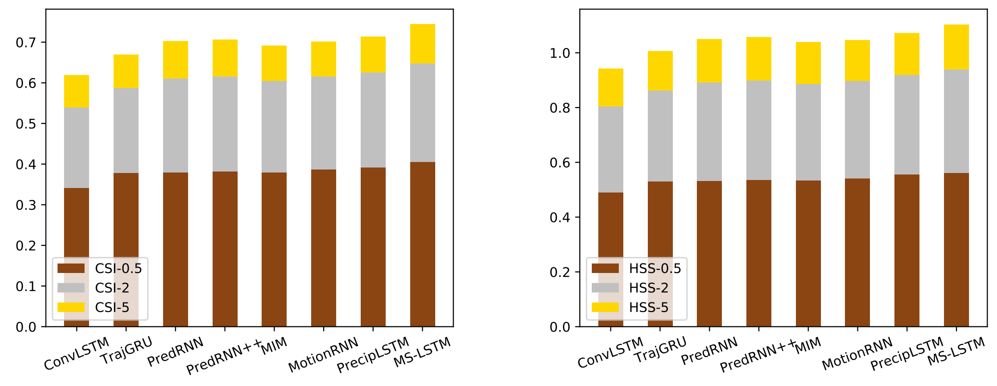


Fig. 9: Qualitative comparison on the KTH dataset.

5.4.1. *Comparison of the Training Cost and Performance*

Table 6 depicts the training cost comparison on the KTH dataset, demonstrating again that MS-LSTM requires less training cost. We use the quantitative metrics Structural Similarity (SSIM) [76] and Peak Signal to Noise Ratio (PSNR) to evaluate the quality of predicted frames. Fig. 8 shows the framewise comparisons of SSIM and PSNR, which are obtained by calculating the average of all test sequences at each time step. Apparently, MS-LSTM yields better results than other competing models. Fig. 9 also demonstrates this. TrajGRU, PredRNN, PredRNN++, MIM, and MotionRNN only focus on the motion of the left leg, and the right leg is lost in subsequent predictions. More-



(a) CSI (b) HSS

Fig. 10: Accumulate CSI and HSS comparison on the Germany radar dataset.

Table 7
Training cost comparison on the Germany radar dataset.

Model	Params	Memory	FLOPs	Time
ConvLSTM [58]	0.4M	6.8G	100.3G	0.8H
TrajGRU [59]	0.5M	14.0G	112.4G	7.7H
PredRNN [72]	0.9M	14.3G	203.5G	1.9H
PredRNN++ [70]	1.4M	21.1G	275.9G	2.3H
MIM [75]	1.8M	27.1G	416.7G	3.8H
MotionRNN [79]	1.9M	29.4G	426.1G	10.4H
PrecipLSTM [45]	1.8M	38.3G	398.5G	14.8H
MS-LSTM	1.9M	14.8G	260.6G	1.8H

Table 8
Quantitative comparison on the Germany radar dataset.

Model	CSI-0.5↑	CSI-2↑	CSI-5↑	HSS-0.5↑	HSS-2↑	HSS-5↑
ConvLSTM [58]	0.341	0.199	0.079	0.490	0.315	0.138
TrajGRU [59]	0.378	0.210	0.082	0.531	0.331	0.144
PredRNN [72]	0.380	0.231	0.092	0.533	0.358	0.158
PredRNN++ [70]	0.382	0.234	0.091	0.536	0.364	0.158
MIM [75]	0.380	0.225	0.087	0.534	0.352	0.153
MotionRNN [79]	0.387	0.229	0.085	0.541	0.357	0.149
PrecipLSTM [45]	0.392	0.234	0.088	0.556	0.364	0.152
MS-LSTM	**0.405**	**0.243**	**0.096**	**0.562**	**0.378**	**0.164**

over, their predictions are ambiguous, and we can't tell the head from the torso. In contrast, MS-LSTM produces the best predictions, which benefit from the application of multi-scale techniques. The large scales concern the movement of human body details such as arms and legs while the small scales care about the movement of the shape and contour of the human body.

5.5. Precipitation Nowcasting

The German radar dataset [2] is collected by 17 Doppler radars, including radar echo maps every 5 minutes from 2006 to 2017, covering the whole area of Germany. The original image size is 900×900, and we interpolate it to 180×180. We use data from 2006 to 2014 to optimize model parameters and data from 2015 to 2017 for testing. We sample one radar map every 30 minutes, which makes extrapolation much more difficult. The job of the model is to foretell 4 future frames based on 4 historical frames, that is, the variation of precipitation within 2 hours from now.

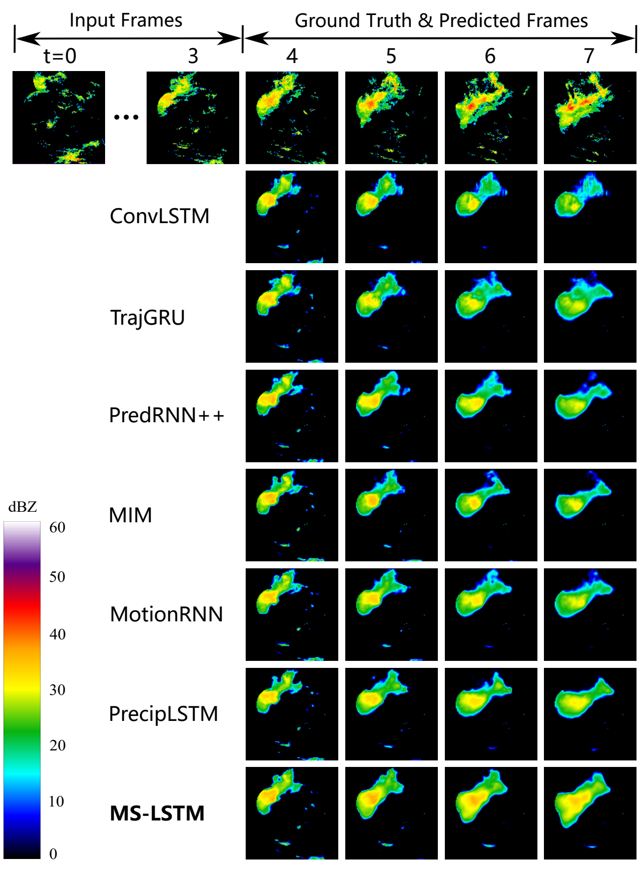


Fig. 11: Qualitative comparison on the Germany radar dataset.

5.5.1. *Comparison of the Training Cost and Performance*

We report comparisons of training cost and precipitation metrics between our model and competing models in

Fig. 10, Table 7, and Table 8, and the results all demonstrate one thing: MS-LSTM is optimal. It is worth noting that we only report the Critical Success Index (CSI) [59] and Heidke Skill Score (HSS) metrics [59] of models at the 0.5, 2, and 5 mm/h thresholds. This is mainly because Germany has a temperate maritime climate, where the precipitation is mostly light rain and light showers, with almost no heavy rain. Fig. 11 depicts one precipitation movement process that diffuses from northwestern to central Germany. As can be seen from the figure, each model predicts differently for the last frame (precipitation after 2 hours). Among them, the prediction of MS-LSTM is closest to the true value, which is thanks to the usage of the large convolution kernel to strengthen the multi-scale architecture to capture fast motion. In addition, although precipitation exhibits long-tail distribution and heavy rain is rare, MS-LSTM still focuses on heavier rain.

6. Conclusion

In this paper, given that the training of previous models is computationally intensive, we propose a lightweight model named MS-LSTM for video prediction, which wholly adopts multi-scale technologies to capture long-distance dependence and long-term trends. MS-LSTM mainly uses three multi-scale methods to construct spatiotemporal multi-scale representations, which are depth, downsampling, and multi-kernels. Experiments on four datasets have demonstrated that it has fewer training occupations but higher performance than the state-of-the-art model.

References

[1] Akan, A.K., Erdem, E., Erdem, A., Güney, F., 2021. Slamp: Stochastic latent appearance and motion prediction, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 14728–14737.

[2] Ayzel, G., Scheffer, T., Heistermann, M., 2020. Rainnet v1.0: a convolutional neural network for radar-based precipitation nowcasting. Geoscientific Model Development 13, 2631–2644.

[3] Castrejon, L., Ballas, N., Courville, A., 2019. Improved conditional vrnns for video prediction, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 7608–7617.

[4] Chai, Z., Xu, Z., Bail, Y., Lin, Z., Yuan, C., 2022. Cms-lstm: Context embedding and multi-scale spatiotemporal expression lstm for predictive learning, in: IEEE International Conference on Multimedia and Expo, IEEE. pp. 01–06.

[5] Chang, Z., Zhang, X., Wang, S., Ma, S., Gao, W., 2022. Strpm: A spatiotemporal residual predictive model for high-resolution video prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13946–13955.

[6] Chang, Z., Zhang, X., Wang, S., Ma, S., Ye, Y., Xinguang, X., Gao, W., 2021. Mau: A motion-aware unit for video prediction and beyond. Advances in Neural Information Processing Systems 34, 26950–26962.

[7] Chatterjee, M., Ahuja, N., Cherian, A., 2021. A hierarchical variational neural uncertainty model for stochastic video prediction, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9751–9761.

[8] Cho, S.J., Ji, S.W., Hong, J.P., Jung, S.W., Ko, S.J., 2021. Rethinking coarse-to-fine approach in single image deblurring, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4641–4650.

[9] Deng, S., Jia, S., Chen, J., 2019. Exploring spatial–temporal relations via deep convolutional neural networks for traffic flow prediction with incomplete data. Applied Soft Computing 78, 712–721.

[10] Denton, E., Fergus, R., 2018. Stochastic video generation with a learned prior, in: International Conference on Machine Learning, PMLR. pp. 1174–1183.

[11] Ding, X., Zhang, X., Han, J., Ding, G., 2022. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11963–11975.

[12] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al., 2020. An image is worth 16x16 words: Transformers for image recognition at scale, in: International Conference on Learning Representations.

[13] Espeholt, L., Agrawal, S., Sønderby, C., Kumar, M., Heek, J., Bromberg, C., Gazen, C., Carver, R., Andrychowicz, M., Hickey, J., et al., 2022. Deep learning for twelve hour precipitation forecasts. Nature Communications 13, 1–10.

[14] Esser, P., Rombach, R., Ommer, B., 2021. Taming transformers for high-resolution image synthesis, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12873–12883.

[15] Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C., 2021. Multiscale vision transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6824–6835.

[16] Fan, H., Zhu, L., Yang, Y., 2019. Cubic lstms for video prediction, in: Proceedings of the AAAI conference on Artificial Intelligence, pp. 8263–8270.

[17] Gao, S.H., Cheng, M.M., Zhao, K., Zhang, X.Y., Yang, M.H., Torr, P., 2019. Res2net: A new multi-scale backbone architecture. IEEE Transactions on Pattern Analysis and Machine Intelligence 43, 652–662.

[18] Gao, Y., Liu, Y., Zhang, H., Li, Z., Zhu, Y., Lin, H., Yang, M., 2020. Estimating gpu memory consumption of deep learning models, in: ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 1342–1352.

[19] Gao, Z., Shi, X., Wang, H., Zhu, Y., Wang, Y., Li, M., Yeung, D.Y., 2022. Earthformer: Exploring space-time transformers for earth system forecasting. Advances in Neural Information Processing Systems .

[20] Ge, S., Hayes, T., Yang, H., Yin, X., Pang, G., Jacobs, D., Huang, J.B., Parikh, D., 2022. Long video generation with time-agnostic vqgan and time-sensitive transformer. arXiv preprint arXiv:2204.03638 .

[21] Gu, J., Dong, C., 2021. Interpreting super-resolution networks with local attribution maps, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9199–9208.

[22] Guen, V.L., Thome, N., 2020. Disentangling physical dynamics from unknown factors for unsupervised video prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11474–11484.

[23] Guo, C., Fan, B., Zhang, Q., Xiang, S., Pan, C., 2020. Augfpn: Improving multi-scale feature learning for object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12595–12604.

[24] Gupta, A., Tian, S., Zhang, Y., Wu, J., Martín-Martín, R., Fei-Fei, L., 2022. Maskvit: Masked visual pre-training for video prediction. arXiv preprint arXiv:2206.11894 .

[25] Han, L., Liang, H., Chen, H., Zhang, W., Ge, Y., 2021. Convective precipitation nowcasting using u-net model. IEEE Transactions on Geoscience and Remote Sensing 60, 1–8.

[26] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

[27] Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation 9, 1735–1780.

[28] Huang, Q., Li, Z., Zheng, L., Yang, T., 2022a. Video frame prediction with dual-stream deep network emphasizing motions and content details. Applied Soft Computing 125, 109170.

[29] Huang, X., Li, X., Ye, Y., Feng, S., Luo, C., Zhang, B., 2022b. On understanding of spatiotemporal prediction model. IEEE Transactions on Circuits and Systems for Video Technology .

[30] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding, in: Proceedings of the ACM International Conference on Multimedia, pp. 675–678.

[31] Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization, in: International Conference on Learning Representations.

[32] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems 25, 1097—-1105.

[33] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

[34] Lee, A.X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., Levine, S., 2019. Stochastic adversarial video prediction, in: International Conference on Learning Representations.

[35] Lee, S., Kim, H.G., Choi, D.H., Kim, H.I., Ro, Y.M., 2021. Video prediction recalling long-term motion context via memory alignment learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3054–3063.

[36] Li, C., Chen, X., 2023. Future video frame prediction based on generative motion-assistant discriminative network. Applied Soft Computing , 110028.

[37] Li, X., Wang, W., Hu, X., Yang, J., 2019. Selective kernel networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 510–519.

[38] Lin, Z., Li, M., Zheng, Z., Cheng, Y., Yuan, C., 2020. Self-attention convlstm for spatiotemporal prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 11531–11538.

[39] Lin, Z., Yuan, C., Li, M., 2021. Haf-svg: hierarchical stochastic video generation with aligned features, in: Proceedings of the International Conference on International Joint Conferences on Artificial Intelligence, pp. 991–997.

[40] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012–10022.

[41] Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., Hu, H., 2022. Video swin transformer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3202–3211.

[42] Luo, W., Li, Y., Urtasun, R., Zemel, R., 2016. Understanding the effective receptive field in deep convolutional neural networks. Advances in Neural Information Processing Systems 29.

[43] Ma, Z., Zhang, H., Liu, J., 2022a. Focal frame loss: A simple but effective loss for precipitation nowcasting. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 15, 6781–6788.

[44] Ma, Z., Zhang, H., Liu, J., 2022b. Ms-rnn: A flexible multi-scale framework for spatiotemporal predictive learning. arXiv preprint arXiv:2206.03010 .

[45] Ma, Z., Zhang, H., Liu, J., 2022c. Preciplstm: A meteorological spatiotemporal lstm for precipitation nowcasting. IEEE Transactions on Geoscience and Remote Sensing 60, 1–8.

[46] Ma, Z., Zhang, H., Liu, J., 2023. Mm-rnn: A multimodal rnn for precipitation nowcasting. IEEE Transactions on Geoscience and Remote Sensing 61, 1–14.

[47] Mathieu, M., Couprie, C., LeCun, Y., 2016. Deep multi-scale video prediction beyond mean square error, in: International Conference on Learning Representations.

[48] Oprea, S., Martinez-Gonzalez, P., Garcia-Garcia, A., Castro-Vargas, J.A., Orts-Escolano, S., Garcia-Rodriguez, J., Argyros, A., 2020. A review on deep learning techniques for video prediction. IEEE Transactions on Pattern Analysis and Machine Intelligence .

[49] Pan, X., Lu, Y., Zhao, K., Huang, H., Wang, M., Chen, H., 2021. Improving nowcasting of convective development by incorporating polarimetric radar variables into a deep-learning model. Geophysical Research Letters 48, e2021GL095302.

[50] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., 2019. Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems 32, 8026–8037.

[51] Pfeuffer, A., Dietmayer, K., 2019. Separable convolutional lstms for faster video segmentation, in: IEEE Intelligent Transportation Systems Conference, IEEE. pp. 1072–1078.

[52] Qi, T., Chen, L., Li, G., Li, Y., Wang, C., 2023. Fedagcn: A traffic flow prediction framework based on federated learning and asynchronous graph convolutional network. Applied Soft Computing , 110175.

[53] Ravuri, S., Lenc, K., Willson, M., Kangin, D., Lam, R., Mirowski, P., Fitzsimons, M., Athanassiadou, M., Kashem, S., Madge, S., et al., 2021. Skilful precipitation nowcasting using deep generative models of radar. Nature 597, 672–677.

[54] Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems 28.

[55] Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Aomputer-assisted Intervention, Springer. pp. 234–241.

[56] Schuldt, C., Laptev, I., Caputo, B., 2004. Recognizing human actions: a local svm approach, in: Proceedings of the International Conference on Pattern Recognition, IEEE. pp. 32–36.

[57] Shen, X., Meng, K., Zhang, L., Zuo, X., 2022. A method of radar echo extrapolation based on dilated convolution and attention convolution. Scientific Reports 12, 1–12.

[58] Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.c., 2015. Convolutional lstm network: A machine learning approach for precipitation nowcasting. Advances in Neural Information Processing Systems 28.

[59] Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.Y., Wong, W.k., Woo, W.c., 2017. Deep learning for precipitation nowcasting: A benchmark and a new model. Advances in Neural Information Processing Systems 30.

[60] Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations.

[61] Sohoni, N.S., Aberger, C.R., Leszczynski, M., Zhang, J., Ré, C., 2019. Low-memory neural network training: A technical report. arXiv preprint arXiv:1904.10631 .

[62] Srivastava, N., Mansimov, E., Salakhudinov, R., 2015. Unsupervised learning of video representations using lstms, in: International Conference on Machine Learning, PMLR. pp. 843–852.

[63] Sun, J., Xie, J., Hu, J.F., Lin, Z., Lai, J., Zeng, W., Zheng, W.s., 2019. Predicting future instance segmentation with contextual pyramid convlstms, in: Proceedings of the ACM International Conference on Multimedia, pp. 2043–2051.

[64] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9.

[65] Tulyakov, S., Liu, M.Y., Yang, X., Kautz, J., 2018. Mocogan: Decomposing motion and content for video generation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1526–1535.

[66] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Advances in Neural Information Processing Systems 30.

[67] Villegas, R., Yang, J., Hong, S., Lin, X., Lee, H., 2017. Decomposing motion and content for natural video sequence prediction, in: International Conference on Learning Representations.

[68] Wang, H., Cao, P., Wang, J., Zaiane, O.R., 2022a. Uctransnet: re-

thinking the skip connections in u-net from a channel-wise perspective with transformer, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 2441–2449.

[69] Wang, Y., Bilinski, P., Bremond, F., Dantcheva, A., 2020a. G3an: Disentangling appearance and motion for video generation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5264–5273.

[70] Wang, Y., Gao, Z., Long, M., Wang, J., Philip, S.Y., 2018a. Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning, in: International Conference on Machine Learning, PMLR. pp. 5123–5132.

[71] Wang, Y., Jiang, L., Yang, M.H., Li, L.J., Long, M., Fei-Fei, L., 2018b. Eidetic 3d lstm: A model for video prediction and beyond, in: International Conference on Learning Representations.

[72] Wang, Y., Long, M., Wang, J., Gao, Z., Yu, P.S., 2017. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. Advances in Neural Information Processing Systems 30.

[73] Wang, Y., Wu, H., Zhang, J., Gao, Z., Wang, J., Yu, P., Long, M., 2022b. Predrnn: A recurrent neural network for spatiotemporal predictive learning. IEEE Transactions on Pattern Analysis and Machine Intelligence .

[74] Wang, Y., Wu, J., Long, M., Tenenbaum, J.B., 2020b. Probabilistic video prediction from noisy data with a posterior confidence, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10830–10839.

[75] Wang, Y., Zhang, J., Zhu, H., Long, M., Wang, J., Yu, P.S., 2019. Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9154–9162.

[76] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P., 2004. Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing 13, 600–612.

[77] Wei, C.C., 2013. Soft computing techniques in ensemble precipitation nowcast. Applied Soft Computing 13, 793–805.

[78] Wu, B., Nair, S., Martin-Martin, R., Fei-Fei, L., Finn, C., 2021a. Greedy hierarchical variational autoencoders for large-scale video prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2318–2328.

[79] Wu, H., Yao, Z., Wang, J., Long, M., 2021b. Motionrnn: A flexible model for video prediction with spacetime-varying motions, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15435–15444.

[80] Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K., 2017. Aggregated residual transformations for deep neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1492–1500.

[81] Xu, J., Ni, B., Yang, X., 2018. Video prediction via selective sampling. Advances in Neural Information Processing Systems 31.

[82] Yan, W., Zhang, Y., Abbeel, P., Srinivas, A., 2021. Videogpt: Video generation using vq-vae and transformers. arXiv preprint arXiv:2104.10157 .

[83] Yang, X., Zhang, F., Sun, P., Li, X., Du, Z., Liu, R., 2022. A spatiotemporal graph-guided convolutional lstm for tropical cyclones precipitation nowcasting. Applied Soft Computing 124, 109003.

[84] Yao, Z., Wang, Y., Long, M., Wang, J., 2020. Unsupervised transfer learning for spatiotemporal predictive networks, in: International Conference on Machine Learning, PMLR. pp. 10778–10788.

[85] Zhang, J., Zheng, Y., Qi, D., 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence.

[86] Zhu, C., Dong, H., Pan, J., Liang, B., Huang, Y., Fu, L., Wang, F., 2022. Deep recurrent neural network with multi-scale bi-directional propagation for video deblurring, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 3598–3607.