

Santa Clara University

Scholar Commons

Engineering Ph.D. Theses

Student Scholarship

6-2024

Deep Learning-Based Video Prediction

Mareeta Mathai

Follow this and additional works at: https://scholarcommons.scu.edu/eng_phd_theses



Part of the [Computer Engineering Commons](#)

Santa Clara University

Department of Computer Science and Engineering

Date: April 18, 2024

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Mareeta Mathai

ENTITLED

Deep Learning-Based Video Prediction

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING

N. Ling

N. Ling (Jun 7, 2024 15:15 PDT)

Thesis Advisor
Prof. Dr. Nam Ling

Ying Liu

Ying Liu (Jun 7, 2024 15:16 PDT)

Thesis Co-Advisor
Prof. Dr. Ying Liu

Silvia Figueira

Department Chair
Prof. Dr. Silvia Figueira

T Ogunfunmi

Thesis Reader
Prof. Dr. Tokunbo Ogunfunmi

yuhong Liu

Thesis Reader
Prof. Dr. Yuhong Liu

Sharon Hsiao

Sharon Hsiao (Jun 10, 2024 20:51 PDT)

Thesis Reader
Prof. Dr. Sharon Hsiao

Deep Learning-Based Video Prediction

By

Mareeta Mathai

Dissertation

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Science and Engineering
in the School of Engineering at
Santa Clara University, 2024

Santa Clara, California

Dedicated to my loving family

Acknowledgements

Foremost, I am grateful to the Almighty God for strengthening and guiding me in my academic journey.

I express my heartfelt gratitude and thanks towards my esteemed supervisor Dr. Nam Ling, whose unwavering support, encouragement and mentorship have guided and motivated me throughout this research journey. It has indeed been a privilege to have the opportunity to do research under Dr. Ling. I am particularly grateful for his assistance in securing the necessary funding opportunities.

I extend my thanks to Dr. Ying Liu, my co-advisor whose guidance and advise have been instrumental in research paper writing and experimentation. Her patience, preciseness and vast knowledge have greatly contributed to my research works.

I am grateful to my thesis committee members Dr. Tokunbo Ogunfunmi, Dr. Yuhong Liu and Dr. Sharon Hsiao for their valuable suggestions and assistance.

My gratitude extends to the School of Engineering for offering Graduate Fellowship funding opportunity and Department of Computer Science and Engineering for providing Teaching Assistantship opportunity.

I would like to thank my friends and colleagues for all the good time and kind co-operation provided. I am also grateful to the Computer Science and Engineering department office staff for extending the necessary assistance required throughout my tenure.

Last but not the least; let me whole heartedly thank my entire family: my parents, husband, kids and siblings for their immense encouragement, love and support. Without their blessings and understanding, I could not have done with the research.

Deep Learning-Based Video Prediction

Mareeta Mathai

Department of Computer Science and Engineering

Santa Clara University

Santa Clara, California

2024

ABSTRACT

The task of video prediction is to generate unseen future video frames based on the past ones. It is an emerging, yet challenging task due to its inherent uncertainty and complex spatiotemporal dynamics. The ability to predict and anticipate future events from video prediction has applications in various prediction systems like self-driving cars, weather forecasting, traffic flow prediction, video compression etc. Due to the success of deep learning in the computer vision field, several deep learning Artificial Intelligence (AI) architectures such as convolutional neural networks (CNNs), long short-term memory (LSTMs), convolutional LSTMs (ConvLSTMs) and transformers have been explored to improve prediction accuracy. The internal representation, mainly the spatial correlations and temporal dynamics of the video, is learned and used to predict the next frames in deep learning-based video prediction. Several state-of-the-art deep learning methods have achieved superior video prediction accuracy at the expense of huge computational cost. In the light of recent wide popularity of Green AI which aims for efficient environment friendly solutions alongside accuracy, this research concentrates on efficient methods for

video prediction. Such methods are suitable for memory-constrained and computation resource-limited platforms, such as mobile and embedded devices. We focus on CNN/LSTM methods and transformer-based architectures with fewer parameters for our lightweight efficient environment-friendly video prediction techniques. We conducted experimental studies on popular video prediction datasets and compared to existing methods, our proposed methods achieved competitive frame prediction accuracy with significantly reduced model size, trainable parameters, and computational complexity.

TABLE OF CONTENTS

DEDICATION.....	iii
ACKNOWLEDGEMENTS.....	iv
ABSTRACT.....	v
TABLE OF CONTENTS.....	vii
LIST OF TABLES.....	x
LIST OF FIGURES.....	xi
CHAPTER 1.....	13
Introduction.....	13
1.1 Overview of Video Prediction.....	13
1.1.1 Problem Statement.....	14
1.2 Motivation of the Research.....	15
1.3 Research Questions.....	16
1.3 Outline of the Dissertation.....	17
CHAPTER 2.....	18
Review of the Existing Methods.....	18
2.1 CNN-based Methods.....	18
2.2 RNN and ConvLSTM-based Methods.....	19
2.3 GAN-based Methods.....	20
2.4 Transformer-based Methods.....	21
2.5 Challenges with Existing Methods.....	22
CHAPTER 3.....	24
Methodology to Improve Network Efficiency.....	24
3.1 2D Convolution vs 2D Separable Convolutions.....	24
3.2 3D Convolutions vs 3D Separable Convolutions.....	25

3.3	Training Loss Function and Evaluation Metrics.....	27
CHAPTER 4.....		29
Proposed Method Using 3D Separable CNN and LSTM		29
4.1	Introduction.....	29
4.2	Proposed Method.....	29
4.2.1	Reversible Networks.....	30
4.2.2	Proposed Prediction Network.....	30
4.2.2.1	AE block with 3D Separable convolutions.....	32
4.2.2.2	ST-LSTM block with 3D Separable Convolutions...	33
4.3	Training and Evaluation Setup.....	34
4.4	Experiments and Analysis.....	35
4.4.1	Moving MNIST Dataset.....	35
4.4.2	KTH Action Dataset.....	38
4.4.3	BAIR Dataset.....	40
4.5	Conclusion.....	41
CHAPTER 5.....		42
Proposed Method using Hybrid Transformer-LSTM Network.....		42
5.1	Introduction.....	42
5.2	Preliminaries.....	44
5.3	Proposed Method.....	45
5.3.1	Algorithm Overview.....	45
5.3.2	Spatial Embedding and Positional Encoding.....	47
5.3.3	3D Transformer-LSTM.....	48
5.4	Training and Inference Strategy.....	56
5.5	Datasets.....	57
5.6	Experimental Settings and Methods in Comparison.....	60
5.7	Experimental Results and Analysis.....	61
5.7.1	MMNIST	61
5.7.2	KTH Action	62

5.7.3	TaxiBJ	65
5.7.4	Human 3.6m	66
5.7.5	KITTI and Caltech Pedestrian	69
5.8	Ablation Study.....	70
5.9	Conclusion.....	72
CHAPTER 6.....		73
Conclusion.....		73
6.1	Summary of the Major Contributions.....	73
6.2	Future Research.....	75
BIBLIOGRAPHY.....		78
PUBLICATIONS.....		87

LIST OF TABLES

4.1	Configuration of Proposed 3D separable ST-LSTM-based network.....	34
4.2	Quantitative results on Moving MNIST-2 and Moving MNIST-3 dataset.....	36
4.3	Quantitative results on KTH Action dataset.....	39
4.4	Quantitative results on BAIR dataset.....	41
5.1	Configuration of the proposed hybrid transformer-LSTM network.....	47
5.2	Quantitative results on MMNIST-3K and KTH Action datasets.	63
5.3	Quantitative results on TaxiBJ dataset.....	66
5.4	Quantitative results on Human 3.6m dataset.....	68
5.5	Quantitative results on Caltech Pedestrian dataset after training on KITTI dataset.....	69
5.6	Ablation study on MMNIST-3K.....	71

LIST OF FIGURES

1.1	Figure 1: The future frames (t_3, t_4, t_5, t_6) are generated from the existing frames(t_0, t_1, t_2).....	13
1.2	Pedestrian crossing the road and the autonomous car predicts the position of pedestrian to apply braking.	14
3.1	(a) The standard 3D convolution, and (b) the 3D depthwise separable convolution.....	25
4.1	Proposed prediction architecture.....	31
4.2	The AE block.....	32
4.3	Separable RPM.....	33
4.4	Qualitative results on Moving MNIST-2 dataset and Moving MNIST-3 dataset	37
4.5	Qualitative results on KTH action dataset.....	39
4.6	Qualitative results on the BAIR dataset.	40
5.1	Transformer architecture.....	44
5.2	The overall architecture of the proposed network for three time steps $t - 1, t$, and $t + 1$	46
5.3	The stack of six 3DTransLSTM blocks for time steps $t - 1, t, t + 1$	48
5.4	(a) Branch 1 of the 3DTransLSTM block, (b) $3DSepFFN$ module and $3DSepSA$ module	50
5.5	The structure of the l -th 3D SepST-LSTM layer at time step t	53
5.6	The visual results on the MMNIST-3K dataset for models trained on the MMNIST-10K dataset.	62

5.7	The visual results on the KTH Action dataset	64
5.8	The visual results on the TaxiBJ dataset	67
5.9	The visual results on the Human 3.6m dataset	68
5.10	The visual results on the Caltech Pedestrian dataset	70

CHAPTER 1

Introduction

1.1 Overview of Video Prediction

With the increasing advent of powerful graphics processing units (GPUs), deep learning is the foremost option of many artificial intelligence (AI) applications, and it has been a crucial part in the advancement of several computer vision (CV) algorithms. In this work, we focus on the task of video frame prediction utilizing efficient deep learning architectures. A video frame prediction model generates future frames from the past existing frames, by learning the complex spatiotemporal content and dynamics of the video data. It aims to predict both the content and motion in the subsequent frames in a video, by analyzing and extracting spatial information and temporal content in the previous sequence of frames. Figure 1. illustrates an example of a few frames from a video of a swinging woman from Davis dataset [1]. Leveraging the first three frames (t_0, t_1, t_2), a video prediction model can accurately generate the subsequent frames (t_3, t_4, t_5, t_6), capturing the woman's upward motion.



Fig. 1.1: The future frames (t_3, t_4, t_5, t_6) are generated from the existing frames(t_0, t_1, t_2)

The task of video prediction finds significance in various real-world applications such as video coding [2], [3], autonomous vehicles [4,5], traffic flow prediction [6], weather forecasting [7] and anomaly detection [8]. The anticipation of a future event and real-time decision making are the keys of an intelligent decision-making system[5]. Fig. 1.2 [5] shows an example of a real-world scenario of an intelligent decision-making system in autonomous driving. In the figure, the autonomous car observes a pedestrian intending to cross the road (context frames). The car anticipates and predicts the future position and movements of the pedestrian (predicted frames), deciding whether to apply braking for safely accommodating pedestrian movements and thus, avoid collision.

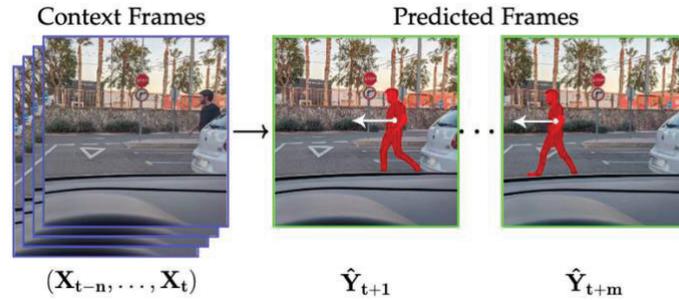


Fig. 1.2: Pedestrian crossing the road and the autonomous car predicts the position of pedestrian to apply braking. Copyright [5]

1.1.1 Problem Statement

The task of video prediction can be formulated as follows. Consider the video prediction process at time slot t . The network takes a 4D tensor $\mathbf{X}_t \in \mathbb{R}^{C \times L \times H \times W}$ as input, which represents L successive video frames with frame indices $\{t, t + 1, \dots, t + L - 1\}$, where each frame has C channels, height H and width W . The problem of video prediction is to

predict the L frames $\hat{\mathbf{X}}_{t+1} \in \mathbb{R}^{C \times L \times H \times W}$ with frame indices $\{t + 1, t + 2 \dots t + L\}$ given \mathbf{X}_t and can be formulated as follows:

$$\hat{\mathbf{X}}_{t+1} = \text{Net}\{\mathbf{X}_t | \theta\} \quad (1.1)$$

where θ is the collection of trainable model parameters. $\text{Net}(\cdot)$ represents the frame prediction network.

1.2 Motivation of the Research

Many existing video prediction models are based on 2- dimensional convolutional neural networks (2D CNNs) [4]– [10], or recurrent neural networks (RNNs) [13]–[15]. Studies show the potential of 3D CNNs in learning spatio-temporal dynamics better than 2-D CNNs and RNNs and it has been used in various video representation learning tasks [14,19]. Additionally, transformers have also emerged successful in various video processing applications. However, few research has been made to find efficient models for predicting future frames, where efficiency encompasses both accuracy and computation complexity. Recently, researchers across the NLP and CV communities [43], [44] advocated to shift the focus to Green Artificial Intelligence (AI) [66] with energy- efficient deep learning solutions, rather than continuously pushing red AI methods to reach state-of-the-art (SOTA) results using massive computational power. Our work focuses on developing an efficient video frame prediction model with reduced model size, fewer parameters, and low computational complexity, while achieving competitive prediction accuracy.

1.3 Research Questions

In our research, we aim to address a spectrum of questions vital to the field of video prediction. Our research questions include:

- **Efficiency and energy reduction:** How can we attain efficiency and reduce energy consumption of deep learning models while preserving accuracy? We seek to explore neural network methodologies which can reduce the memory consumption and model complexity, paving the way for sustainable AI solutions.
- **Architectural combinations:** How can we devise a methodology by combining various deep learning architectures (2D/3D CNNs, LSTMs, transformers) to accurately learn the spatio-temporal relations among video frames? By examining hybrid combinations and integrating other deep learning models, we aim to devise optimal frameworks to predict video frames accurately.
- **Trade-offs between efficiency and accuracy:** What trade-offs exist between model complexity, computational efficiency and prediction accuracy, in the context of Green AI? We closely examine the balance required between these factors to achieve resource-efficient solutions.
- **Real-world applications:** What is the real-world application of efficient video prediction models? Furthermore, we conduct experiments to validate the

generalization ability of our model and also discuss the scenarios suitable for lightweight video prediction models.

1.4 Outline of the Dissertation

The dissertation is organized as follows:

Chapter 2 introduces existing methods for video prediction methods and the problems associated with these.

In Chapter 3, we discuss the principles of 2D and 3D depthwise separable convolutions which plays an important role in our research and the rationale behind its usage. We also introduce the training loss function evaluation metrics used to compare the state-of-the-art (SOTA) methods.

Chapter 4 presents a lightweight method incorporating 3D separable convolution-based LSTM for predicting future frames. It drastically improves the computational complexity of the prediction process by maintaining competitive accuracy.

In Chapter 5, we introduce another approach for video prediction with separable convolutions. A hybrid transformer-LSTM method is proposed to extract long-range dependencies in video, along with parallelizing the computation with self-attention mechanism.

Chapter 6 concludes the dissertation and discusses future research directions.

CHAPTER 2

Review of Existing Video Prediction Methods

Existing video prediction methods can be broadly classified into four categories: CNN-based methods, RNN-based methods, generative adversarial network (GAN)-based methods, and transformer-based methods.

2.1 CNN-based Methods

CNNs efficiently capture the spatial structure of the images (video frames), but lack the ability to model time dimension efficiently in video processing. Many 2D CNN-based video prediction approaches [9], [10], [11], [12] were devised to model spatiotemporal dynamics in video data. In [9], the content encoder and motion encoder focused on the static scene information and the temporal dynamics of consecutive frames, respectively. The deep multi-branch mask network (DMMNet) [10] proposed a future appearance synthesizer to synthesize RGB pixels of the future frame, and a future flow synthesizer to generate optical flows. In [11], a convolutional encoder-decoder network was proposed to explicitly incorporate a time-related input variable to model temporal correlations. Deformable convolutions were used to fuse features from previous frames in [12].

Certain 2D CNN-based frame prediction schemes were proposed for inter-frame prediction [2],[13],[14] in traditional video coding such as the high efficiency video coding (HEVC) [15] and versatile video coding (VVC) [16], or in learning-based video coding. For example, a 2D CNN-based deep network was proposed for both uni-directional and bi-directional frame prediction in HEVC and avoided coding additional motion information

[2]. Another CNN-based multi-resolution video prediction network (VPN) [13] utilized two sub-VPN architectures in cascade to generate virtual reference frame from previously coded frames in HEVC. Further, recurrent and bi-directional in-loop prediction modules were proposed in [14] as part of a deep learning-based video compression system.

3D CNN is another way to extract spatiotemporal features. It was used along with optical flow images to predict future frames based on a single image in [17]. Spatially displaced convolution network (SDC-Net) [18] utilized a 3D CNN for video prediction, conditioning on both past frames and past optical flows.

2.2 RNN and ConvLSTM-based Methods

Using CNNs alone can only consider local structures or short-range dependencies in video data due to the limited size of convolution kernels. To effectively capture long-range correlations in a video sequence, methods based on RNNs [19], [20] were proposed to predict future frames. For example, an LSTM-based encoder-decoder network was developed in [19]. It used an encoder LSTM to map an input video sequence into a fixed length representation, which was then decoded using single or multiple decoder LSTMs to predict future frames. Folded recurrent neural network (FRNN) [20] presented a recurrent auto-encoder with state sharing between the encoder and the decoder. It utilized stacked double-mapping gated recurrent unit (GRU) layers to enable bidirectional information flow between the input and the output.

Although RNNs effectively learn sequential representations, they fail to accurately learn spatial content [21]. To address this issue, convolutions were incorporated into LSTMs to

form ConvLSTMs [21], where the internal fully connections in LSTM were replaced by convolution operations. For example, ConvLSTM was utilized in [9] for motion prediction and was used in [10] to generate appearance features of the next frame from two previous frames. In addition, the stacked ConvLSTM architecture was explored in the dynamic neural advection (DNA) module [22], which predicted the distribution of each pixel in the current frame based on the previous frame. E3D-LSTM [23] integrated 3D convolutions into LSTM to capture short-term frame dependencies and utilized a gate-controlled self-attention module to perceive long-term correlations. The PredRNN [24] network proposed the popular spatiotemporal LSTM (ST-LSTM) structure. It adopted a temporal memory cell and a novel spatiotemporal memory cell to simultaneously memorize spatial and temporal information. Later, several video prediction methods adopted ST-LSTM as their building blocks [25]–[28]. For example, the memory-in-memory (MIM) [25] model improved PredRNN [24] by replacing the simple forget gate in the ST-LSTM block with two cascaded memory transitions, which more effectively captured non-stationary dynamics. CrevNet [26] used a reversible auto-encoder and stacked ST-LSTM blocks for future frame prediction and object detection. PredRNN-V2 [27] improved PredRNN [24] by introducing a memory decoupling loss to ST-LSTM to keep the memory cells from learning redundant features. Other ConvLSTM or convolutional GRU (ConvGRU) approaches such as TrajGRU [29], PredRNN++ [30], Conv-TTLSTM [31], STGRU [32] and ASTM [33] were also developed for the video prediction task.

2.3 GAN-based Methods

Due to the mean-squared error (MSE) loss adopted in model training, CNN-based video

prediction models tend to generate blurry predicted frames which are inconsistent with human perception. To overcome this limitation, GAN-based models adopt adversarial training such that the predicted frames are sharper and present more details than pure CNN-based methods. In GAN-based methods, the generator produces future frames, and the discriminator tries to distinguish the generated frames and the ground truth as real or fake. BeyondMSE [34] was the pioneer in applying adversarial training for video prediction. It used a multi-scale architecture and an image gradient difference loss along with the MSE loss. Dual-Motion GAN (DM-GAN) [35] used a dual adversarial training mechanism with two pairs of generator and discriminator to generate future frames and future flows simultaneously. CycleGAN [36] adopted a forward-backward prediction scheme by training a generator to produce both future frames and past frames. Attention-based inter-frame prediction method in [37] enhanced coding efficiency of VVC by incorporating GAN-based deep attention map estimation and deep frame interpolation methods.

2.4 Transformer-based Methods

In recent years, transformers have been developed for NLP and CV tasks. Compared with RNN-based methods, the transformer architecture can extract long-term dependencies more efficiently and get rid of the limitation of seriality. In particular, a few approaches combined transformers and CNNs for video frame prediction. For instance, ConvTransformer [40] used an end-to-end encoder-decoder transformer architecture for video interpolation and extrapolation tasks. It proposed multi-head convolutional self-attention layers with 2D convolutions in both the encoder and decoder. The temporal convolutional transformer network (TCTN) [41] used a transformer-based encoder for

video prediction, where 3D convolutional layers were employed to extract short-term dependencies and masked self-attention layers were used to capture long-term dependencies. The video prediction transformer (VPTR) [42] proposed to separately perform spatial attention and temporal attention. First, spatial attention was performed locally on each feature patch using multi-head self-attention (MHSA), followed by a 2D separable convolution-based feed-forward neural network. Afterwards, a temporal MHSA was adopted to model the temporal dependency between frames.

2.5 Challenges with Existing Methods

Although the aforementioned SOTA methods achieved accurate video prediction results, their accuracy comes at a price of big model size, large amount of model parameters and heavy computational complexity. For example, the transformer-based models TCTN [41] and VPTR [42] have large model size and FLOPs due to standard 3D convolutions and complicated attention mechanisms, respectively. The ConvLSTM-based models E3D-LSTM [23] and CrevNet [26] adopted standard 3D convolutions too. MIM [25] also has relatively larger model size and FLOPs since it adopted additional memory modules inside the original ST-LSTM blocks.

For memory-constrained and computation resource-limited platforms, such as mobile and embedded devices, it's difficult to deploy the huge models. Thus, we aim to produce efficient models with low complexity and competitive accuracy.

Some of the research focusses on next-frame prediction, which predicts only the next frame which has seen profound advancements in the recent times. However, the long-term

predictions remain a challenge due to the modeling of interactions over extended time horizons. Additionally, while maintaining visual quality, several models lack the ability to accurately predict object movements in video frames.

Furthermore, most research works do not demonstrate their success in generalization, i.e. testing on an unseen dataset different from the training dataset. Generalization is important in our task since the training datasets may not wholly represent the diversity of real-world scenarios.

In our research, we aim at developing a lightweight video prediction network which still offers competitive frame prediction accuracy. Besides, while existing video prediction networks adopt 2D convolutions [9]–[14], [21], [24], [25], [27], [30] or standard 3D convolutions [17], [18], [23], [26] our proposed methods adopt the idea of 3D separable convolution. This not only leverages spatiotemporal correlations, but also effectively reduces model size, trainable parameters, and computational cost. We also demonstrate that our method possesses generalization ability to test on unseen videos.

CHAPTER 3

Methodology to Improve Network Efficiency

Separable convolution was first conceived in MobileNet [47] to develop lightweight models suitable for mobile and embedded devices. Later, 2D separable CNN was utilized for faster video segmentation [48], moving object detection [49], and violence detection [50]. To alleviate the computation burden of standard 3D convolution in deep networks, 3D separable CNN was proposed for dynamic hand gesture recognition and video moving object segmentation [51],[52].

In this section, we explain the 2D, 2D separable, 3D and 3D separable convolutions and the rationale behind the usage of depthwise separable convolutions in our research.

3.1 2D Convolutions vs 2D Separable Convolutions

2D convolutions are computationally simpler than 3D convolutions as it takes only spatial dimension into consideration. Many video processing tasks explored 2D convolutions in the past. In this section we briefly go through the 2D convolutions and its decomposition to 2D separable convolutions. Our proposed model utilizes 2D depthwise separable convolutions to learn the spatial dependencies in image patches (described in Section 4.3.2).

For an RGB image of size $C \times H \times W$, where C is the number of channels, H is the height and W is the width of the image, 2D convolutions move in horizontal direction x and vertical direction y across the image. N number of 3D filters of size $C \times K \times K$ convolves in height and width dimension to produce feature maps of size $N \times \hat{H} \times \hat{W}$.

To alleviate its computational complexity, 2D convolutions are separated to 2D depthwise and 1D pointwise convolutions. In 2D depthwise convolution, filters of size $1 \times K \times K$ convolves with each input channel to produce an intermediate output $C \times \check{H} \times \check{W}$. As the following step, we apply 1D pointwise convolutions with N filters of original input channel dimension, $N \times \check{H} \times \check{W}$.

3.2 3D Convolutions vs 3D Separable Convolutions

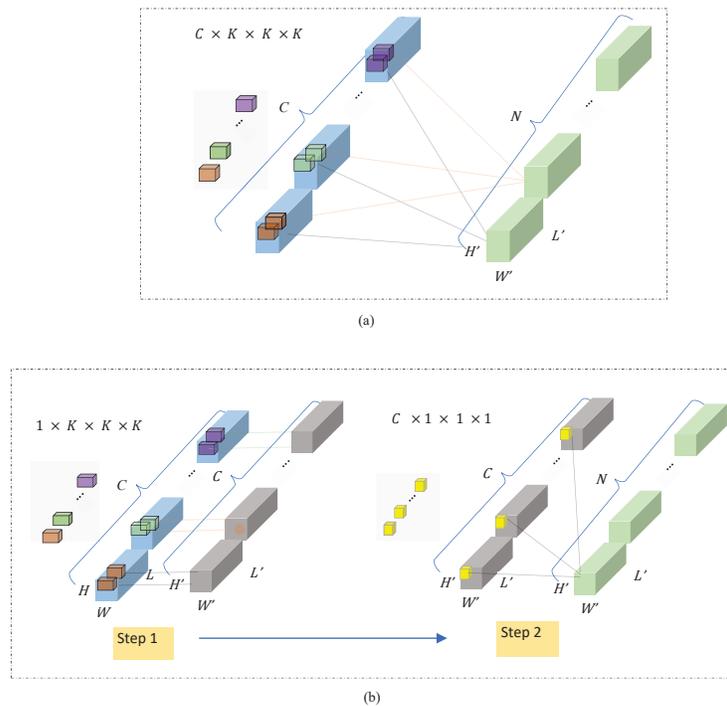


Fig. 3.1: (a) The standard 3D convolution, and (b) the 3D depthwise separable convolution.

Video prediction needs a model to learn spatiotemporal information abundantly and deeply. To achieve this goal, prior arts adopted standard 3D convolutions [23], [26], [41], since it is better to learn the temporal dependencies among the input frames than the 2D convolutions. It takes C channels of 3D input $L \times H \times W$, where C is the number of input

channels, L is the temporal length i.e the number of frames in time dimension, H is the height and W denotes the width. Let \mathbf{X}_{in} be the 4D input to standard 3D convolution with size $C \times L \times H \times W$. 4D filters F of size $C \times K \times K \times K$ (channel \times time \times height \times width) move in three directions (time, height, width) to generate a 3D output tensor \mathbf{X}_{out} of size $L' \times H' \times W'$, where L , H' and W' are the length, height, and width of the output tensor, respectively. Such N filters would create the final output tensor \mathbf{X}_{out} of size $N \times L' \times H' \times W'$. Mathematically, the output tensor, \mathbf{X}_{out} can be formulated as below:

$$\mathbf{X}_{out}[l, h, w] = \sum_{c=0}^{C_i-1} \sum_{k=0}^{K-1} \sum_{j=0}^{K-1} \sum_{i=0}^{K-1} F[c, k, j, i] \times \mathbf{X}_{in}[c, l-k, h-j, w-i] \quad (3.1)$$

where (k, j, i) and (l, h, w) are the dimensions of the filter F and input \mathbf{X}_{in} respectively. The illustration of 3D convolution is given in Fig. 3.1 (a). The number of computations involved in the traditional 3D convolution is $C \times K \times K \times K \times N \times L \times H' \times W'$.

Video prediction is a low-level vision task, which needs a model to learn spatiotemporal information abundantly and deeply. Though 3D convolutions are amply used in many existing approaches [7], it has made their architectures much complex and model sizes bigger. To reduce computational complexity, 3D standard convolution is separated to 3D depth-wise convolution and pointwise convolution, which is combined to term as 3D depthwise separable convolution.

As shown in Fig. 3.1 (b) Step 1, depthwise convolution applies filters of size $1 \times K \times K \times K$ to each of the C input channels to produce an intermediate output of size $C \times L' \times H' \times W'$. This process has $C \times K \times K \times K \times L \times H' \times W'$ number of multiplications. The intermediate feature map goes through pointwise convolution described in Fig. 3.1 (b) Step

2. Filters of size $C \times 1 \times 1 \times 1$ are applied to along the channel direction to produce an output of size $1 \times L' \times H' \times W'$. The final 4D output tensor of size $N \times L' \times H' \times W'$ is generated by applying N number of filters. Pointwise convolution involves $C \times N \times L \times H' \times W'$ number of multiplications. To compare the computational cost of 3D depthwise separable convolution with the standard 3D convolution, we compute the ratio of the number of multiplications involved in these two types of convolutions as follows:

$$\frac{\text{3D depthwise separable convolution}}{\text{3D traditional convolution}} = \frac{C \times K \times K \times K \times L \times H' \times W' + C \times N \times L \times H' \times W'}{C \times K \times K \times K \times N \times L \times H' \times W'} = \frac{1}{N} + \frac{1}{K^3} \quad (3.2)$$

Hence this decomposition process can reduce the computational cost of the standard 3D convolution by $\frac{1}{N} + \frac{1}{K^3}$, where N is the number of output channels and K is the filter size.

3.3 Training Loss Function and Evaluation Metrics

To train the network, we choose the widely used mean squared error (MSE) as the loss function. The MSE between the ground-truth frame \mathbf{Y}_k with time index k and the corresponding predicted frames $\hat{\mathbf{Y}}_k$ is calculated as follows:

$$\text{MSE} = \frac{1}{C \times M \times N} \sum_{c=1}^C \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (Y_k(c, i, j) - \hat{Y}_k(c, i, j))^2 \quad (3.3)$$

To evaluate the model, we use the MSE as a metric in Chapter 4. Additionally, the quality of the predicted frame $\widehat{\mathbf{Y}}_k$ compared to the original frame \mathbf{Y}_k , is evaluated using the peak-signal-to-noise ratio (PSNR), which is defined as,

$$\text{PSNR} = 10 \log_{10} \frac{255^2}{\text{MSE}(\mathbf{Y}_k, \widehat{\mathbf{Y}}_k)} \quad (3.4)$$

Besides, the structural similarity index (SSIM) is also calculated. It is a metric consistent with human subjective opinion, which evaluates the structural similarity between two images and is a combination of three functions [58]:

$$\text{SSIM}(\mathbf{Y}_k, \widehat{\mathbf{Y}}_k) = l(\mathbf{Y}_k, \widehat{\mathbf{Y}}_k)^\alpha \cdot c(\mathbf{Y}_k, \widehat{\mathbf{Y}}_k)^\beta \cdot g(\mathbf{Y}_k, \widehat{\mathbf{Y}}_k)^\gamma, \quad (3.5)$$

where l, c and g are the luminance, contrast and structure comparison measures between \mathbf{Y}_k and $\widehat{\mathbf{Y}}_k$ and α, β and γ are parameters to define the relative importance of these three components [58]. The final reported PSNR and SSIM are averaged over all N predicted frames. Higher PSNR and SSIM indicate that the predicted frames have higher quality.

Besides the aforementioned accuracy metrics, the model efficiency is also evaluated by calculating the model size measured in megabytes (MB) and the number of trainable parameters measured in millions (M). We also use the evaluation metric giga floating point operations (GFLOPs) to infer the computational complexity of the model. GFLOPs calculate the total number of floating point operations, such as addition, subtraction, multiplication and division needed for model inference. Lower GFLOPs usually indicate less computationally expensive models.

CHAPTER 4

Proposed Method Using 3D Separable CNN and LSTM

4.1 Introduction

Recent studies show the potential of 3D CNNs in learning spatio-temporal dynamics of videos better than 2-D CNNs and RNNs and it has been used in various video representation learning tasks [26,41]. One major challenge of the task of video prediction is its heavy computational intensity, due to their complex structures and large amount of model parameters along with the inherent uncertainty of the task. Hence, a lightweight approach for video prediction which significantly reduces the number of parameters while achieving similar prediction capability is proposed in this chapter. A lightweight deep networking model with 3D separable convolutions and 3D depth-wise separable ST-LSTM (spatio-temporal LSTM) is incorporated for the first time in literature for the task of video prediction. Section 4.2 elaborates the method which uses a reversible network [60, 61] as baseline. Section 4.3 demonstrates the effectiveness of the method through experimental setups and results compared with the current state-of-the-art models on the three datasets, which can be achieved with reasonable accuracy-complexity trade-offs.

4.2 Proposed Method

The network consists of a two-way auto-encoder (AE) and a reversible predictive module (RPM), both of which are built with 3D separable convolution layers. The proposed network was inspired by CrevNet [26], which uses the i-RevNet [61] architecture to

preserve information during the feature extraction process. Section 4.2.1 describes the reversible networks.

4.2.1 Reversible Networks

In a neural network, we need to store the activations in each layer in memory in order to calculate gradients during backpropagation, which in turn increases the memory consumption of neural networks. Reversible networks (RevNet) [60] were designed to resolve this issue. RevNet is composed of a series of *reversible blocks*. Each layer’s activations can be reconstructed exactly from the subsequent layer’s activations, using invertible operations. This enables the network to perform backpropagation without storing the activations in memory. Our proposed method utilizes i-RevNet [61] which extends RevNet by replacing the remaining non-invertible components of RevNet by invertible ones.

4.2.2 Proposed Prediction Network

Fig. 4.1 depicts the proposed video prediction architecture. The input of the network is a 4D tensor \mathbf{X}_{t-1} of shape $C \times 3 \times H \times W$ (channel \times temporal length \times height \times width), representing three consecutive video frames at time steps $t - 1$, t , $t + 1$. The number of channels C is set as 1 and 3 for grayscale and RGB images, respectively. The output of the network is a 4D tensor \mathbf{X}_t , representing the predicted video frames at time steps t , $t + 1$, $t + 2$.

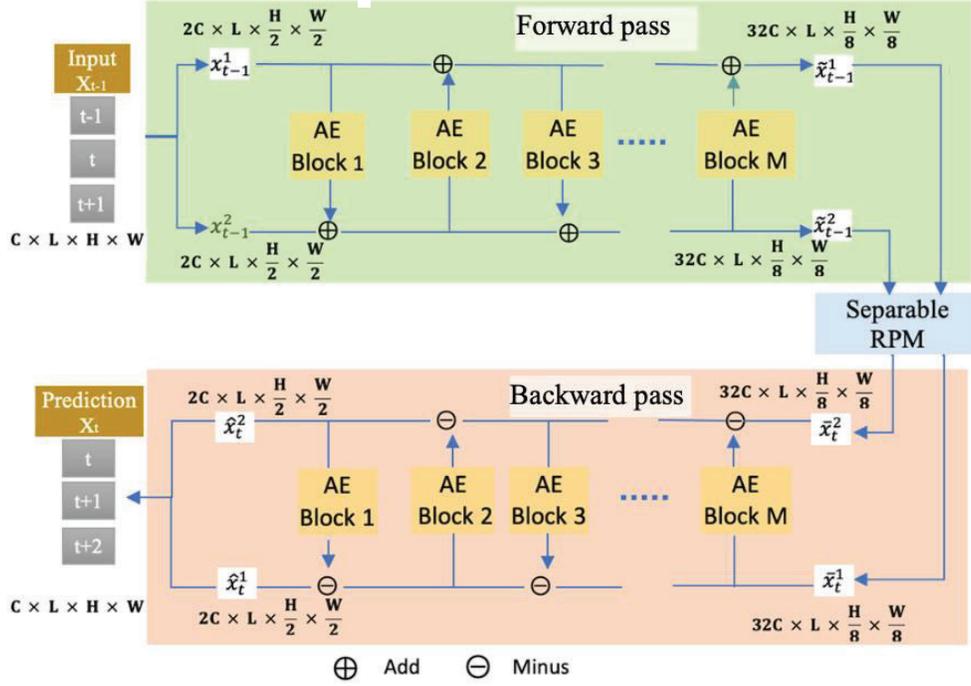


Fig. 4.1: Proposed prediction architecture

As shown in the upper branch of Fig. 4.1, the network input X_{t-1} is split channel-wise into two groups, x_{t-1}^1 and x_{t-1}^2 each with dimension $2C \times L \times \frac{H}{2} \times \frac{W}{2}$. Each of these input groups goes through a forward pass of the AE, consisting of M layers of AE blocks. During the forward pass, one group passes through an AE block and is added to the other input group. This process continues in an alternating fashion, thus forming the two output groups \tilde{x}_{t-1}^1 and \tilde{x}_{t-1}^2 , each of size $32C \times L \times \frac{H}{8} \times \frac{W}{8}$.

Afterwards, \tilde{x}_{t-1}^1 and \tilde{x}_{t-1}^2 are fed to the separable RPM, as shown in Fig. 4.3. The separable ST-LSTM outputs two groups of feature maps \tilde{x}_t^1 , \tilde{x}_t^2 . They go through the backward pass of the two-way AE in an alternating manner similar to the forward pass, as

shown in the lower branch in Fig. 4.1, to output the two predicted channel groups \hat{x}_t^1 and \hat{x}_t^2 . Finally, the two predicted groups are merged to form the final predicted video clip \mathbf{X}_t (frames at time steps $t: t + 2$).

4.2.2.1 AE Block with 3D Separable Convolutions

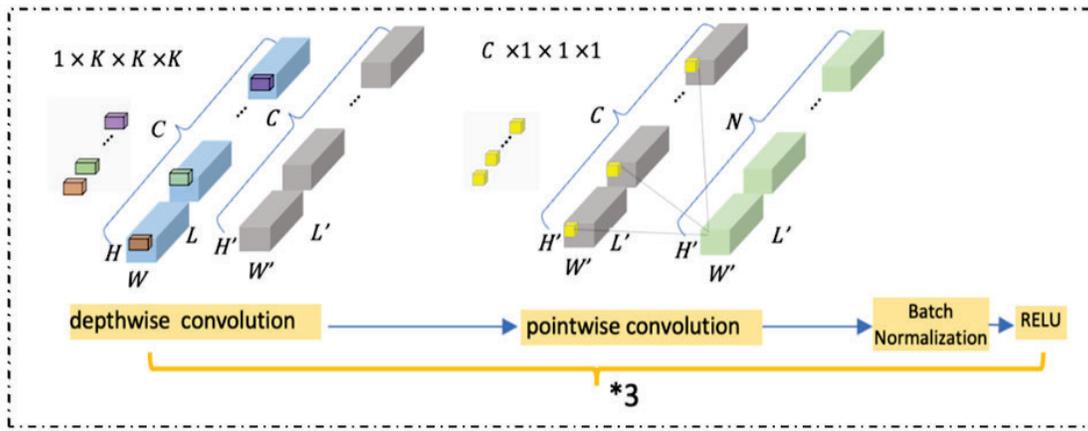


Fig. 4.2: The AE block

While the baseline model CrevNet [26] adopts standard 3D convolution to extract spatial-temporal features, our proposed model adopts separable 3D convolutions to reduce model size and computational complexity. This subsection explains Fig. 4.2 (an AE block) along with the proposed 3D separable convolution. An AE block consists of three 3D separable convolutional layers, each of which is followed by batch normalization and ReLU activation function.

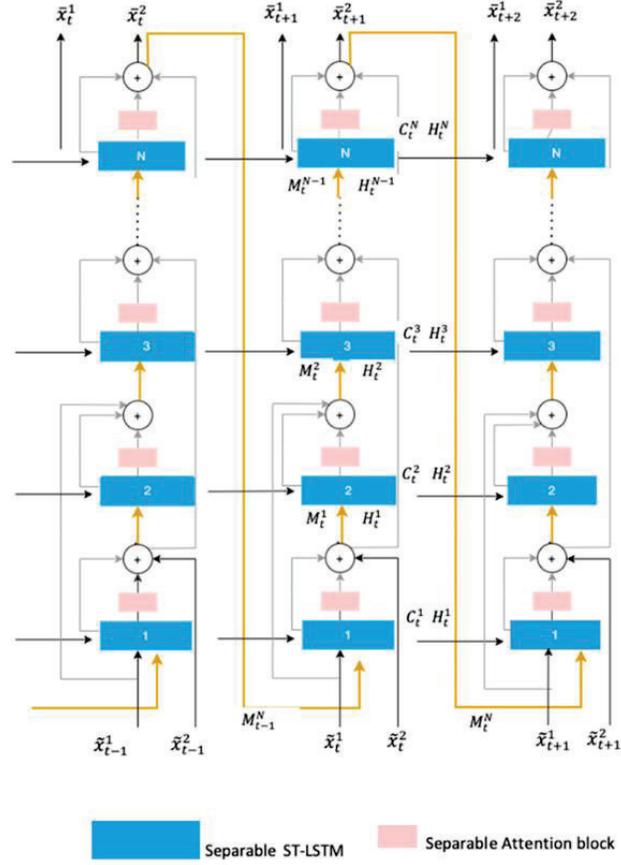


Fig. 4.3: Separable RPM

4.2.2.2 ST-LSTM Block with 3D Separable Convolutions

The proposed separable RPM as shown in Fig. 4.3 processes the output feature groups of the two-way AE. At time step $t - 1$, the input feature groups are \tilde{x}_{t-1}^1 and \tilde{x}_{t-1}^2 . They go through N blocks, each consists of a separable ST-LSTM module (the blue rectangle in Fig. 4.3) and a separable attention module (the pink rectangle in Fig. 4.3). Both of these two modules are constructed by the proposed 3D separable convolutions. In particular, the

attention module helps to form a weighted sum of the two groups, and it consists of one 3D separable convolution layer followed by a sigmoid activation operation.

In Fig. 4.3 the orange arrows denote the spatiotemporal memory flow M_t^l , and the black arrows denote the temporal memory flow C_t^l and the hidden state H_t^l transitions of ST-LSTMs. The superscript l denotes the l -th layer of ST-LSTM, $l = 1, 2, \dots, N$, and t denotes the time step. The outputs of the separable RPM at time step $t - 1$ are two feature groups \bar{x}_t^1 and \bar{x}_t^2 , along with a spatiotemporal feature M_{t-1}^N that is taken as an input of the separable RPM at time step t . The separable RPM processes the data in a similar way as that for time step $t - 1$ for every time step.

Table 4.1. Configuration of the proposed 3D separable ST-LSTM - based network

Block	Output tensor/tensors	Size
Input	\mathbf{X}_{t-1}	$C \times L \times H \times W$
Input split	x_{t-1}^1, x_{t-1}^2 ,	$2C \times L \times H/2 \times W/2$
Forward pass	$\tilde{x}_{t-1}^1, \tilde{x}_{t-1}^2$	$32C \times L \times H/8 \times W/8$
Separable RPM	\bar{x}_t^1, \bar{x}_t^2	$32C \times L \times H/8 \times W/8$
Backward pass	\hat{x}_t^1, \hat{x}_t^2	$2C \times L \times H/2 \times W/2$
Output	$\hat{\mathbf{X}}_t$	$C \times L \times H \times W$

4.3 Training and Evaluation Setup

The models were trained with the PyTorch framework using an NVIDIA Tesla V100 32 GB GPU. The ADAM optimizer was used to minimize the L2 loss between the input and

the predicted frames. The initial learning rate was set at 0.002 with an exponential decay factor of 0.2 for every 50 epochs.

To evaluate the performance of our model, we calculated the mean squared error (MSE) and/or peak-signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) between the ground-truth and predicted frames. Lower MSE and higher PSNR/SSIM indicates better predictions.

4.4 Experiments and Analysis

In this section, we demonstrate the effectiveness of the proposed method, through extensive experiments done on the synthetic Moving MNIST dataset [21], and two real-world datasets KTH action [53] and BAIR [63].

4.4.1 Moving MNIST Dataset

This is a widely used synthetic grayscale video prediction dataset, with a frame size of $1 \times 64 \times 64$. The first dimension represents the grayscale channel. This dataset has two subsets: Moving MNIST-2 and Moving MNIST-3. Moving MNIST-2 consists of sequences of 20 frames, in which two digits continuously move with constant velocity and angle, bouncing inside a black 64×64 frame, potentially overlapped and occluded. Moving MNIST-3 contains frames with 3 digits, potentially overlapped. The digits move in a constant velocity and angle, bouncing off the edges of the frame. Our model and the state-of-the-art models were trained on Moving MNIST-2 and tested on both Moving MNIST-2 and Moving MNIST-3.

Table 4.2. Quantitative Results on Moving MNIST Dataset [21]. The best, and second-best results of each metric are highlighted in red and blue, respectively.

Model	MNIST-2		MNIST-3		# Params	Model size (bytes)	FLOPs (G)
	<i>MSE</i>	<i>SSIM</i>	<i>MSE</i>	<i>SSIM</i>			
PredRNN [24]	55.4	0.879	83.6	0.838	23.86M	93 MB	115.9
PredRNN++ [30]	46.2	0.902	68.4	0.864	15.09M	57.42 MB	106.8
CrevNet [26]	24.3	0.936	40.6	0.916	5M	60.2 MB	1.0
Proposed Model	44.1	0.916	63.1	0.891	368.96 K	4.8 MB	0.08

The proposed prediction architecture for this dataset is composed of a two-way autoencoder with 12 AE blocks for both forward and backward pass and 8 RPMs. The batch size was chosen as 32 and model training was stopped after 250,000 iterations.

Table 4.2 compares the prediction accuracy, model parameters, model size, and computational complexity of our proposed model with state-of-the-art methods PredRNN [24], PredRNN++ [30] and CrevNet [26]. The best and second-best results of each metric are highlighted in red and blue, respectively. Our proposed model is superior in terms of fewest model parameters, smallest model size and lowest computational complexity measured by giga floating point operations (GFLOPs). In terms of prediction accuracy, the proposed model achieves the second best MSE and SSIM values.

From the visual results in Fig. 4.4 we observe that PredRNN[24] suffers from blurring and maintaining the shape of digits over time. For example, digit 3 for Moving MNIST-3 in

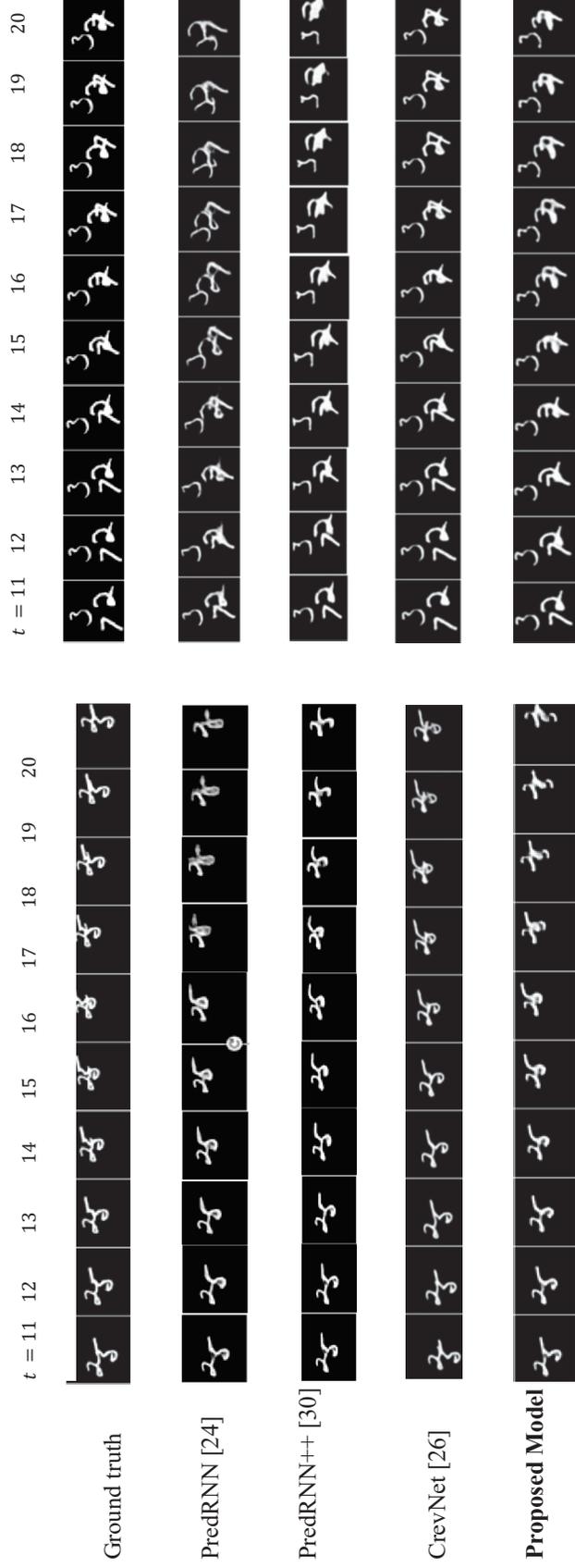


Fig. 4.4: Qualitative results on Moving MNIST-2 dataset (left) and Moving MNIST-3 dataset (right).

Top row: the ground-truth frames to be predicted at time steps $t = 11$ to $t = 20$.

Remaining rows: the predictions of different models and our proposed model.

the right figure predicted by PredRNN[24] and PredRNN++[30] loses its shape with the passage of time. Though CrevNet produces sharper images, our proposed model successfully tracked the motion of the digits without blurring, yet requiring much smaller model size and less computational complexity.

4.4.2 KTH Action Dataset

KTH Action [53] is a grayscale dataset initially used for action recognition. It has sequences of 25 individuals doing six types of actions: walking, running, jogging, boxing, hand waving and clapping. Videos are shot with static camera at 25 fps in four settings, namely, outdoors, outdoors with scale variation, outdoors with different clothes, and indoors. Individuals 1-16 are used for training and individuals 17-25 are used for testing. The training strategy in [30] is followed and the frames were resized to a resolution of 128×128 pixels.

The prediction architecture of our proposed model for this dataset consists of a two-way autoencoder with 14 AE blocks for both forward and backward pass and 16 RPMs. In the testing phase, the model observed the first 10 frames in each test sequence and predicts the next 20 frames. Table 4.3 summarizes the quantitative results of our model compared to the state-of-the-art models PredRNN [24], PredRNN++ [30], and CrevNet [26]. Again, our model achieves the best performance in terms of model size and complexity. Besides, our model easily outperforms PredRNN in PSNR and SSIM. Although our PSNR is ranked third, the SSIM of our proposed model is the second-best. Since SSIM is more consistent with human perception than PSNR, this indicates the predicted frames of our proposed model have better visual quality than PredRNN and PredRNN++.

Table 4.3 Quantitative Results on KTH dataset. The best, and second-best results of each metric are highlighted in red and blue, respectively.

Model	PSNR	SSIM	# Params	Model size	FLOPS (G)
PredRNN [24]	26.23	0.839	23.86M	93 MB	123.9
PredRNN++ [30]	28.41	0.865	15.09M	57.42 MB	115.8
CrevNet [26]	28.70	0.8768	9.89M	70.9 MB	7.76
Proposed Model	27.02	0.8671	727.26K	9.2 MB	0.6

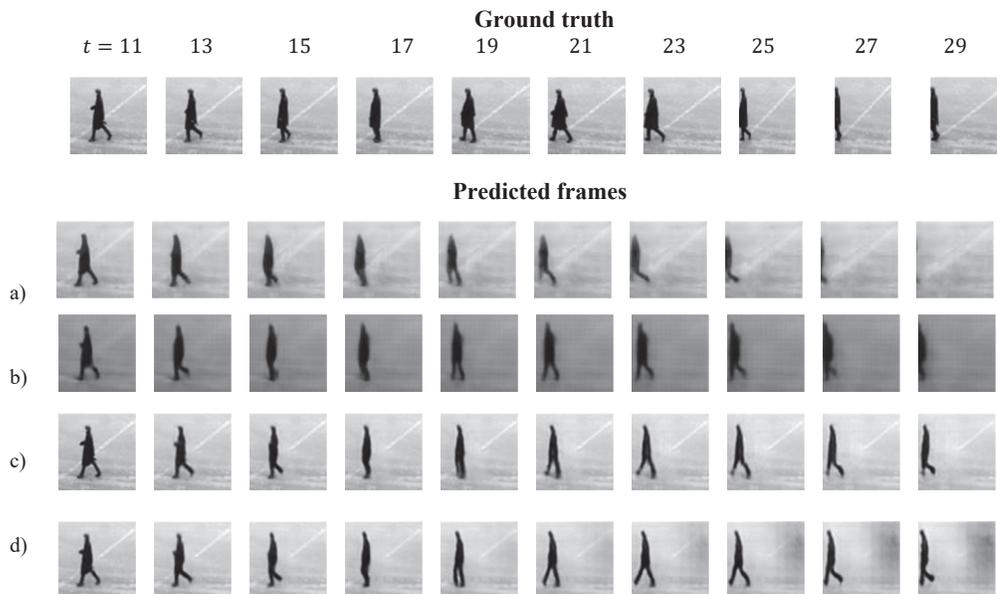


Fig. 4.5: Qualitative results on KTH action dataset.

Top row: the ground-truth frames to be predicted at time steps $t = 11, 13, 15, 17, 19, 21, 23, 25, 27, 29$. Remaining rows: the predictions of a) PredRNN, b) PredRNN++, c) CrevNet, and d) our proposed model.

Fig. 4.5 shows the predicted frames of all compared models. Due to space limitations, we included frames only from specific time steps. We observe that our model outperforms PredRNN and PredRNN++ by carrying motion information and protecting detailed structure of the person across longer time steps, while the prediction results of PredRNN

and PredRNN++ become blurry over time. Though some of the detailed spatial features (e.g, the white line) are not preserved by the 25th frame, our model captures key information of the moving object. CrevNet [26] does produce better images, but we observe that at some time steps, our model is adept in learning features. For example, at $t = 19$, the shape of legs is better shown in our model than in CrevNet.

4.4.3 BAIR Dataset

This is a popular color video dataset in video prediction literature, the BAIR towel-pick dataset [63], which has sequences of a robotic arm picking and placing objects like towels, shirts, and jackets. This is a challenging dataset due to the stochastic arm movements of the robot. The original frames were resized to a resolution of 64×64 . The model’s prediction architecture and evaluation metrics are similar to those for the KTH Action

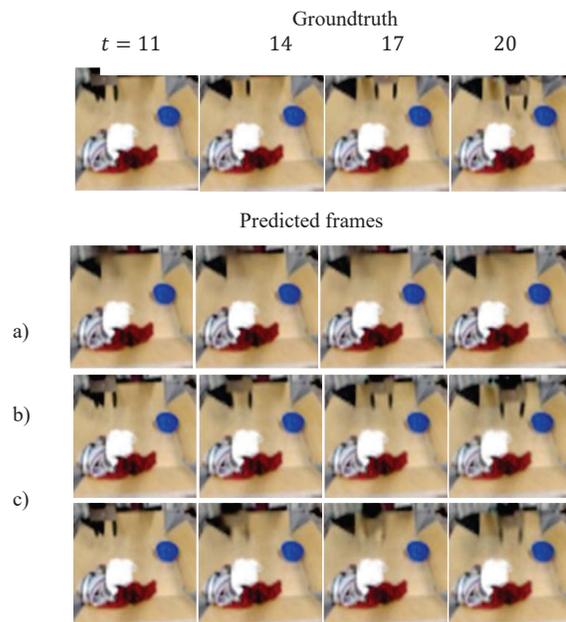


Fig. 4.6: Qualitative results on the BAIR dataset.

Top row: the ground-truth frames to be predicted at time steps $t = 11, 14, 17, 20$. Remaining rows: the predictions of a) SVG, b) CrevNet, and c) our proposed model.

dataset. The future 10 frames were predicted after observing 10 preceding frames. We compare our model’s efficacy to the state-of-the-art models SVG [64] and CrevNet [26]. As shown in Table 4.4, our model outperforms these two models in computational efficiency along with the second best PSNR and SSIM values. Fig. 4.6 demonstrates a few samples from a clip where a robotic arm moves above the objects on table. We observe that all the spatial features and colors of the objects are captured by our model till the 20th frame. While the movements of robotic arms are not all captured along time in SVG, we accurately predict it in the frame $t = 11$. Though the arms of the robot become blurry compared to CrevNet, we predict its positions accurately.

Table 4.4 Quantitative Results on BAIR dataset. The best, and second-best results of each metric are highlighted in red and blue, respectively.

Model	PSNR	SSIM	# Params	Model size	FLOPS (G)
SVG-LP [64]	19.13	0.7742	22.8M	91.5 MB	123.9
CrevNet [26]	23.16	0.8139	9.89M	70.9 MB	7.76
Proposed Model	22.92	0.7963	727.26K	9.2 MB	0.6

4.5 Conclusion

In this chapter, we propose a lightweight video prediction method based on 3D separable convolutions and LSTMs. Experimental studies demonstrate the efficiency of our model on both synthetic and real-world datasets. With significantly fewer model parameters and lower computational complexity, our proposed model is able to achieve reasonable prediction accuracy and visually pleasing results.

Chapter 5

Proposed Method Based on Hybrid Transformer-LSTM Network

5.1 Introduction

While CNNs can extract local features, RNNs are specifically used to learn sequential representations. To benefit from both CNNs and RNNs, other approaches [23]–[28] proposed to combine CNN and RNN and learn spatiotemporal features from video data. Among these works, many adopted the long short-term memory (LSTM) as their RNN structure, which led to the family of ConvLSTM models.

Transformers which have primarily demonstrated success in natural language processing (NLP) [36] and several vision tasks [37]–[39] were also recently utilized for video prediction [40]–[42]. Transformer models are capable of capturing long-range dynamics without the vanishing gradient problem of recurrent networks and have the advantage of parallelism with the self-attention mechanism [36]. However, the accuracy of transformer models usually comes at the price of huge computational cost [43].

Recently, researchers across the NLP and CV communities [44], [45] advocated to shift the focus to Green AI with energy-efficient deep learning solutions, rather than continuously pushing red AI methods to reach state-of-the-art (SOTA) results using massive computational power. Our work focuses on developing an efficient video frame prediction model with reduced model size, fewer parameters, and low computational

complexity, while achieving competitive prediction accuracy. Since both transformer and ConvLSTM models have achieved superior accuracy in predictive learning, in this chapter, we propose a hybrid transformer-LSTM (3DTransLSTM) model to predict future video frames. To learn spatiotemporal dynamics from video data, the proposed 3DTransLSTM network adopted 3D separable convolutions to extract features along the temporal, height, and width dimensions. Our main contributions can be summarized as follows:

- For the first time in the literature, we proposed a hybrid transformer-LSTM (3DTransLSTM) network for the video prediction task. On one hand, the transformer module can leverage long-range correlations among multiple successive video frames, and parallelize the computation with its self-attention mechanism. On the other hand, the LSTM module can enable spatiotemporal information flow vertically within each time step and horizontally among multiple time steps.
- The proposed 3DTransLSTM adopts 3D convolutions to effectively learn spatiotemporal dynamics. To reduce computational cost, the standard 3D convolution is decomposed into a 3D depthwise convolution and a pointwise convolution, which reduced model size, trainable parameters, and floating-point operations (FLOPs). To the best of our knowledge, this is the first time that 3D separable convolution is utilized in a hybrid transformer-LSTM network.
- Qualitative and quantitative experimental results on popular video prediction datasets show that, compared to SOTA methods, the proposed model achieves competitive video frame prediction accuracy with significantly smaller model size, fewer model parameters, and less computational cost. Further, we demonstrated the

generalization ability of the proposed model by testing the model on video sequences completely unseen in the training dataset.

The chapter is organized as follows. In Section 5.2, we describe the preliminaries on vision transformer architecture, previous hybrid transformer-LSTM on NLP tasks and explain the rationale behind our proposed model. Section 5.3 elaborates on our proposed 3DTransLSTM model in detail. Section 5.4 presents experiments on four video prediction datasets and comparison studies with prior arts.

5.2 Preliminaries

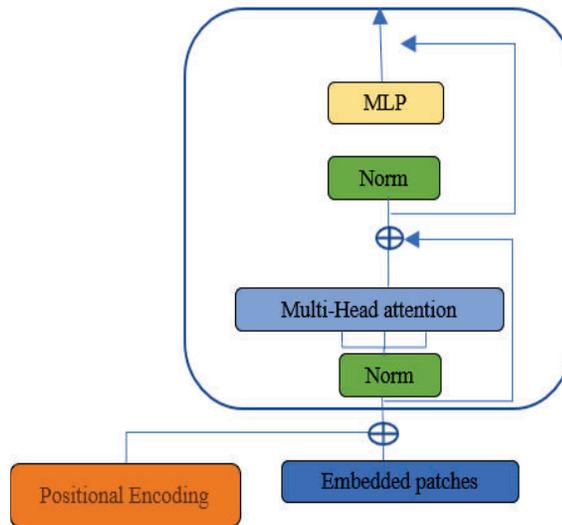


Fig. 5.1: Transformer architecture

Vision Transformer (ViT) [46] introduced the computer vision world to the arena of transformers, which was initially created for NLP tasks. ViT splits images to non-overlapping patches as tokens and then embeds it into a set of encoded vectors. The embeddings are added with the positional encodings to preserve positional identity and are

sent to N layers of transformer encoder blocks, as depicted in Fig. 5.1. Transformer-based video prediction methods are further discussed in Section 2.4.

5.3 Proposed Method

In this section, we elaborate our algorithm in detail. Fig. 5.2 shows the overall framework of the proposed architecture for three time steps $t - 1, t, t + 1$. The proposed video frame prediction network enables temporal information flow indicated by the four arrows connecting adjacent time steps.

5.3.1 Algorithm Overview

As shown in the middle column of Fig. 5.2, at time step t , each frame in the 4D video tensor $\mathbf{X}_t \in \mathbb{R}^{C \times L \times H \times W}$ is spatially split into a sequence of $p \times p$ patches, forming a tensor $\mathbf{P}_t \in \mathbb{R}^{Cp^2 \times L \times H/p \times W/p}$, where $\frac{H}{p} \times \frac{W}{p}$ is the resulting number of patches, and Cp^2 is the length of each flattened patch. Next, \mathbf{P}_t is processed by spatial embedding and positional encoding to generate an output tensor $\mathbf{Z}_t \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$, where d_{model} is the embedded dimension. \mathbf{Z}_t is then passed through six 3DTransLSTM blocks, which leverage transformers and LSTM to extract spatiotemporal features and generate an output tensor $\mathbf{Z}_{t+1} \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$. It is then processed by the prediction head to generate the predicted patches $\hat{\mathbf{P}}_{t+1} \in \mathbb{R}^{Cp^2 \times L \times H/p \times W/p}$, which is reshaped to form the final output frames $\hat{\mathbf{X}}_{t+1} \in \mathbb{R}^{C \times L \times H \times W}$. Table 5.1 summarizes the configuration of the proposed network. In the following subsections, we elaborate the proposed network components in detail.

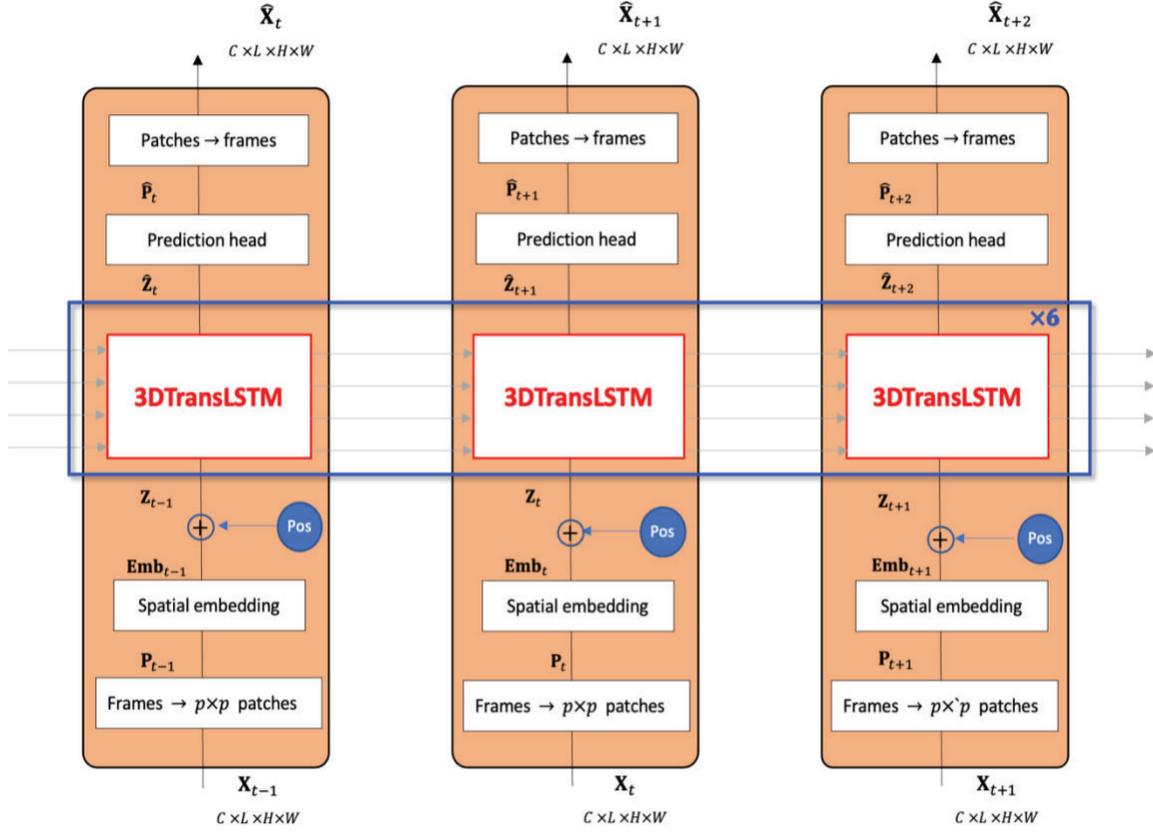


Fig. 5.2: The overall architecture of the proposed network for three time steps $t - 1$, t , and $t + 1$.

Table 5.1. Configuration of the proposed network

Block	Output tensor	Size
Frames \rightarrow patches	\mathbf{P}_t	$Cp^2 \times L \times H/p \times W/p$
Spatial embedding	\mathbf{Emb}_t	$d_{model} \times L \times H/p \times W/p$
Pos + \mathbf{Emb}_t	\mathbf{Z}_t	$d_{model} \times L \times H/p \times W/p$
6 \times 3D TransLSTM	$\hat{\mathbf{Z}}_{t+1}$	$d_{model} \times L \times H/p \times W/p$
3D separable predictor	\mathbf{P}_{t+1}	$Cp^2 \times L \times H/p \times W/p$
Patches \rightarrow frames	$\hat{\mathbf{X}}_{t+1}$	$C \times L \times H \times W$

5.3.2 Spatial Embedding and Positional Encoding

The proposed 2D separable embedding module processes the input patches $\mathbf{P}_t \in \mathbb{R}^{(Cp^2) \times L \times H/p \times W/p}$ to produce the embedded feature maps. It adopts two 2D depthwise separable convolutional layers. As shown in (5.1), 2D depthwise separable convolution G is adopted to output intermediate feature map $\mathbf{J}_1 \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$ and another 2D depthwise separable convolution S outputs $\mathbf{J}_2 \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$. Both G and S of the 2D separable convolutions are applied frame by frame. \mathbf{J}_1 and \mathbf{J}_2 are then added and passed through the *Dropout* layer to produce the embedded feature maps $\mathbf{Emb}_t \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$.

$$\begin{aligned}
 \mathbf{J}_1 &= \text{LeakyReLU}(G^{Cp^2 \times 1 \times 1} * (G^{1 \times 7 \times 7} \otimes \mathbf{P}_t)) \\
 \mathbf{J}_2 &= \text{LeakyReLU}(S^{d_{model} \times 1 \times 1} * (S^{1 \times 5 \times 5} \otimes \mathbf{Y}_1)) \\
 \mathbf{Emb}_t &= \text{Dropout}(\mathbf{J}_1 + \mathbf{J}_2)
 \end{aligned} \tag{5.1}$$

In (5.1), \otimes is the 2D depthwise operation, and $*$ is the 1D pointwise convolution operation. G adopts a 2D depthwise convolution with (Cp^2) filters of size $1 \times 7 \times 7$ followed by a 1D pointwise convolution with d_{model} filters of size $((Cp^2) \times 1 \times 1)$. Similarly, S adopts a 2D depthwise convolution with d_{model} filters of size $1 \times 5 \times 5$ followed by a 1D pointwise convolution with d_{model} filters of size $d_{model} \times 1 \times 1$.

To preserve the positional information of the input sequence, fixed positional encoding $\mathbf{Pos} \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$ is calculated via (5.2) [41], where i represents the channel

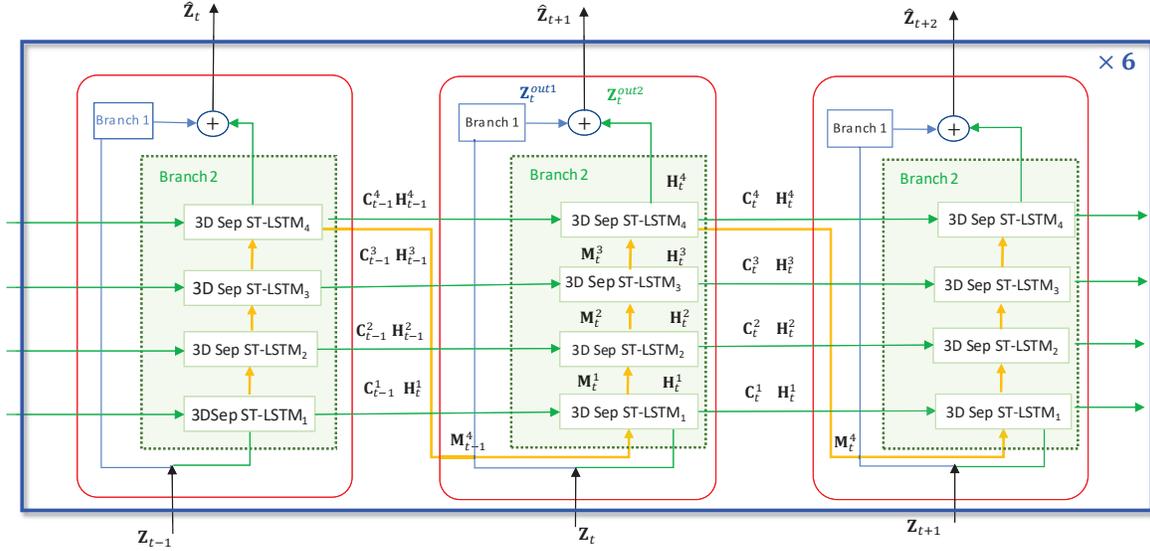


Fig. 5.3. The stack of six 3DTransLSTM blocks for time steps $t - 1, t, t + 1$. Each 3DTransLSTM block consists of Branch 1 (blue arrow) and Branch 2 (green arrow). Branch 2 further consists of 4 3D separable convolution-based ST-LSTM (3D SepST-LSTM) layers

index, $0 \leq i < d_{model}$, l is the temporal index, (h, w) are the spatial indices, and $k = 10^4$.

$$\mathbf{Pos}_{i,l,h,w} = \sin(l/k^{i/d_{model}}) \quad i \text{ even}$$

$$\mathbf{Pos}_{i,l,h,w} = \cos(l/k^{i/d_{model}}) \quad i \text{ odd} \quad (5.2)$$

\mathbf{Pos} is added to the embedded feature maps \mathbf{Emb}_t to generate the output $\mathbf{Z}_t \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$ by

$$\mathbf{Z}_t = \mathbf{Emb}_t + \mathbf{Pos}. \quad (5.3)$$

5.3.3 3D Transformer-LSTM

The stack of six 3DTransLSTM blocks across three time steps $t - 1, t, t + 1$ as shown in Fig. 5.2 are described in detail in Fig. 5.3. Take time step t as an example: the proposed

network takes the positioned feature maps $\mathbf{Z}_t \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$ as input and passes it through a stack of six 3DTransLSTM blocks to output $\hat{\mathbf{Z}}_{t+1}$. In each 3DTransLSTM block, the input is processed by two parallel branches. Branch 1 is indicated by the blue arrow. It extracts features within time step t through transformers, and outputs \mathbf{Z}_t^{out1} . Branch 2 enables information flow across three time t steps $t-1, t, t+1$ through ST-LSTM structures, and outputs \mathbf{Z}_t^{out2} . In the following we give detailed descriptions of Branch 1 and Branch 2.

Branch 1

Its network structure is separately depicted in Fig. 5.4. It adopts two sub-blocks: 1) 3D separable convolution-based self-attention (*3DSepSA*), and 2) 3D separable convolution-based feed-forward network (*3DSepFFN*). Layer normalization (LN) is applied before each sub-block, and residual connection is applied after each sub-block.

3DSepSA

As shown in Fig. 5.4 (a), first, the input tensor \mathbf{Z}_t goes through an LN layer by

$$\mathbf{Z}_t^{attn_in} = LN(\mathbf{Z}_t), \quad (5.4)$$

As shown in Fig. 5.4 (b) the *3DSepSA* module then takes $\mathbf{Z}_t^{attn_in} \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$ as the input and then calculates three tensors $\bar{\mathbf{Q}}, \bar{\mathbf{K}}, \bar{\mathbf{V}}$ with dimension $d_{model} \times L \times H/p \times W/p$ using 3D separable convolutional kernels W_q, W_k, W_v , as follows:

$$\bar{\mathbf{Q}} = W_q^{d_{model} \times 1 \times 1 \times 1} * (W_q^{1 \times 3 \times 3 \times 3} \odot \mathbf{Z}_t^{attn_in})$$

$$\begin{aligned}\bar{\mathbf{K}} &= W_k^{d_{model} \times 1 \times 1 \times 1} * (W_k^{1 \times 3 \times 3 \times 3} \circledast \mathbf{z}_t^{attn_in}) \\ \bar{\mathbf{V}} &= W_v^{d_{model} \times 1 \times 1 \times 1} * (W_v^{1 \times 3 \times 3 \times 3} \circledast \mathbf{z}_t^{attn_in})\end{aligned}\quad (5.5)$$

where $*$ denotes 1D pointwise convolution and \circledast denotes 3D depthwise convolution.

To calculate the attention of the above tensors, we first transpose the tensors $\bar{\mathbf{Q}}, \bar{\mathbf{K}}$ and $\bar{\mathbf{V}}$ to dimension $H/p \times W/p \times L \times d_{model}$. Let $H/p \times W/p$ be the batch size, then the batch has $H/p \times W/p$ samples and each sample is a tensor of size $L \times d_{model}$, denoted by, $\mathbf{Q} \in \mathbb{R}^{L \times d_{model}}$, $\mathbf{K} \in \mathbb{R}^{L \times d_{model}}$, and $\mathbf{V} \in \mathbb{R}^{L \times d_{model}}$.

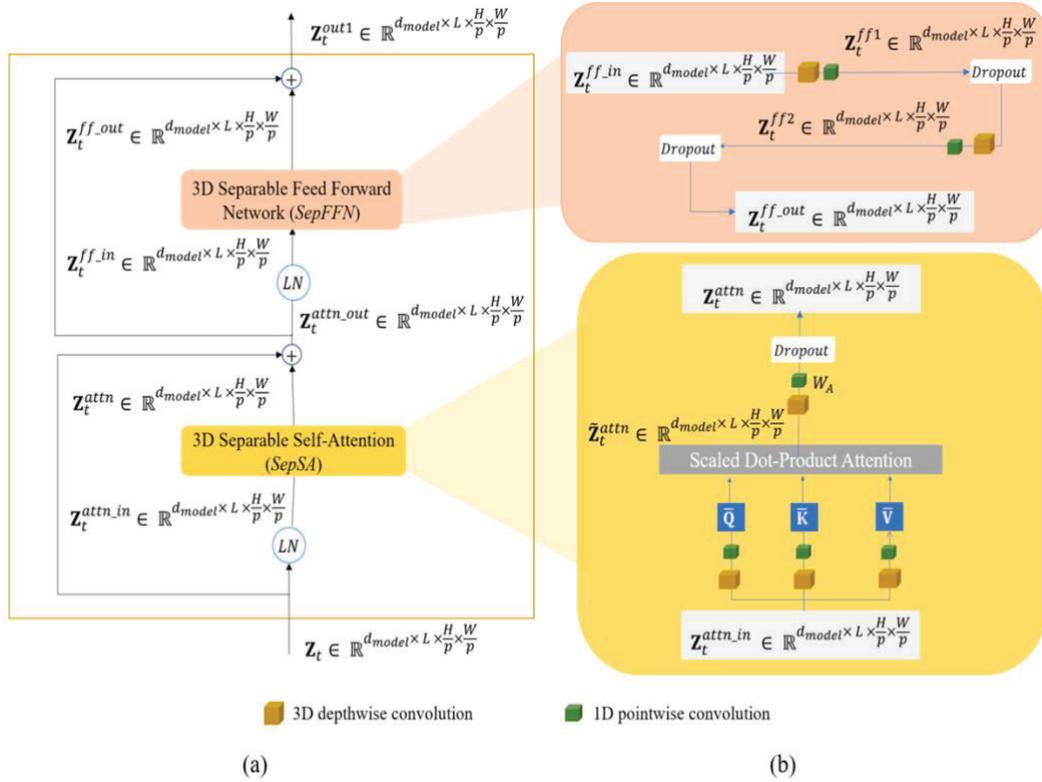


Fig. 5.4. (a) Branch 1 of the 3DTransLSTM block, (b) 3DSepFFN module (top), 3DSepSA module (bottom).

Within each sample, the self-attention $\mathbf{Z} \in \mathbb{R}^{L \times d_{model}}$ among L temporal elements is calculated as,

$$\mathbf{Z} = \text{softMax} \left(\text{Mask} \left(\frac{\mathbf{QK}^T}{\sqrt{d_{model}}} \right) \right) \mathbf{V} \quad (5.6)$$

We use a single-head masked self-attention where the masking mechanism only allows a position to look at the previous tokens and prevents the leaking of information to future tokens [41]. We mask out the attention to future tokens by setting their attention scores to $-\infty$, which generates zero weights after they are passed through the $\text{Softmax}(\cdot)$ function. Next, the self-attentions of all samples in the batch are grouped and transposed to form the output self-attention $\tilde{\mathbf{Z}}_t^{attn} \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$, which is then processed by 3D depthwise separable convolution W_A and a *Dropout* layer as shown in (5.7),

$$\mathbf{Z}_t^{attn} = \text{Dropout} \left(W_A^{d_{model} \times 1 \times 1 \times 1} * (W_A^{1 \times 3 \times 3 \times 3} \odot \tilde{\mathbf{Z}}_t^{attn}) \right) \quad (5.7)$$

Afterwards, a residual connection from the input \mathbf{Z}_t is applied to \mathbf{Z}_t^{attn} to produce the final output of the attention module $\mathbf{Z}_t^{attn_out}$ as

$$\mathbf{Z}_t^{attn_out} = \mathbf{Z}_t^{attn} + \mathbf{Z}_t \quad (5.8)$$

3DSepFFN

The second sub-block of Branch 1, *3DSepFFN* takes the layer normalized $\mathbf{Z}_t^{attn_out}$ as the input

$$\mathbf{Z}_t^{ff_in} = \text{LN}(\mathbf{Z}_t^{attn_out}) \quad (5.9)$$

As shown in Fig. 5.4 (b), inside the $3DSepFFN$ sub-block, there are two 3D depthwise separable convolutional layers W_{ff1} , and W_{ff2} each followed by a *Dropout* layer, consists of the two 3D depthwise separable convolutional layers and transformations inside this sub-block can be formulated as follows:

$$\begin{aligned}
\mathbf{Z}_t^{ff1} &= \text{LeakyRELU}(W_{ff1}^{d_{model} \times 1 \times 1 \times 1} * (W_{ff1}^{1 \times 3 \times 3 \times 3} \odot \mathbf{Z}_t^{ff-in})) \\
\mathbf{Z}_t^{ff2} &= W_{ff2}^{d_{model} \times 1 \times 1 \times 1} * (W_{ff2}^{1 \times 3 \times 3 \times 3} \odot \text{Dropout}(\mathbf{Z}_t^{ff1})) \\
\mathbf{Z}_t^{ff-out} &= \text{Dropout}(\mathbf{Z}_t^{ff2})
\end{aligned} \tag{5.10}$$

The residual connection from the $3DSepSA$ sub-block $\mathbf{Z}_t^{attn-out} \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$ and the output from the 3D separable feed-forward sub-block $\mathbf{Z}_t^{ff-out} \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$ are added to form the final output $\mathbf{Z}_t^{out1} \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$ of Branch 1.

$$\mathbf{Z}_t^{out1} = \mathbf{Z}_t^{ff-out} + \mathbf{Z}_t^{attn-out}. \tag{5.11}$$

Branch 2

At time step t , Branch 2 shown by the green arrow in Fig. 5.3 enables spatiotemporal information flow. It contains four layers of 3D separable convolution- based ST-LSTM (3DSepST-LSTM), and outputs $Z_t^{out2} \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$.

As shown in Fig. 5.3, at the l -th 3D SepST-LSTM layer, $l = 1, 2, 3, 4$, the inputs are the temporal memory cell \mathbf{C}_{t-1}^l and hidden state \mathbf{H}_{t-1}^l delivered horizontally from the same l -th layer in the previous time step $t - 1$, as well as the spatiotemporal memory \mathbf{M}_t^{l-1} and

the hidden state \mathbf{H}_t^{l-1} delivered vertically from the $(l - 1)$ -th 3D SepST-LSTM layer of the current time step t . The outputs are the spatiotemporal memory cell \mathbf{M}_t^l , the hidden state \mathbf{H}_t^l , and the temporal memory cell \mathbf{C}_t^l . While \mathbf{M}_t^l is delivered vertically to the $(l + 1)$ -th 3D SepST-LSTM layer, \mathbf{C}_t^l is delivered horizontally to the $(t + 1)$ -th time step, and \mathbf{H}_t^l is delivered both vertically and horizontally. It is noteworthy that when $l = 1$, the input spatiotemporal memory cell $\mathbf{M}_t^0 = \mathbf{M}_{t-1}^4$, which is the output spatiotemporal memory cell of the 4-th 3D SepST-LSTM layer of the previous time step $t - 1$, and the input hidden state $\mathbf{H}_t^0 = \mathbf{Z}_t$, which is the positional feature input of the entire stack of six 3DTransLSTM blocks.

All the gates and the input \mathbf{Z}_t , hidden \mathbf{H}_t^l and memory states \mathbf{M}_t^l and \mathbf{C}_t^l are 4D tensors of size $\mathbb{R}^{d_{model} \times L \times H/p \times W/p}$, where d_{model} is the channel, L is the temporal length, and H/p and W/p are the height and width, respectively.

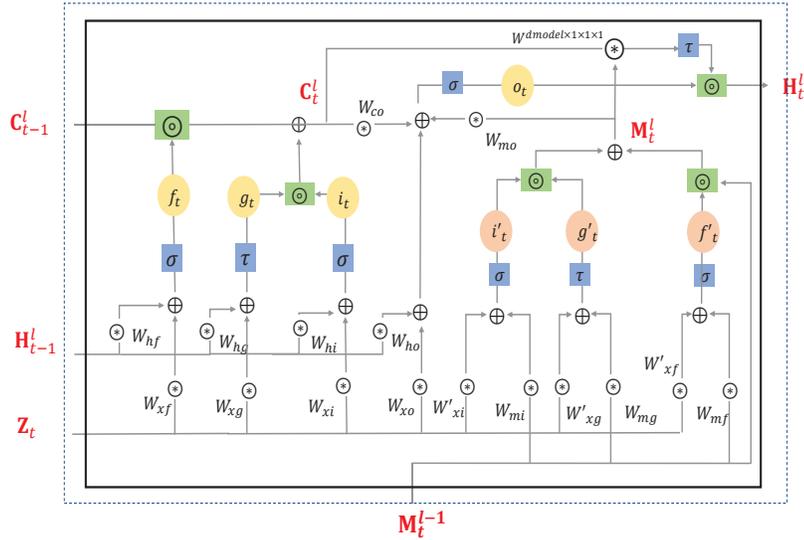


Fig. 5.5 The structure of the l -th 3D SepST-LSTM layer at time step t .

Fig. 5.5 shows the structure of the l -th 3D SepST-LSTM layer at time step t . One set of input gate i_t , forget gate f_t , and input modulation gate g_t are generated by processing hidden states \mathbf{H}_t^{l-1} and \mathbf{H}_{t-1}^l by separable 3D convolution as shown in (5.12). Here \odot denotes the 3D depth-wise convolution, $*$ is the 1D pointwise convolution, σ is the sigmoid function and τ is the tanh function.

$$\begin{aligned}
i_t &= \sigma(W_{xi} * (W_{xi} \odot \mathbf{H}_t^{l-1}) + W_{hi} * (W_{hi} \odot \mathbf{H}_{t-1}^l)) \\
f_t &= \sigma(W_{xf} * (W_{xf} \odot \mathbf{H}_t^{l-1}) + W_{hf} * (W_{hf} \odot \mathbf{H}_{t-1}^l)) \\
g_t &= \tau(W_{xg} * (W_{xg} \odot \mathbf{H}_t^{l-1}) + W_{hg} * (W_{hg} \odot \mathbf{H}_{t-1}^l))
\end{aligned} \tag{5.12}$$

An extra set of input gate i'_t forget gate f'_t , and input modulation gate g'_t are generated by processing the hidden state \mathbf{H}_t^{l-1} and spatiotemporal memory \mathbf{M}_t^{l-1} using 3D separable convolution, as shown in (5.13).

$$\begin{aligned}
i'_t &= \sigma(W'_{xi} * (W'_{xi} \odot \mathbf{H}_t^{l-1}) + W_{mi} * (W_{mi} \odot \mathbf{M}_t^{l-1})) \\
f'_t &= \sigma(W'_{xf} * (W'_{xf} \odot \mathbf{H}_t^{l-1}) + W_{mf} * (W_{mf} \odot \mathbf{M}_t^{l-1})) \\
g'_t &= \tau(W'_{xg} * (W'_{xg} \odot \mathbf{H}_t^{l-1}) + W_{mg} * (W_{mg} \odot \mathbf{M}_t^{l-1}))
\end{aligned} \tag{5.13}$$

Afterwards, the output temporal memory cell \mathbf{C}_t^l and spatiotemporal memory \mathbf{M}_t^l are generated by (e3) and (e4), respectively, where \odot represents the Hadamard product.

$$\mathbf{C}_t^l = f_t \odot \mathbf{C}_{t-1}^l + i_t \odot g_t \tag{5.14}$$

$$\mathbf{M}_t^l = f'_t \odot \mathbf{M}_t^{l-1} + i'_t \odot g'_t \tag{5.15}$$

The output gate o_t is generated by processing \mathbf{H}_t^{l-1} , \mathbf{H}_{t-1}^l , \mathbf{M}_t^l , and \mathbf{C}_t^l again by 3D separable convolution, as shown in (5.16).

$$\begin{aligned}
o_t = & \sigma(W_{xo} * (W_{xo} \circledast \mathbf{H}_t^{l-1}) + W_{ho} * (W_{ho} \circledast \mathbf{H}_{t-1}^l) \\
& + W_{mo} * (W_{mo} \circledast \mathbf{M}_t^l) + W_{co} * (W_{co} \circledast \mathbf{C}_t^l))
\end{aligned} \tag{5.16}$$

Finally, the hidden state \mathbf{H}_t^l is generated by

$$\mathbf{H}_t^l = o_t \odot \tau(W^{d_{model} \times 1 \times 1 \times 1} * [\mathbf{C}_t^l, \mathbf{M}_t^l]) \tag{5.17}$$

where $[\mathbf{C}_t^l, \mathbf{M}_t^l]$ is the channel concatenation of \mathbf{C}_t^l and \mathbf{M}_t^l in equations (5.12), (5.13) and (5.16), the depthwise 3D convolutional kernel size is set as $1 \times 3 \times 3 \times 3$ and pointwise convolutional kernel size is set as $d_{model} \times 1 \times 1 \times 1$.

Branch 1 + Branch 2: As shown in Fig. 5.3 the output \mathbf{Z}_t^{out1} from Branch 1 and the output from \mathbf{Z}_t^{out2} from Branch 2 are added together to form the output of one 3DTransLSTM block, $\overline{\mathbf{Out}}_t \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$. Six such 3DTransLSTM blocks are stacked to generate the final output $\hat{\mathbf{Z}}_{t+1} \in \mathbb{R}^{d_{model} \times L \times H/p \times W/p}$.

Prediction Head

As shown in Fig. 5.2, this final output of the 3DTransLSTM blocks $\hat{\mathbf{Z}}_{t+1}$ is processed by a prediction head using pointwise convolution $W_{\hat{\mathbf{Z}}}$ with Cp^2 filters to output the predicted frame patches $\hat{\mathbf{P}}_{t+1} \in \mathbb{R}^{Cp^2 \times L \times H/p \times W/p}$ as

$$\hat{\mathbf{P}}_{t+1} = W_{\hat{\mathbf{Z}}}^{Cp^2 \times 1 \times 1 \times 1} * \hat{\mathbf{Z}}_{t+1} \tag{5.12}$$

These frame patches are reshaped to generate the L predicted frames $\hat{\mathbf{X}}_{t+1} \in \mathbb{R}^{C \times L \times H \times W}$ with frame indices $t + 1 : t + L$.

5.4 Training and Inference Strategy

As described in Section 5.3, in each time step, our proposed network takes L frames as the input (for example, frame t to frame $t + L - 1$), and generates L predicted frames (for example, frame $t + 1$ to frame $t + L$), with one-time index shift. It is noteworthy that the video sequence length L can be different during the training and inference stages, and L can also vary during the inference stage, while the dimensions of the trained filters are fixed.

We trained our model *Net* to predict the next L frames $\hat{\mathbf{X}}_{t+1} = \{\hat{\mathbf{Y}}_{t+1}, \hat{\mathbf{Y}}_{t+2}, \dots, \hat{\mathbf{Y}}_{t+L}\} \in \mathbb{R}^{C \times L \times H \times W}$ by learning from the previous L frames $\mathbf{X}_t = \{\mathbf{Y}_t, \mathbf{Y}_{t+1}, \dots, \mathbf{Y}_{t+L-1}\}$ in three successive iterations. In iteration 1, \mathbf{X}_t is used to predict $\hat{\mathbf{X}}_{t+1}$. In iteration 2, \mathbf{X}_{t+1} is used to predict $\hat{\mathbf{X}}_{t+2} \in \mathbb{R}^{C \times L \times H \times W}$. In iteration 3, \mathbf{X}_{t+2} is used to predict $\hat{\mathbf{X}}_{t+3} \in \mathbb{R}^{C \times L \times H \times W}$. Since there are three iterations and each iteration has L predicted frames, the MSE loss used to train the network is averaged over $3L$ predicted frames.

During inference, the model takes the previous K frames $\mathbf{X}_t = \{\mathbf{Y}_t, \mathbf{Y}_{t+1}, \dots, \mathbf{Y}_{t+K-1}\} \in \mathbb{R}^{C \times K \times H \times W}$ as the input, and predicts the future N frames $\hat{\mathbf{X}}_{t+K} = \{\hat{\mathbf{Y}}_{t+K}, \hat{\mathbf{Y}}_{t+K+1}, \dots, \hat{\mathbf{Y}}_{t+K+N-1}\} \in \mathbb{R}^{C \times N \times H \times W}$. The inference process is given in Algorithm 1.

Algorithm 1: The Inference Process

Input: $\mathbf{X}_t = \{\mathbf{Y}_t, \mathbf{Y}_{t+1}, \dots, \mathbf{Y}_{t+K-1}\}$
Output: $\hat{\mathbf{X}}_{t+K} = \{\hat{\mathbf{Y}}_{t+K}, \hat{\mathbf{Y}}_{t+K+1}, \dots, \hat{\mathbf{Y}}_{t+K+N-1}\}$

$\mathbf{X}_{in} = \mathbf{X}_t$
for $i \in 0$ *to* $N - 1$ **do**
 $\{\hat{\mathbf{Y}}_{t+1}, \hat{\mathbf{Y}}_{t+2}, \dots, \hat{\mathbf{Y}}_{t+K+i}\} \leftarrow \text{Net}(\mathbf{X}_{in})$
 $\mathbf{X}_{in} \leftarrow \text{Concat}(\mathbf{X}_{in}, \hat{\mathbf{Y}}_{t+K+i})$
end
 $\hat{\mathbf{X}}_{t+K} \leftarrow \{\hat{\mathbf{Y}}_{t+K}, \hat{\mathbf{Y}}_{t+K+1}, \dots, \hat{\mathbf{Y}}_{t+K+N-1}\}$
return $\hat{\mathbf{X}}_{t+K}$

As described in Algorithm 1, the inference process consists of N iterations. In i -th iteration, the last predicted frame $\hat{\mathbf{Y}}_{t+K+i}$ is concatenated with the current input video sequence to form a new input sequence $\mathbf{X}_{in} \in \mathbb{R}^{C \times (K+i+1) \times H \times W}$ of $K + i + 1$ frames. After N iterations, a total of N future frames are predicted, denoted as $\hat{\mathbf{X}}_{t+K} = \{\hat{\mathbf{Y}}_{t+K}, \hat{\mathbf{Y}}_{t+K+1}, \dots, \hat{\mathbf{Y}}_{t+K+N-1}\}$.

The input and output sequence length L during training, the input video frame number K and the predicted future frame number N during testing are provided for each dataset in the following section 5.5.

5.5 Datasets

We conducted experimental studies on the synthetic MovingMNIST (MMNIST) dataset [21] and real-world datasets KTH Action [53], TaxiBJ [62], and Human 3.6m [54]. To evaluate the generalization ability of the proposed method, we also trained the model on the KITTI [55] dataset and tested it on an unseen Caltech Pedestrian [56] dataset.

5.5.1 Moving MNIST (MMNIST)

Section 4.4.1 briefly introduces MMNIST dataset. Following TCTN [41], we used two subsets of this dataset for training: MMNIST-2K with 2,000 video sequences and MMNIST-10K with 10,000 video sequences. Each sequence has 20 frames. We trained one model on each subset and tested both trained models on another unseen MMNIST subset of 3,000 video sequences (MMNIST-3K). During training, the input and output sequence lengths were both $L = 17$. During testing, we used $K = 10$ previous frames to predict $N = 10$ future frames.

5.5.2 KTH Action

Details of the dataset are provided in Section 4.4.2. The training set contained 8,488 sequences and the testing set had 5,041 sequences. Each sequence had 20 frames, and each frame was resized to $1 \times 64 \times 64$, where the first dimension represented the grayscale channel. Similar to the MMNIST dataset, during training, the input and output sequence length was $L = 17$ frames. During testing, the previous $K = 10$ frames were used to predict $N = 10$ future frames.

5.5.3 TaxiBJ

This dataset represents the trajectory data derived from the GPS information of more than 34,000 taxicabs throughout Beijing and meteorology data collected over 16 discrete months, from June 2013 to April 2016. The city of Beijing is divided to 32×32 grids and the traffic flow information is reported in a time interval of 30 minutes, which generates 20,904 traffic heat maps of size $2 \times 32 \times 32$. The first dimension represents two channels

for the inflow and outflow traffic information. Following [27], the data of the last date interval was chosen for testing and the rest was used for training. The training dataset has 19,560 sequences and the testing dataset has 1,344 sequences. Each sequence has 8 frames. During training, the input and output sequence lengths were both $L = 5$. The testing was done using $K = 4$ input frames to predict the future $N = 4$ frames.

5.5.4 Human 3.6m

This is a complex human pose action dataset, originally with 3.6 million RGB images. It comprises of video sequences with humans performing different types of actions. Each sequence has 8 frames. We followed the experimental setup in [27] and chose only the ‘walking’ scenario. The original $3 \times 1000 \times 1000$ resolution frames were resized to $3 \times 128 \times 128$ resolution in our experiments, where the first dimension represents the RGB channels. Subjects S1, S5, S6, S7, S8 were used for training which made to 2,624 sequences and subjects S9, S11 were used for testing with 1,135 sequences. For training we set the input and output sequence length as $L = 5$ and for testing we used the previous $K = 4$ frames to predict the future $N = 4$ frames.

5.5.5 KITTI and Caltech Pedestrian

The KITTI and Caltech Pedestrian datasets are two car-mounted camera video datasets with real-world scenarios, widely used for video frame prediction. We followed [32] to preprocess the datasets. The frames from both datasets were center cropped and resized to $3 \times 128 \times 160$, where the first dimension represents the RGB channels. The 30 fps frame rate of Caltech was changed to 10 fps to match the frame rate of the KITTI dataset. The

KITTI dataset was used for model training. It consists of 3,150 sequences and each sequence has 13 frames. A subset of the Caltech dataset was used for model testing, which had 1,983 sequences and each sequence had 11 frames. We followed [57] to preprocess the datasets. The frames from both datasets were center-cropped and resized to $3 \times 128 \times 160$, where the first dimension represented the RGB channels. During training, we set the input and output sequence length to be $L = 10$. During testing, the previous $K = 10$ frames were used to predict the next $N = 1$ frames.

5.6 Experiment Settings and Methods in Comparison

The models were trained with the PyTorch framework using an NVIDIA Tesla V100 32 GB GPU. The ADAM optimizer was used to minimize the MSE loss between the ground truth and the predicted frames. To prevent overfitting, dropout layers of the proposed network adopt a dropout rate of 0.05 during training. The model was trained for 200 epochs for MMNIST and KTH Action, and 300 epochs for Human 3.6m and KITTI. We set the patch size as $p = 4$ and the hidden dimension as $d_{model} = 128$ for MMNIST, KTH Action, and Human 3.6m. For KITTI dataset, we set the patch size as $p = 2$ and the hidden dimension as $d_{model} = 256$. We used a stride of 1 for the depthwise separable convolutions in our network.

We compared the performance of our proposed model to seven existing methods. They included three transformer-based models: ConvTransformer [40], TCTN [41], and VPTR [42], and four RNN-based models: FRNN [20], E3D-LSTM [23], MIM [25], and PredRNN-V2 [27]. All the models were implemented with the same experiment settings proposed in their original works. To verify the generalization ability of our proposed

model, we chose the following existing methods for comparison studies: the ConvLSTM-based models DNA [22], CrevNet [26], the GAN-based DM-GAN [35], the transformer-based VPTR [42], as well as the CNN-based BeyondMSE [34].

5.7 Experimental Results and Analysis

5.7.1 MMNIST

Though the dynamics of MMNIST seem simple, the frequent occlusion and overlapping of the digits make the prediction complex. Table 5.2 compares the frame prediction accuracy performance between the proposed model and existing models on the MMNIST-3K test set after the models were trained on the MMNIST-2K and MMNIST-10K datasets. The best, second-best, and third-best results of each metric are highlighted in red, blue and brown, respectively. Our proposed model 3DTransLSTM achieved the highest PSNR and SSIM scores when it was trained on MMNIST-2K, and it achieved the second highest PSNR and SSIM scores when it was trained on MMNIST-10K. It is superior in terms of model size and parameters, and it requires the second fewest GFLOPs for inference. When the models were trained on MMNIST-10K, PredRNN-V2 [27] achieved the highest PSNR, but our proposed model has 39.2% decrease in model size and 51.5% decrease in parameters, compared to PredRNN-V2. To visually demonstrate the effectiveness of our proposed model, Fig. 5.6 shows the qualitative results of models trained on MMNIST-10K. We observe that ConvTransformer [40] and VPTR [42] generated blurry results and E3D-LSTM [23] failed to correctly predict the relative positions of the digits. The results of our proposed 3DTransLSTM model correctly captured the trajectory of moving digits and

looked similar to the ground truth without blurriness. MIM [25], PredRNN- V2 [27], and TCTN [41] produced good visual results at the expense of huge model size, parameters and computational complexity compared to our proposed model.



Fig. 5.6. The visual results on the MMNIST-3K dataset for models trained on the MMNIST-10K dataset.

5.7.2 KTH Action

The KTH Action dataset has a similar frame size (64×64) and input/output sequence length as the MMNIST dataset. The quantitative results are also summarized in Table 5.2.

Table 5.2 Quantitative results on MMNIST-3K and KTH Action datasets. The best, second-best, and third-best results of each metric are highlighted in red, blue and brown, respectively.

Methods	MMNIST-3K (Trained on MMNIST-2K)		MMNIST-3K (Trained on MMNIST- 10K)		KTH Action	Model size ↓ (MB)	#Params ↓ (M)	FLOPs ↓ (G)
	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑				
VPTR [42]	19.12	0.854	19.40	0.877	28.93	1988.2	167.9	201.8
TCTN [41]	20.45	0.886	22.66	0.918	25.38	344.1	31.9	511.3
MIM [25]	20.78	0.882	23.14	0.927	28.96	211.9	41.6	795.2
E3D-LSTM [23]	18.83	0.843	20.86	0.89	27.15	201.5	38.7	253.6
PredRNN-V2 [27]	20.52	0.891	23.78	0.921	28.56	93.1	23.9	116.6
FRNN [20]	17.89	0.824	19.23	0.837	26.12	89.2	-	-
ConvTransformer [40]	17.68	0.833	19.41	0.810	27.61	81.0	20.2	228.0
3DTransLSTM	21.04	0.893	23.57	0.921	28.78	56.6	11.6286	140.1

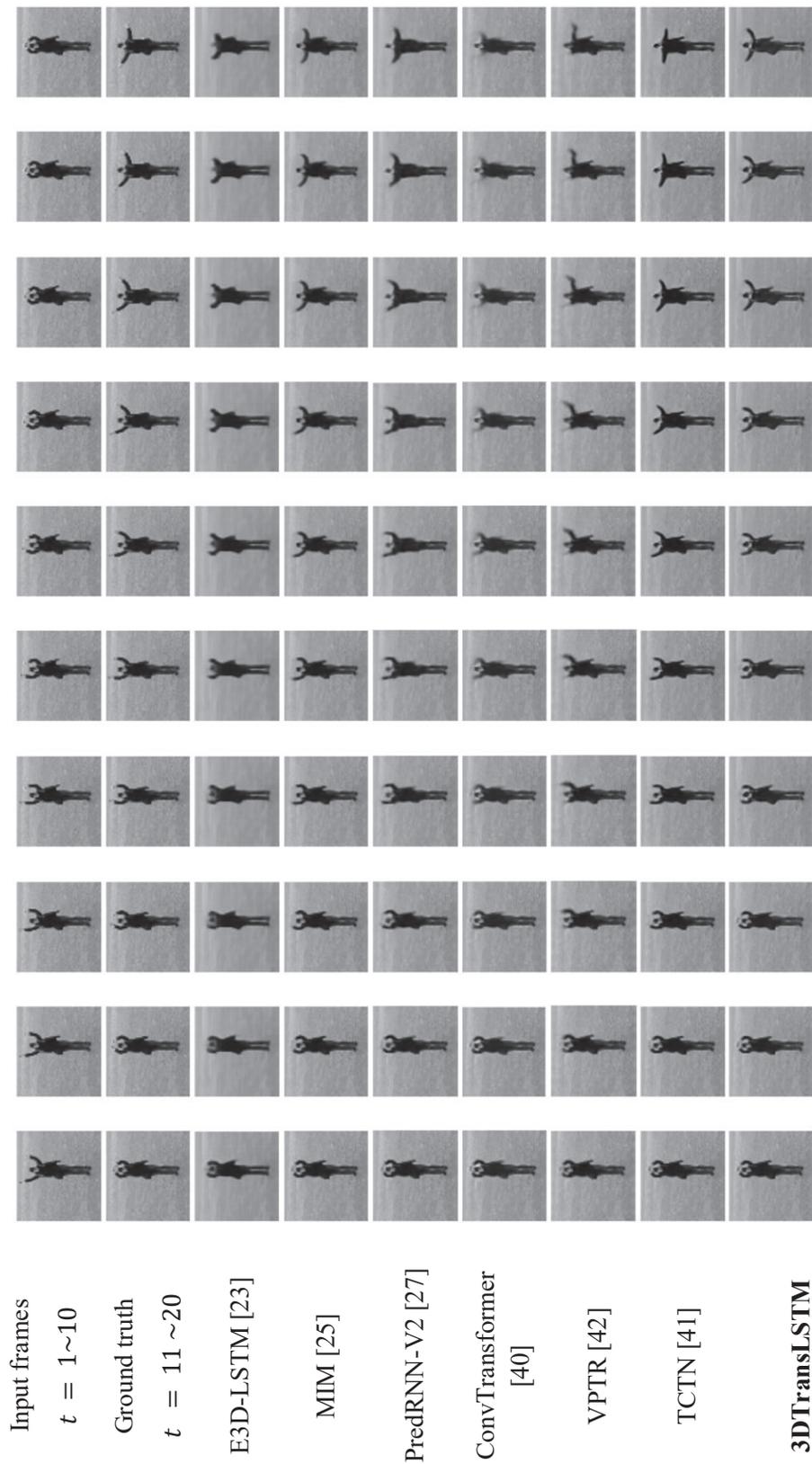


Fig. 5.7. The visual results on the KTH Action dataset

Obviously, the proposed 3DTransLSTM model achieved the third highest PSNR and SSIM values. Although the MIM [25] and VPTR [42] models achieved slightly higher PSNR and SSIM values than 3DTransLSTM, their model sizes were much larger, and they required much more parameters and GFLOPs. Fig. 5.7 shows the predicted frames of different models. We observe that the proposed 3DTransLSTM model predicted the motion of the hands correctly and it preserved the structure of the person. In contrast, the transformer-based models VPTR [42], ConvTransformer [40], and the ConvLSTM-based model E3D-LSTM [23] generated blurry results and had missing arms. The visual results of MIM [25], TCTN [41], and PredRNN-V2 [27] are close to the proposed 3DTransLSTM, but they require much more computational cost.

5.7.3 TaxiBJ

This is a challenging dataset as the traffic flows are not uniformly varied over time and there are external factors such as weather which affects the traffic conditions [23]. We followed the experimental settings in [23] and predicted the next 4 frames, given the previous 4 frames. It can be observed from the quantitative results provided in Table 5.3 that 3DTransLSTM shows the best model efficiency compared to the SOTA. methods, in terms of the smallest model size, fewest model parameters, and fewest FLOPs. Besides, it achieves the highest PSNR and the second-highest SSIM. Although E3D-LSTM [23] achieves the highest SSIM, compared to 3DTransLSTM, its model size is 3.61 times larger, its parameters are 3.33 times more, and it requires 3.4 times more FLOPs to performance inference. Fig. 5.8 shows the visual comparisons of the predicted flows of the different methods. High color intensity indicates higher traffic flow. We can infer that unlike the pure RNN models like E3D-LSTM [23] where the predicted frames are more similar to the

past input frames, our model’s predictions look very similar to the target future frames. The performance on the TaxiBJ dataset indeed demonstrates that the proposed 3DTransLSTM is capable of predicting real-world scenarios as well.

Table 5.3 Quantitative results on TaxiBJ. The best, second-best, and third-best results of each metric are highlighted in red, blue and brown, respectively.

(4 → 4 FRAMES)					
Methods	PSNR ↑	SSIM↑	Model Size (MB)	Params (M)	FLOPs (G)
VPTR [42]	37.97	0.961	1300	161.24	12.74
TCTN [41]	39.6	0.963	345.4	32	30.75
E3D-LSTM [23]	39.9	0.979	202.3	39.7	23.95
MIM [25]	38.31	0.971	152.2	28.53	48.39
PredRNN-V2 [27]	37.32	0.969	94.7	24.2	11.4
ConvTransformer [40]	38.5	0.961	81	20.16	57
3DTransLSTM	41.4	0.971	56.1	11.6335	6.94

5.7.4 Human3.6m

This human pose dataset has video frames of dimension $3 \times 128 \times 128$, where the first dimension represents the RGB channels. Our proposed model predicted the future four frames based on previous four frames. Table 5.4 summarizes the quantitative results of the compared models. The proposed 3DTransLSTM achieves the smallest model size, fewest number of parameters and GFLOPs. In terms of prediction accuracy, it achieves the second highest PSNR and SSIM values. Although VPTR [42] achieves the highest frame prediction accuracy, it is 44 times the size of our model, and it requires 17 times more parameters than our model. Fig. 5.9 shows the visual quality of the predicted frames. We

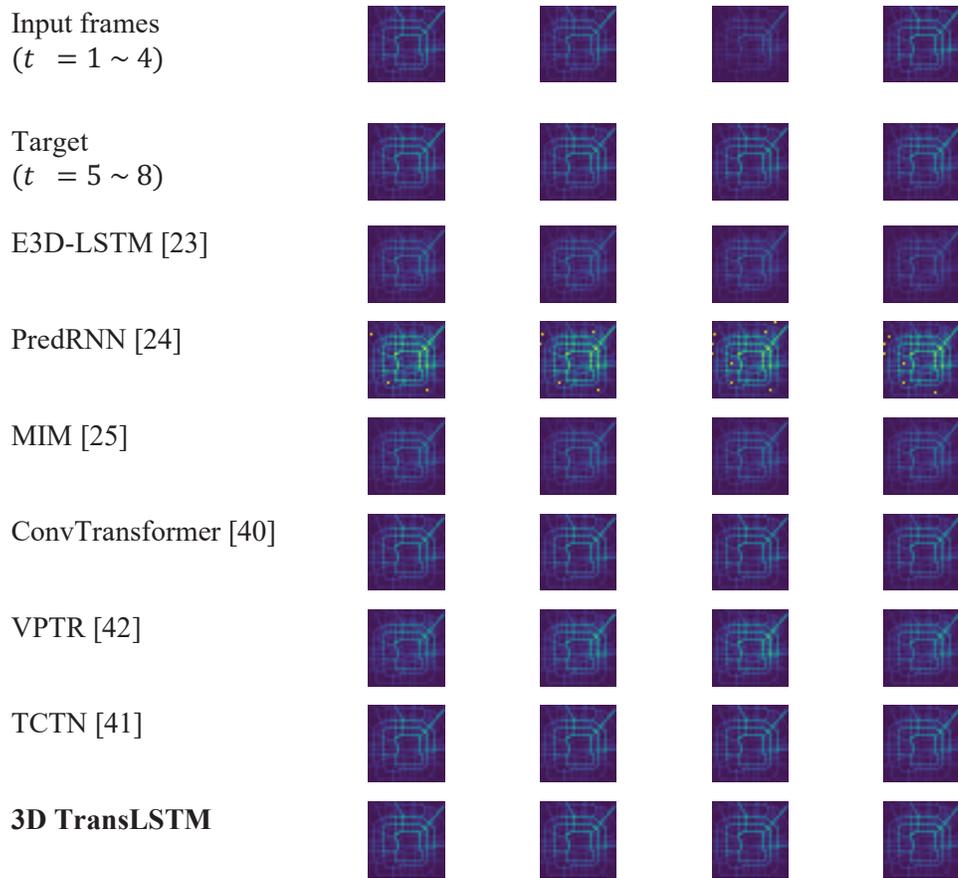


Fig. 5.8. Visual results on the TaxiBJ

observe that the proposed 3DTransLSTM model is able to preserve the structure of the person's body and shape of arms. In contrast, the solely ConvLSTM-based models E3D-LSTM [23], MIM [25] and PredRNN-V2 [27] produced vague results and had inconsistencies in the physical appearance of the person, especially the shape of arms. Besides, the transformer-based model TCTN [41] also generated blurry results. The quantitative and qualitative results show that our model works effectively for RGB video frames.

Table 5.4 Quantitative Results on Human 3.6m Dataset. The best, second-best, and third-best results of each metric are highlighted in red, blue and brown, respectively.

Methods	<i>PSNR</i>	<i>SSIM</i>	Model size (MB)	Params (M)	FLOPs (G)
VPTR [42]	29.31	0.893	2513.9	198.8	180.9
TCTN [41]	27.63	0.839	344	32.1	492.4
E3D-LSTM [23]*	19.79	0.73	263.5	40.9	395.7
MIM [25]*	21.80	0.790	196.2	41.9	775.2
PredRNN-V2 [27]	20.7	0.790	156	24.6	176.3
FRNN [20]*	21.16	0.771	109.2	-	-
ConvTransformer[40]	20.97	0.798	81.0	20.1	228.7
3DTransLSTM	28.6	0.875	56.2	11.6	107.0

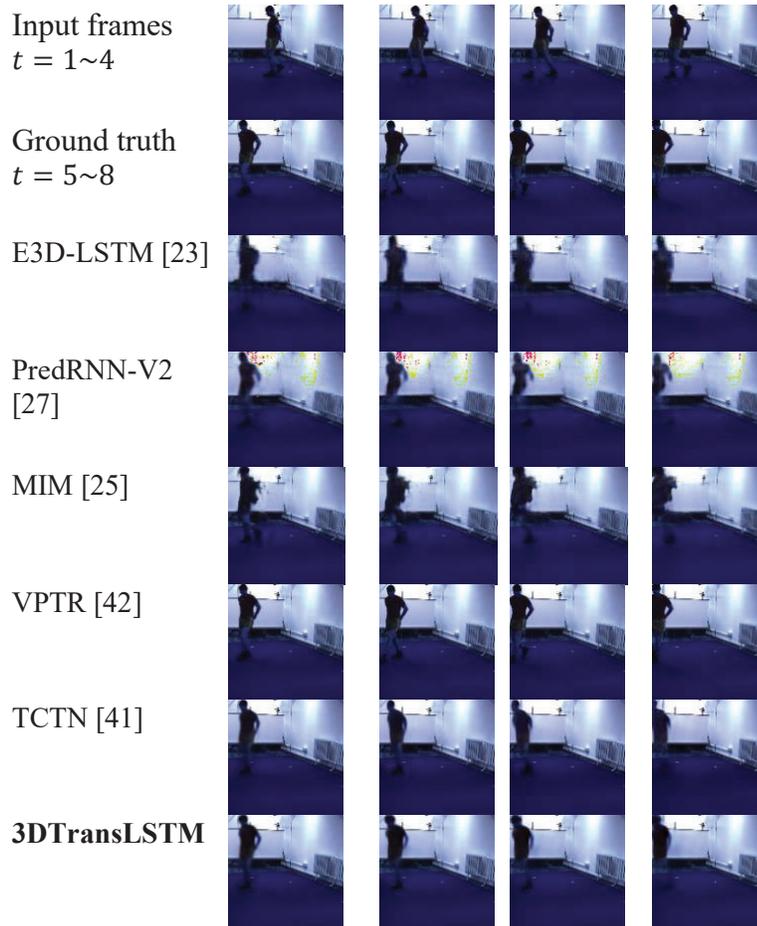


Fig. 5.9. Visual results on the Human3.6m dataset

5.7.5 KITTI and Caltech Pedestrian

To evaluate the generalization ability of video prediction models, we trained the models on the KITTI dataset [55] using the previous 10 frames to predict the next one frame and evaluated the trained models on the Caltech Pedestrian dataset [56]. Both the KITTI and Caltech Pedestrian datasets are complex datasets comprised of real-world scenarios with multiple moving objects in the background. We observe from Table 5.5 that the proposed 3DTransLSTM achieved the best model efficiency in terms of model size, number of parameters, and GFLOPs. Besides, it achieves the third highest PSNR and SSIM values.

Although CrevNet [26] has the highest PSNR and SSIM values, it required much larger model size, much more parameters and GFLOPs than our model did. Fig. 5.10 shows that our proposed model generated accurate visual results similar to CrevNet [26] and VPTR [42], producing clear objects and texts in the predicted frames.

Table 5.5 Quantitative results on Caltech Pedestrian dataset. * The results are reported in [18]. † The results are reported in [59]. The best, second-best, and third-best results of each metric are highlighted in red, blue and brown, respectively.

Methods	<i>PSNR</i>	<i>SSIM</i>	Model size (MB)	Params (M)	FLOPs (G)
BeyondMSE [34] [†]	24.87	0.881	218.4	-	-
DNA [22] [†]	-	0.896	-	>160	-
STN [65] [†]	-	0.902	-	>160	-
DMGAN* [58]	26.29	0.899	-	113	-
CrevNet* [26]	29.25	0.925	2700	-	350.4
VPTR [42]	29.12	0.921	2513.9	208.69	201.56
3DTransLSTM	28.81	0.915	207.2	43.05	55.94

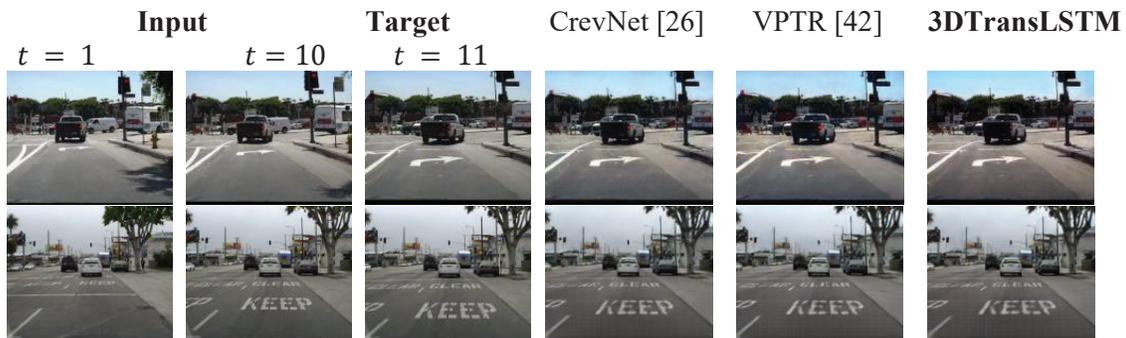


Fig. 5.10. Visual results on the Caltech Pedestrian dataset. Input is previous 10 frames ($t = 1 \sim 10$) and the output is frame at $t = 11$

5.8 Ablation Study

We conducted a series of experiments with various designs of the proposed TransLSTM network to validate the advantages of using 3D separable convolutions. The models were trained on the MMNIST-2K dataset and tested on the MMNIST-3K dataset. The results are summarized in Table 5.6.

First, we designed Model A, which is the TransLSTM network with traditional 2D convolutions in the spatial embedding module, the self-attention and feed-forward layers of Branch 1, and in the ST-LSTM layers of Branch 2. It achieves an average PSNR of 19.44 dB and an average SSIM of 0.844. Its model size is 745 MB. It has 78.7 M trainable parameters, and it requires 934.7 GFLOPs to conduct inference.

To reduce the model size, parameters, and computational complexity of Model A, we designed Model B, which is the TransLSTM network with 2D separable convolutions. It apparently reduces the model size, parameters, and computation complexity to 50.9 MB,

Table 5.6. Ablation study on MMNIST. The best and second-best results of each metric are highlighted in red and blue, respectively.

Model	Method	MMNIST-3K (Trained on MMNIST-2K)		Model size (MB)	Para ms (M)	FLOPs (G)
		<i>PSNR</i>	<i>SSIM</i>			
A	3DTransLSTM (traditional 2D)	19.44	0.844	745	78.72	934.73
B	3DTransLSTM (2D separable)	18.03	0.791	50.9	10.5	126.1
C	3DTransLSTM (traditional 3D)	21.48	0.913	1100	231.49	2914.3 1
	3DTransLSTM	21.03	0.893	56.6	11.6	140.1

10.5 M, and 126.09 GFLOPs, respectively. Nevertheless, the video frame prediction accuracy has decreased to an average PSNR of 18.03 dB and an average SSIM of 0.791.

To leverage temporal information in the video sequence, we further designed Model C, the TransLSTM with traditional 3D convolutions in self-attention and feed-forward layers of Branch 1 and ST-LSTM layers of Branch 2. The results in Table 5.6 showed that the prediction accuracy is significantly improved compared to traditional 2D convolutions (Model A), achieving an average PSNR of 21.48 dB and an average SSIM of 0.913. However, this comes at a price of dramatically increased model size (1,100 MB), number of trainable parameters (231.5 M), and GFLOPs (2,914.3).

In contrast, our proposed model, 3DTransLSTM with 3D separable convolutions, significantly reduces the model size, number of parameters, and GFLOPs by 94.9%, 95%,

and 95.2%, respectively, compared to Model C. Meanwhile, it only slightly decreased the prediction accuracy by 2.1% for the PSNR metric and by 2.2% for the SSIM metric.

The above ablation studies thoroughly validate the benefits of the proposed 3DTransLSTM with 3D separable convolutions. This network architecture not only leverages the spatiotemporal correlations in the video to provide competitive frame prediction accuracy, but also significantly reduces model size, trainable parameters and computational complexity compared to traditional 3D convolutions. Therefore, it offers a good trade-off between model accuracy and model complexity.

5.9 Conclusion

In this chapter, we present a novel video prediction framework 3DTransLSTM, incorporating both transformer and LSTM structures with 3D separable convolutions. Extensive experimental results demonstrate the effectiveness of our proposed scheme on both synthetic and real-world datasets. Compared to existing approaches, our method is able to achieve competitive prediction accuracy with significantly reduced model size, number of parameters, and computational complexity. Hence, the results suggest that our model could be better suited for memory-constrained and computation resource- limited platforms, such as mobile and embedded devices.

CHAPTER 6

Conclusion

6.1 Summary of Major Contributions

Video frame prediction is a challenging yet essential vision task in various real-world scenarios. which has demonstrated its applications in various spectrum of fields, from video coding to autonomous driving. Deep learning community has vastly explored computationally expensive models in the past few decades for improving model accuracy and surpassing state-of-the art results. In this era of Green AI [66], it is extremely important for machine learning models to offer both model efficiency and accuracy. Apart from such Red AI (Artificial Intelligence)-focused models, we focus on such Green AI methods by developing environment friendly solutions with reduced model complexity and competitive accuracy for video prediction. Extensive experimental results demonstrate the effectiveness of our proposed scheme on both synthetic and real-world datasets, as well as its generalization capability. Our model is more suitable for memory-constrained and computation resource- limited platforms, such as mobile, IoT and embedded devices.

Addressing the research questions outlined in Chapter 1, our study yield significant insights and contributions, as mentioned below.

Efficiency and Energy Reduction:

- The proposed models adopted 3D convolutions to effectively learn spatiotemporal dynamics. To reduce computational cost, the standard 3D convolution is

decomposed into a 3D depthwise convolution and a pointwise convolution, which reduces the model size, trainable parameters, and floating-point operations (FLOPs).

Architectural Combinations:

- We propose a lightweight deep networking model with 3D separable convolutions for the video prediction. Spatiotemporal long short-term memory (ST-LSTM) based on 3D separable convolutions is proposed for the first time in literature for the task of video prediction. Our model based achieved more than 90% reduction in model size, FLOPs, and number of model parameters when compared to several SOTA methods, on both synthetic and real-world, grayscale and colored datasets.
- We propose a hybrid transformer-LSTM (3DTransLSTM) network for the video prediction task. On the one hand, the transformer module can leverage long-range correlations among multiple successive video frames and parallelize the computation with its self-attention mechanism. On the other hand, the LSTM module can enable spatiotemporal information flow vertically within each time step and horizontally among multiple time steps. Such a hybrid network consisting of transformer and LSTM is proposed in video prediction task for the first time in literature. Moreover, to the best of our knowledge, this is the first time that 3D separable convolutions is utilized in a hybrid transformer- LSTM network.

Trade-offs Between Efficiency and Accuracy:

- Qualitative and quantitative experimental results on popular video prediction datasets show that, compared to SOTA methods using ConvLSTM-based

approaches and transformer-based approaches, the proposed 3DTransLSTM achieves competitive video frame prediction accuracy with significantly smaller model size, fewer model parameters, and less computational cost.

Real-world Applications:

- We demonstrate the effectiveness and generalization ability of the proposed model by testing the model on video sequences completely unseen in the training dataset. Additionally, we demonstrate the applicability of the method in color and grayscale videos, traffic flow prediction dataset and real-world pedestrian dataset.
- The promising result of our models suggests a viable direction for memory-constrained and embedded devices.

Despite these contributions, our research also faces certain limitations. For instance, usage of our models for critical applications such as autonomous vehicles, may pose challenges, as model's accuracy is paramount in such scenarios. Additionally, while our models demonstrate effectiveness across synthetic and real-world datasets, further validation on larger and diverse datasets could provide deeper insights into the scalability and robustness of our approach.

6.2 Future Research

In this work, we demonstrate the 3D separable based LSTM and hybrid transformer LSTM models for efficient video prediction. We also demonstrate the potential of integrating other low memory-consuming networks such as reversible networks (Chapter 4) in conjunction with our algorithm to improve the model efficiency.

As emphasized in Chapter 1, the applications of video prediction are vast and in order to adapt the task for many real-world scenarios, it's crucial that the deep learning models need to keep improving the efficiency, while delivering accuracy. In future, we intend to upgrade our model by incorporating other efficient lightweight architectures and thereby achieve better prediction performance, while maintaining the lower number of parameters.

Since our method is more suitable for resource-constrained environments, we can leverage our lightweight architecture for the new field of 'Video Coding for Machines (VCM)'. It's memory efficiency without much compromising the accuracy can focus on regions of interests with significant motion and optimize resource usage for applications for machines.

The role of video prediction in video coding is crucial, where video prediction aims to achieve better compression by exploiting temporal and spatial redundancies. Particularly, inter-frame prediction, such bi-prediction where prediction is based on both preceding and subsequent frames plays an indispensable role in current video codecs such as HEVC. Novel deep learning methods for bi-prediction [14] were proposed to obtain accurate motion prediction and better coding performance. We plan to adopt our method in bi-directional way to include it in video coding framework. Replacing unidirectional prediction with bi-directional prediction can also reduce prediction errors, since it considers both past and current frames. This method can also be extended to frame interpolation, where intermediate frames are synthesized using neighboring frames which facilitates smooth motion in frames.

Another possibility is that we can extend our methods to integrate with other optical-flow synthesizer methods. Optical flow techniques allow to estimate motion between

consecutive frames and thus can enhance the motion content in frames, thus leading to more realistic predictions. Synthesizing optical flow between frames can provide a better vision to the models to accurately understand the motion patterns present in video, leading to better predictions.

BIBLIOGRAPHY

- [1] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, M. and A. Sorkine-Hornung, A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 724-732). 2016.
- [2] H.Choi and I.V.Bajic,“Deep frame prediction for video coding,” *IEEE Transactions on Circuits Systems and Video Technology*, vol. 30, no. 7, pp. 1843–1855, July 2019.
- [3] B.Liu, Y.Chen, S.Liu,and H.S.Kim,“Deep learning in latent space for video prediction and compression,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition*, Nashville, TN, USA, Jun. 2021, pp. 701–710.
- [4] Y.Zhou, H.Dong, and A.ElSaddik, “ Deep learning in next-frame prediction: A benchmark review,” *IEEE Access*, vol. 8, pp. 69 273–69 283, 2020.
- [5] S. Oprea, P. Martinez-Gonzalez, A. Garcia-Garcia, J. A. Castro-Vargas, S. Orts-Escolano, J. Garcia-Rodriguez, and A. Argyros, “A review on deep learning techniques for video prediction,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 6, pp. 2806–2826, Jun. 2022.
- [6] Yu, Q. Kuang, and R. Yang, “ATMConvGRU for weather forecasting,” *IEEE Geoscience and Remote Sensing Letters*, 2021.
- [7] L. Liu, J. Zhen, G. Li, G. Zhan, Z. He, B. Du and L. Lin, “Dynamic spatial temporal representation learning for traffic flow prediction,” *IEEE Transactions on Intelligent Transportation Systems*, doi: 10.1109/TITS.2020.3002718, 2020.

- [8] W. Liu, W. Luo, D. Lian, and S. Gao, “Future frame prediction for anomaly detection—a new baseline,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 6536–6545.
- [9] X. Lin, Q. Zou, X. Xu, Y. Huang, and Y. Tian, “Motion-aware feature enhancement network for video prediction,” *IEEE Transactions on Circuits Systems and Video Technology*, vol. 31, no. 2, pp. 688–700, 2020.
- [10] Li S, J. Fang, H. Xu, and J. Xue, “Video frame prediction by deep multi-branch mask network,” *IEEE Transactions on Circuits Systems and Video Technology*, vol. 31, no. 4, pp. 1283–1295, 2020.
- [11] V. Vukotić, S. L. Pintea, C. Raymond, G. Gravier, and J. C. V. Gemert, “One-step time-dependent future video frame prediction with a convolutional encoder-decoder neural network,” in *Proc. Int. Conf. Image Anal. Process.* Catania, Italy: Springer, Sep. 2017, pp. 140–151.
- [12] M. A. Yılmaz and A. M. Tekalp, “DFPN: Deformable frame prediction network,” in *IEEE Int. Conf. Image Process.*, Anchorage, AK, USA, Sept. 2021, pp. 1944–1948.
- [13] J.-K. Lee, N. Kim, S. Cho, and J.-W. Kang, “Deep video prediction network-based inter frame coding in HEVC,” *IEEE Access*, vol. 8, pp. 95 906–95 917, 2020.
- [14] R. Yang, R. Timofte, and L. V. Gool, “Advancing learned video compression with in-loop frame prediction,” *IEEE Transactions on Circuits Systems and Video Technology*, vol. 33, no. 5, pp. 2410–2423, 2023.
- [15] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard,” *IEEE Transactions on Circuits Systems and Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

- [16] B. Bross, Y.-K. Wang, S. Liu Y. Ye, J. Chen, G. J. Sullivan, and J.-R. Ohm, “Overview of the versatile video coding (VVC) standard and its applications,” *IEEE Transactions on Circuits Systems and Video Technology*, vol. 31, no. 10, pp. 3736–3764, 2021.
- [17] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M. H. Yang, “Flow- grounded spatial temporal video prediction from still images,” in *Proc. European Conf. Comput. Vis.*, Munich, Germany, Sept. 2018, pp. 600– 615.
- [18] F. A. Reda, G. Liu, K. J. Shih, R. Kirby, J. Barker, D. Tarjan, A. Tao A, and B. Catanzaro, “SDC-Net: Video prediction using spatially-displaced convolution,” in *Proc. European Conf. Comput. Vis.*, Munich, Germany, Sept. 2018, pp. 718–733.
- [19] N. Srivastava, E. Mansimov, and R. Salakhudinov, “Unsupervised learning of video representations using LSTMs,” in *Proc. PMLR Int. Conf. Mach. Learn.*, Lille, France, Jul. 2015, vol. 37, pp. 843–852.
- [20] M. Oliu, J. Selva, and S. Escalera, “Folded recurrent neural networks for future video prediction,” in *Proc. European Conf. Comput. Vis.*, Munich, Germany, Sept. 2018, pp. 716–731.
- [21] 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 *Proc. Int. Conf. Neural Inf. Process. Syst*, Montreal, Quebec, Canada, Dec. 2015, vol. 28, pp. 802—810.
- [22] C. Finn, I. Goodfellow, and S. Levine, “Unsupervised learning for physical interaction through video prediction,” in *Proc. Advances Neural Inf. Proc. Syst.*, Barcelona, Spain, Dec. 2016, vol. 29.

- [23] 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 *Proc. Int. Conf. Learn. Representations*, New Orleans, LA, USA, May 2019.
- [24] Y. Wang, M. Long, J. Wang, Z. Gao, and S. Y. Philip, “PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, vol. 30, pp. 879–888.
- [25] 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Long Beach, CA, USA, Jun. 2019, pp. 9154–9162.
- [26] W. Yu, Y. Lu, S. Easterbrook, and S. Fidler, “Efficient and information-preserving future frame prediction and beyond,” in *Proc. Int. Conf. Learn. Representations*, Virtual, Apr. 2020.
- [27] Y. Wang, H. Wu, J. Zhang, Z. Gao, J. Wang, S. Y. Philip, and M. Long, “PredRNN: A recurrent neural network for spatiotemporal predictive learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2208–2225, 2022.
- [28] M. Mathai, Y. Liu, and N. Ling, “A lightweight model with separable CNN and LSTM for video prediction,” in *IEEE Int. Symp. Circuits Syst.*, Austin, TX, USA, May 2022, pp. 516–520.
- [29] S. X. 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 *Proc. Advances Neural Inf. Proc. Syst.*, vol. 30, Long Beach, CA, USA, Dec. 2017, p. 5617–5627.

- [30] 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 *Proc. PMLR Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 5123–5132.
- [31] J. Su, W. Byeon, F. Huang, J. Kautz, and A. Anandkumar, “Convolutional tensor-train LSTM for spatio-temporal learning,” in *Proc. Int. Conf. Advances Neural Inf. Proc. Syst.*, vol. 33, Virtual, Dec. 2020, pp. 13 714– 13 726.
- [32] Z. Chang, X. Zhang, S. Wang, S. Ma, and W. Gao, “IPRNN: An information-preserving model for video prediction using spatiotemporal GRUs,” in *Proc. IEEE Int. Conf. Image Process.*, Anchorage, Alaska, USA, Sept. 2021, pp. 2703–2707.
- [33] Z. Chang, X. Zhang, S. Wang, S. Ma, Y. Ye, and W. Gao, “ASTM: An attention based spatiotemporal model for video prediction using 3D convolutional neural networks,” in *Proc. IEEE Int. Conf. Mult. Expo*, Shenzhen, China, Jul. 2021, pp. 1–6.
- [34] M. Mathieu, C. Couprie, and Y. LeCun, “Deep multi-scale video prediction beyond mean square error,” in *Proc. 4th Int. Conf. Learning Represent.*, San Juan, Puerto Rico, May 2016.
- [35] Z. Yi, H. Zhang, P. Tan, and M. Gong, “DualGAN: Unsupervised dual learning for image-to-image translation,” in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 2849–2857.
- [36] Y.H.Kwon and M.G.Park, “Predicting future frames using retrospective cycle GAN,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Long Beach, CA, USA, Jun. 2019, pp. 1811–1820.

- [37] Q.N.Tran and S.-H.Yang, “Attention-based inter-prediction for versatile video coding,” *IEEE Access*, 2023.
- [36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proc. Advances Neural Inf. Proc. Syst.*, Long Beach, CA, USA, Dec. 2017, vol. 30.
- [37] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *Proc. Eur. Conf. Comput. Vis.* Springer, 2020, pp. 213–229.
- [38] F. Yang, H. Yang, J. Fu, H. Lu, and B. Guo, “Learning texture transformer network for image super-resolution,” in *Proc. IEEE/CVF Int. Conf. on Comput. Vis.*, Seattle, WA, USA, Jun. 2020, pp. 5791–5800.
- [39] J. Li, W. Wang, J. Chen, L. Niu, J. Si, C. Qian, and L. Zhang, “Video semantic segmentation via sparse temporal transformer,” in *Proc. ACM Int. Conf. Multimedia*, Chengdu, China, Oct. 2021, pp. 59–68.
- [40] Z. Liu, S. Luo, W. Li, J. Lu, Y. Wu, S. Sun, C. Li, and L. Yang, “ConvTransformer: A convolutional transformer network for video frame synthesis,” *arXiv:2011.10185*, 2020.
- [41] Z.cYang, X. Yang, and Q. Lin, “TCTN: A 3D-temporal convolutional transformer network for spatiotemporal predictive learning,” *arXiv:2112.01085*, 2021.
- [42] X. Ye and G. A. Bilodeau, “VPTR: Efficient transformers for video prediction,” in *Proc. IEEE Int. Conf. Pattern Recognit.*, Montre´al, Que´bec, Canada, Aug. 2022, pp. 3492–3499.

- [43] W. Li, X. Wang, X. Xia, J. Wu, X. Xiao, M. Zheng, and S. Wen, “SepViT: Separable vision transformer,” *arXiv:2203.15380*, 2022.
- [44] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green AI,” *Commun. ACM*, vol. 63, no. 12, pp. 54–63, Nov. 2020.
- [45] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, “On the dangers of stochastic parrots: Can language models be too big?,” in *Proc. ACM Conf. Fairness Accountability Transparency*, Virtual Event, Canada, 2021, p. 610–623.
- [46] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. and Uszkoreit, J., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.
- [47] A.G.Howard,M.Zhu,B.Chen,D.Kalenichenko,W.Wang,T.Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” *arXiv:1704.04861*, 2017.
- [48] A. Pfeuffer and K. Dietmayer, “Separable convolutional LSTMs for faster video segmentation,” in *Proc. IEEE Intell. Transp. Syst. Conf.*, Auckland, New Zealand, Oct. 2019, pp. 1072–1078.
- [49] B. Hou and N. Ling Y. Liu, “A super-fast deep network for moving object detection,” in *Proc. IEEE Int. Symp. Circuits Syst.*, Seville, Spain, Oct. 2020, IEEE, pp. 1–5.
- [50] Z. Islam, M. Rukonuzzaman, R. Ahmed, M. H. Kabir, and M. Farazi, “Efficient two-stream network for violence detection using separable convolutional LSTM,” in *Proc. Int. Joint Conf. Neural Netw.*, Shenzhen, China, Jul. 2021, IEEE, pp. 1–8.

- [51] X. Zhang, Y. Tie, and L. Qi, “Dynamic gesture recognition based on 3d separable convolutional LSTM networks,” in *Proc. IEEE Int. Conf. Software Eng. Service Sci.*, Beijing, China, Oct. 2020, pp. 180–183.
- [52] B. Hou, Y. Liu, N. Ling, L. Liu, and Y. Ren, “A fast lightweight 3D separable convolutional neural network with multi-input multi-output for moving object detection,” *IEEE Access*, vol. 9, pp. 148433–148448, 2021.
- [53] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: a local svm approach,” in *Proc. 17th Int. Conf. Pattern Recognit.*, Cambridge, UK, Aug. 2004, vol. 3, pp. 32–36.
- [54] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, “Human3.6m: Large scale datasets and predictive methods for 3D human sensing in natural environments,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1325–1339, July 2014.
- [55] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [56] P. Dollár, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: A benchmark,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 304–311.
- [57] W. He, T. Xiong, H. Wang, J. He, X. Ren, Y. Yan, and L. Tan, “Radar echo spatiotemporal sequence prediction using an improved ConvGRU deep learning model,” *Atmosphere*, vol. 13, no. 1, pp. 88, 2022.
- [58] Y. Liu, P. Du, and Y. Li, “Hierarchical motion-compensated deep network for video compression,” in *Proc. SPIE 11730 Big Data III: Learn. Analytics Appl.*, Apr. 2021, vol. 11730, pp. 124–131.

- [59] Y. H. Ho, C. Y. Cho, and W. H. Peng, “Deep reinforcement learning for video prediction,” in *Proc. IEEE Int. Conf. Image Process.*, Taipei, Taiwan, Sept. 2019, pp. 604–608.
- [60] A.N, Gomez, M. Ren, R. Urtasun, and R.B.Grosse, The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017.
- [61] J. Jacobsen, A.W.M. Smeulders, and E. Oyallon, “i-RevNet: Deep invertible networks,” *International Conference on Learning Representations (ICLR)*, 2018.
- [62] J. Zhang, Y. Zheng, and D.Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.
- [63] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, “Visual foresight: Model-based deep reinforcement learning for vision-based robotic control,” *arXiv preprint arXiv:1812.00568*, 2018.
- [64] E. Denton, and R. Fergus, “Stochastic video generation with a learned prior,” *International Conference on Machine Learning*, pp. 1174-1183, 2018.
- [65] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al., “Spatial Transformer Networks,” in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2015.
- [66] Kuo, C.C.J. and Madni, A.M., 2023. Green learning: Introduction, examples and outlook. *Journal of Visual Communication and Image Representation*, 90, p.103685.

PUBLICATIONS

M. Mathai, Y. Liu and N. Ling, "A Hybrid Transformer-LSTM Model With 3D Separable Convolution for Video Prediction," in *IEEE Access*, vol. 12, pp. 39589-39602, 2024.

M. Mathai, Y. Liu, and N. Ling, A Lightweight Model with Separable CNN and LSTM for Video Prediction. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 516-520). IEEE, 2022.

M. Mathai, D. Rajan, and S. Emmanuel, Video forgery detection and localization using normalized cross-correlation of moment features. In *2016 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)* (pp. 149-152). IEEE, 2016.

Master of Engineering Thesis

M.Mathai, Video forgery detection, Nanyang Technological University, 2016.