Video Frame Prediction using Deep Learning

Bala Murali Krishna Kola

Video Frame Prediction using Deep Learning

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Bala Murali Krishna Kola

[Roll No: CS-1929]

under the guidance of

Prof. Bhabatosh Chanda

Professor

Electronics and Communication Sciences Unit

Indian Statistical Institute

Kolkata-700108, India

July 2021

To my parents and my guide

CERTIFICATE

This is to certify that the dissertation entitled **"Video Frame Prediction using Deep Learning"** submitted by **Bala Murali Krishna Kola** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

[illegible]

Prof. Bhabatosh Chanda
Professor,
Electronics and Communication Sciences Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would like to show my highest gratitude to my advisor, *Prof. Bhabatosh Chanda*, Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, for his guidance, continuous support and encouragement.

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable discussions which added an important dimension to my work.

Finally, I am very much thankful to my *parents* for their everlasting support.

Last but not the least, I would like to thank all of my friends for their help and support especially *Prajyot, Deepak* and *Ganesh* who stuck with me through thick and thin.

[illegible]

Bala Murali Krishna Kola
Indian Statistical Institute

Abstract

Advances in Deep Learning helped in developing exiting applications in the past few years like Style Transfer, Image Generation. It has helped us achieving super human performance and state-of-the-art results in various tasks like Object Recognition/Classification. It has established itself as a go-to technique for solving variety of tasks like Machine Translation in Natural Language Processing to Image Classification, Image Captioning in Computer Vision etc. Future Frame prediction is one of the problems in Computer Vision that received a lot of interest in the recent past for its use in Autonomous Driving, Weather Forecasting, Traffic Estimation etc. Next Frame Prediction is the focus of this thesis. This work will explain the fundamentals of future frame prediction and it will give an overview of existing approaches and present an approach to solve the problem.

Keywords: Video prediction, Future frame prediction, Convolutional LSTMs, Long Short Term Memory networks, Recurrent Neural Networks

Contents

Chapter 1

Introduction

1.1 Motivation

Intelligent Decision Making systems require understanding of the surrounding environment they interact with to take smart decisions. Better decision can be made if future can be predicted reliably. Ability to infer the position and motion of vehicles/pedestrians in the scene helps Self driving cars to drive effectively. Usually such systems get input from environment through video camera. Thus next Frame Prediction in essence tries to predict future image(s) from the previously observed sequence of images(events). Next frame prediction has been used in various applications, thus enabling us to have a better understanding of our surroundings. In order to achieve this, we should come up with a mechanism that can model the contents and dynamics of the scene efficiently. We need to extract spatio-temporal correlations. The use of Deep Learning over traditional Machine Learning techniques is that former doesn't require features to be picked manually and is capable of extracting features/representations from high dimensional image data. Recent studies have shown that Deep Learning has been successful in learning useful image representations[7]

1.2 Applications

Some of the applications where future frame prediction proven to be useful are

- Weather Nowcasting [15]

- Pedestrian Trajectories in traffic [2]
- Predicting future Object locations [11]
- Autonomous Driving [6]
- Video Interpolation [10]
- Precaution against unusual or unexpected activity detection [20]

1.3 Problem Statement

Humans find it trivial to estimate the position of objects in the scene by means of the physical rules they learn over time, but it is non-trivial for a machine. For example, consider the video of a ball thrown in air, we can estimate the future image sequence at least for a frame or two based on our intuition (laws of physics) about the motion of the ball. The task is to predict the subsequent frames, given a sequence of past video frames.

To define the problem formally, let $\boldsymbol{X}_t \in R^{w \times h \times c}$ be the t-th frame in the video sequence $\boldsymbol{X} = (X_{t-n+1}, ..., X_{t-1}, X_t)$ with n frames, where w, h and c denote width, height and number of channels respectively. The goal is to find out the subsequent sequence of frames $\hat{\boldsymbol{Y}} = (\hat{Y}_{t+1}, \hat{Y}_{t+2}, ..., \hat{Y}_{t+m})$ based on the given input $\boldsymbol{X}$, where $m \geq 1$ and $\boldsymbol{Y}_t \in R^{w \times h \times c}$.

Note that videos are generated in various ways and purposes. Most common form is movie: either feature film or documentary or promotional. Other two well known forms are surveillance and navigational. In case of movies for telling the story in an interesting way, video is captured continuously only for a short duration, called shot, and then shots are concatenated to make the full movie. Within a shot camera movement and also the movements of object in the scene are nil to small. As a result, content of the frames within a shot changes very slowly. On the other hand, frame content changes significantly from one shot to next shot in a movie. Surveillance and navigational videos may be considered to have single shot only. The task of video frame detection assumes that both $\boldsymbol{X}$ and $\boldsymbol{Y}$ are from the same shot.

1.4 Our approach and contribution

We have developed a model based on ConvLSTM [15] for predicting the next frame. Note that while the previous works used encoder-decoder type network, we stack multiple (three, in this work) layers of ConvLSTM followed by a 3D Convolution layer. This is clearly a major architectural modification to achieve the goal and may be considered as our contribution.

Ability to infer the next frame from fewer number of past frames is desirable. So, we predicted the next frame using past k frames, We have, through rigorous experiments, shown that $k = 5$ is sufficient to achieve acceptable result. This is another contribution that arises out of this work.

This is done as follows. Set of past k frames are used to predict the next frame and value of k is varied from 1 to 10. We evaluated each set of these predictions against the original frames. These predictions are evaluated using Structural Similarity Index (SSIM) [19] and Peak signal-to-noise ratio (PSNR). We find out that the improvement in visual quality isn't really appealing as we keep increasing the number of past frames beyond $k = 5$ for predicting the next frame.

1.5 Thesis Outline

Subsequent chapters of the thesis are organized as follows:

- **Chapter 2** covers the fundamentals required for understanding various techniques that are presented in the later chapters
- **Chapter 3** presents the literature review of existing approaches, underlying architecture. This gives a detailed view of each of the techniques.
- **Chapter 4** discusses experimental results
- **Chapter 5** summarizes overall results and prospective directions

Chapter 2

Fundamentals

To better understand the material presented in the later chapters let's review the basics of Recurrent Neural Networks, encoder-decoder models etc.

2.1 Recurrent Neural Networks

Recurrent Neural Networks(RNNs) [14] are suitable for processing sequential information. They are designed to extract long term dependencies. In a typical Neural Network, outputs are independent of each other but that's not the case with certain applications like sentence completion. Output at current time step relies on the computation in the previous time steps. It's useful to think of RNNs as having memory which captures what has been processed till the current time step.

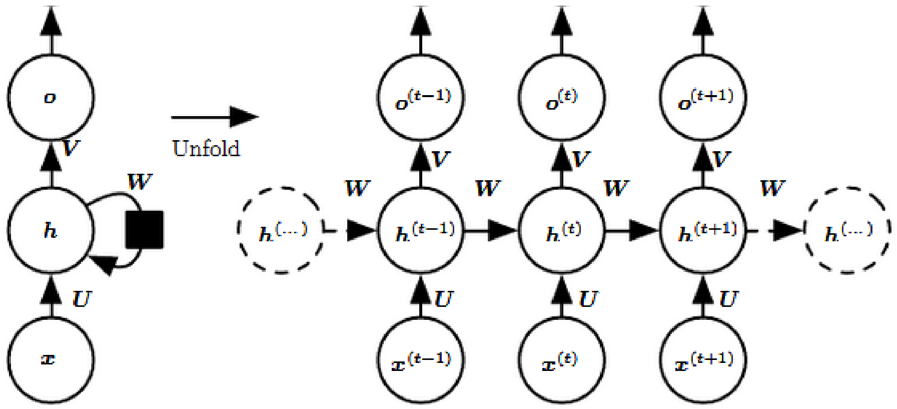

Figure 2.1: RNN Unfolded, Source: Goodfellow-et-al

Figure 2.1 explains how an RNN looks like on unrolling and how sequential data flows through the network. The parameters invovled are

- $x^{(t)}$ is the input at time t. It could be a embedding vector of a word or a frame in an image.
- $h^{(t)}$ is the hidden state at time t. This state acts like memory. $h^{(t)}$ is calculated on the basis of $h^{(t-1)}$ and $x^{(t)}$. This is how the long term dependencies are propagated forward through the network.
- $o^{(t)}$ is the output of the network for the input $x^{(t)}$ at time step t

The equations governing RNNs are

$$a^{(t)} = \mathbf{U}x^{(t)} + \mathbf{W}h^{(t-1)} + \mathbf{b} \tag{2.1}$$

$$h^{(t)} = \tanh(a^{(t)}) \tag{2.2}$$

$$o^{(t)} = \mathbf{V}h^{(t)} + \mathbf{c} \tag{2.3}$$

$$\hat{y}^{(t)} = softmax(o^{(t)}) \tag{2.4}$$

where $\mathbf{b}, \mathbf{c}$ are biases and $\mathbf{U}, \mathbf{W}, \mathbf{V}$ are parameters of the network. Here $\hat{y}^{(t)}$ is the prediction for the time step t. We use same parameters across all time steps of an RNN. This is how RNNs exploit weight sharing. Time invariance is also achieved in this manner. Take text generation for example, slight difference in position of a subject in a sentence should not alter the corresponding pronoun and we can't have different weights that can identify the subject at different time steps.

Theoretically, RNNs are capable of capturing long term dependencies irrespective of the number of time steps but in practice they aren't really promising. While training the network, RNNs suffer from Vanishing/Exploding gradient problem during the process of back-propagation through time, especially if the RNN is too deep. As a work around, other variants of RNNs like LSTM, GRU are proposed.

2.2 Long Short Term Memory networks

Long Short Term Memory networks [5] (LSTMs) are designed to mitigate the issue of Vanishing/Exploding gradient problems that crop up during training of vanilla RNNs. They are proven to be better than RNNs at capturing long term dependencies and are extensively used in variety of applications.

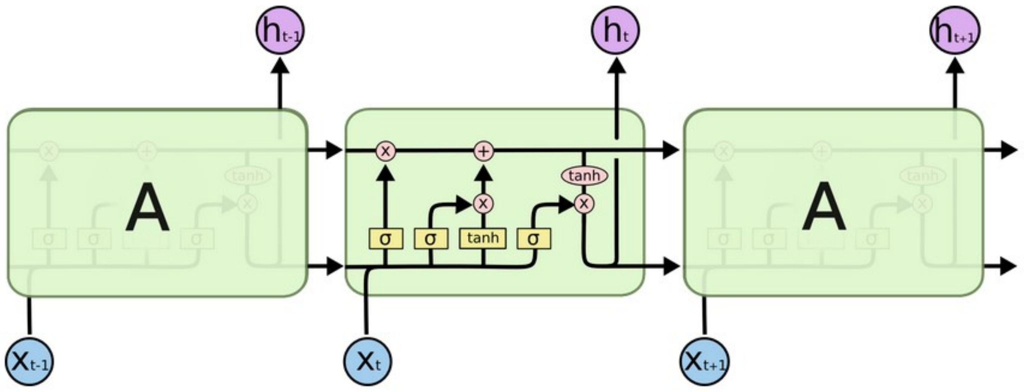

Figure 2.2: LSTM Architecture, Source: colah's blog (see text)

Figure 2.2, borrowed from Colah's blog*, gives a clear picture of internal structure of an LSTM cell. There are three sigmoid gates namely f, i, o respectively from left to right in figure 2.2 above in an LSTM cell. These are the equations for LSTM Cell in its vanilla form.

$$f^{(t)} = \sigma(\mathbf{U_f} x^{(t)} + \mathbf{W_f} h^{(t-1)} + \mathbf{b_f}) \tag{2.5}$$

$$i^{(t)} = \sigma(\mathbf{U_i} x^{(t)} + \mathbf{W_i} h^{(t-1)} + \mathbf{b_i}) \tag{2.6}$$

$$g^{(t)} = \tanh(\mathbf{U_g} x^{(t)} + \mathbf{W_g} h^{(t-1)} + \mathbf{b_g}) \tag{2.7}$$

$$o^{(t)} = \sigma(\mathbf{U_o} x^{(t)} + \mathbf{W_o} h^{(t-1)} + \mathbf{b_o}) \tag{2.8}$$

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot g^{(t)} \tag{2.9}$$

$$h^{(t)} = o^{(t)} \odot \tanh(c^{(t)}) \tag{2.10}$$

One thing to note is that g is exactly the hidden state h in the standard RNN. It's a candidate state determined purely using the input at the current time step. $\odot$ is point-wise vector product.

*http://colah.github.io/posts/2015-08-Understanding-LSTMs/

- $c^{(t)}$ is the internal memory of the unit and it's a combination of memory from previous units $c^{(t-1)}$ in the network and the current candidate state $g^{(t)}$. The combination is determined from the f and i gates.
- The forget gate f determines what info from $c^{(t-1)}$ to be retained/forget from the previous time steps.
- The input gate i estimates what info from the candidate state g to pass through to the next time steps i.e to calculate the new cell state $c^{(t)}$
- The output gate o measures how much of cell state to be exposed to the network for future time steps.

It's this gating mechanism, particularly the cell state $c^{(t)}$, that allows LSTM to overcome the challenge of vanishing gradients thereby successfully propagate the error backwards and also gives the ability to maintain dependencies that were captured over relatively longer sequences.

2.3 Sequence-to-Sequence Models

Typically RNNs map input sequences to output sequences i.e they generate an output $\hat{y}^{(t)}$ for input $x^{(t)}$. But that's not always desirable take for example of sentence translation. It's not always the case that every input word has a corresponding output word. Length of the input sequence and output sequences might not be same. In such cases, it's better to produce/predict the output sequence once we process the whole input sequence. Such models are called sequence-to-sequence models [17].

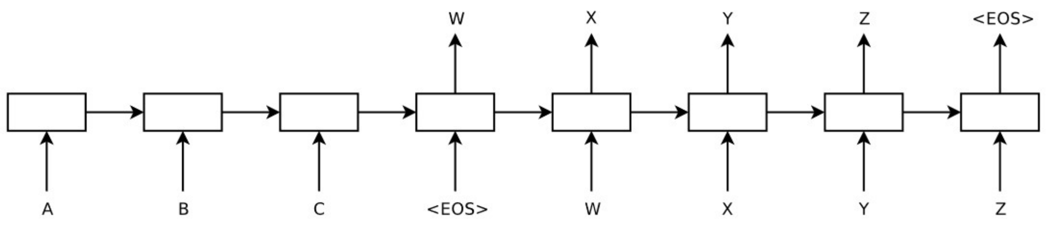

Figure 2.3: The sequence-to-sequence architecture, Source: [17]

A sequence-to-sequence models a re also called encoder-decoder models as they typically contains two RNN based modules an Encoder and a Decoder. Encoder distills the inputs sequence such that its output along with the internal cell states captures the input representations concisely. The decoder unit takes the outputs of encoder along with encoders final cell state and then starts producing the target sequence. The difference between the vanilla RNNs and the sequence-to-sequence models is that former models the distribution

$$P(Y_t|X_1, \dots, X_t, Y_1, \dots, Y_{t-1})$$

whereas the second one models the distribution

$$P(Y_t|X_1, \dots, X_T, Y_1, \dots, Y_{t-1})$$

where T is the length of the input sequence i.e seq-to-seq Models takes into account the whole input sequence for generating the output sequences.

2.4 Quality Evaluation Metrics

We need to evaluate the quality of the predicted frames against the ground truth frames. We have used the following metrics

2.4.1 Structual Similarity Index

Structural Similarity (SSIM)[19] is used to calculate the perceptual similarity between two images. It compares the two images based on luminance $l(\mathbf{x}, \mathbf{y})$, contrast and structural similarity metric $cs(\mathbf{x}, \mathbf{y})$ [21]. SSIM is calculated as:

$$SSIM(\mathbf{x}, \mathbf{y}) = l(\mathbf{x}, \mathbf{y}) \cdot cs(\mathbf{x}, \mathbf{y}) \tag{2.11}$$

This value lies in range [0, 1], *higher the value of SSIM indicates greater the similarity between the two images.* The metrcis $l(\mathbf{x}, \mathbf{y})$ and $cs(\mathbf{x}, \mathbf{y})$ are defined as:

$$l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{2.12}$$

$$cs(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{2.13}$$

where $\mu_\bullet$, $\sigma_\bullet$ are the mean, standard deviation respectively and σ_{xy} is the covariance of x and y. The constants $C_1 = (K_1 \cdot L)^2, C_2 = (K_2 \cdot L)^2$ where $K_1 = 0.01, K_2 = 0.03$ and L is the dynamic range of pixel values in the image. Typically for 8-bit gray scale images $L = 255$, since we normalize our images, we set L to 1 while calculating SSIM

2.4.2 Peak Signal-to-Noise Ratio

Peak Signal-to-Noise Ratio (PSNR) is another metric used to assess two images. It is the ratio between maximum possible pixel intensity and the noise that corrupts the image during the reconstruction process.

$$PSNR(\mathbf{x}, \mathbf{y}) = 10 \cdot \log \left(\frac{\mathbf{y}_{max}^2}{\frac{1}{wh} \sum_{i=1}^{w} \sum_{j=1}^{h} (\mathbf{x}_{ij} - \mathbf{y}_{ij})^2} \right) \tag{2.14}$$

where $\mathbf{y}_{max}$ is the maximum intensity possible for a pixel in an image with the size $w \times h$. It is measured in log decibel scale. Higher the PSNR means lesser the reconstruction error. This implies the quality of the prediction is better.

Chapter 3

Literature Review

Video frame prediction has attracted attention of the researchers since last decade associated with various applications. So quite a few methods have already been reported [2], [6], [10], [11]. Bhattacharyya et al. tried to predict the movement of pedestrians in next couple of frames in traffic scenario [2]. This may help autonomous driving system in a busy area. However, the method proposed by Hu et al. [6] proposed a frame prediction technique for explicitly autonomous driving. Liu et al. [10] suggested a more general method for various types of appliactions. They considered video data as a volume (3-dimensional: x, y, t) and have built voxel flow network to synthesize the future frames. A very important application is addressed in [11] where using RADAR cloud images from the past the precipitation and rainfall are predicted for one hour into the future. They have used a recurrent model on image sequences to achieve the goal. Two other neural network based systems for video frame prediction is presented below in a more detail because of their proximity to our work.

3.1 Fully Connected LSTM for Future Prediction

A video sequence is a time series of frame. We need to use an RNN based model to capture temporal correlations among the video frames. Srivastava et al. [16] came up with the idea of FC-LSTMs to solve the problem. They used a a pretrained convolutional network to convert the video frames into some high-level representations (vectors) or vectorized by flattening the frame. Then they feed these vectors to LSTM based encoder-decoder model to produce the future frames. Figure 3.1 shows how the future frames are generated.

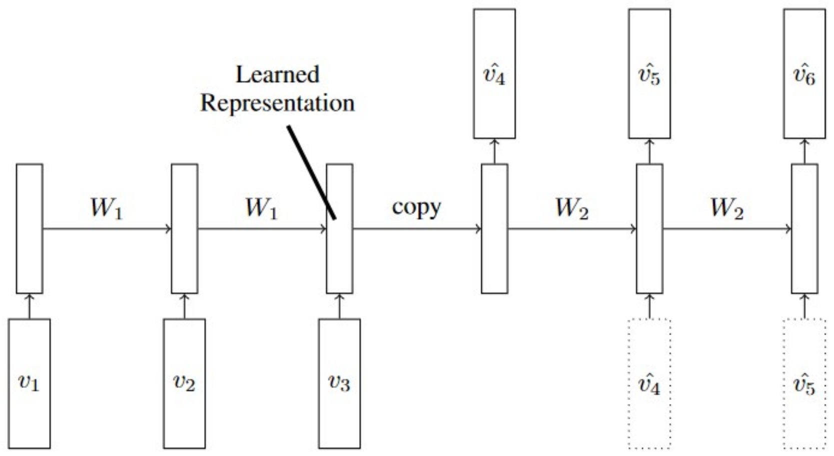

Figure 3.1: LSTM Future Predictor Model, Source: [16]

Video frames are fed to the encoder module. Once the encoder process the input sequence, the final state of the encoder is the representation of the input video sequence. It is then copied to the decoder. Decoder takes over from this point and then start producing the future frames of the video, one frame per time step. A frame is generated by feeding the previously predicted frame as input at the current time step.

The reason why it is called Fully Connected LSTM is that every pixel in the input image is connected to every pixel in the output image. Connections among uncorrelated regions are redundant. This is the issue with this approach as it requires huge number of parameters. To overcome this caveat, ConvLSTM has come into the picture and it is discussed next.

3.2 Convolutional LSTM Model

In order to predict the frame in future, we need to capture spatial inner structure within the frames and how it evolves over time. It's ideal to use convolutions in an RNN based model to extract spatial correlations. This is the motivation for Convolutional LSTM (ConvLSTM) unit. Shi et al.[15] introduced the idea of using convolutions in LSTM. Mathematically we can write the ConvLSTM as follows:

$$f^{(t)} = \sigma(\mathbf{U_f} * X^{(t)} + \mathbf{W_f} * H^{(t-1)} + \mathbf{P_f} \odot C^{(t-1)} + \mathbf{b_f}) \tag{3.1}$$
$$i^{(t)} = \sigma(\mathbf{U_i} * X^{(t)} + \mathbf{W_i} * H^{(t-1)} + \mathbf{P_i} \odot C^{(t-1)} + \mathbf{b_i}) \tag{3.2}$$
$$G^{(t)} = \tanh(\mathbf{U_g} * X^{(t)} + \mathbf{W_g} * H^{(t-1)} + \mathbf{b_g}) \tag{3.3}$$
$$o^{(t)} = \sigma(\mathbf{U_o} * X^{(t)} + \mathbf{W_o} * H^{(t-1)} + \mathbf{P_o} \odot C^{(t-1)} + \mathbf{b_o}) \tag{3.4}$$
$$C^{(t)} = f^{(t)} \odot C^{(t-1)} + i^{(t)} \odot G^{(t)} \tag{3.5}$$
$$H^{(t)} = o^{(t)} \odot \tanh(C^{(t)}) \tag{3.6}$$

where '*' denotes convolution operation and $\odot$ denotes Hadamard(point-wise) product. The cell state $C^{(t)}$ and the hidden states $H^{(t)}$ are feature maps i.e tensors. The idea of *Peephole* [3] for LSTM cells is also extended to ConvLSTM cells. These equations are similar to ones used in the variant of LSTM using *peephole* connections except that ConvLSTM use convolutions instead of matrix vector multiplications. The variant of LSTM we mentioned here uses *peephole* connections through which direct access to cell state is given to gates f, i, o. $\mathbf{P}_\bullet$ corresponds to the weights associating access to the cell state for the gates f, i, o. So the parameters that the networks learns are kernel weights $\mathbf{U}_\bullet, \mathbf{W}_\bullet$ and $\mathbf{P}_\bullet$. Hence the parameters required are substantially less compared to the technique presented above.

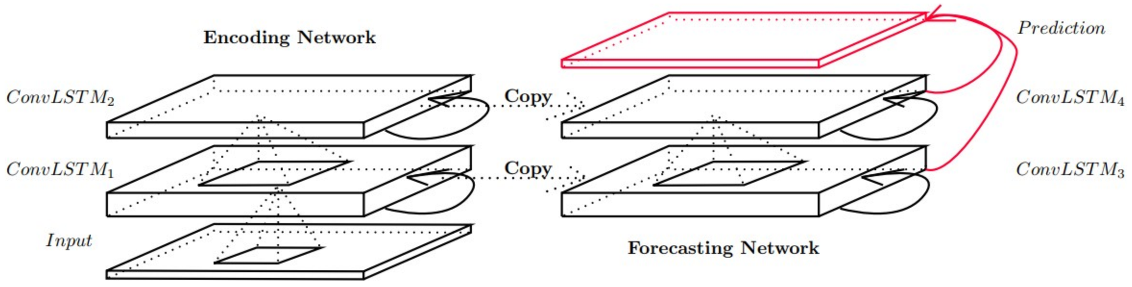

Figure 3.2: Encoding-forecasting Architecture, Source: [15]

Shi et al.[15] extended the idea presented in [16] with ConvLSTM layers. They developed a model for precipitation nowcasting [15]. Nowcasting means forecasting the rainfall in the immediate future, say, in the next 1 hour. They used precipitation radar echo images from the past to predict the future radar echo frames. The model developed is an encoder-decoder model as earlier which uses ConvLSTM as basic computational unit. The recurrent nature of the model allows it to memorize the temporal information in the radar echoes. Early work on nowcasting is due to Mukherjee et al. [12] and [13]. In these works Mukherjee et al. tried to estimate each pixel in the predicted frame by means of fully connected neural network, where input is candidate pixel of the previous frames as well as its neighbours.

Chapter 4

Experiments

4.1 Model Architecture

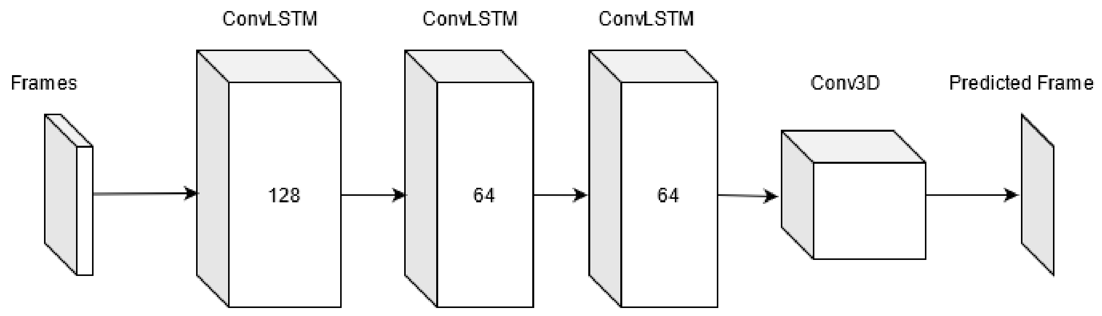

Figure 4.1: The proposed architecture for future frame prediction

Our proposed architecture for future frame prediction is shown in Figure 4.1. This model is inspired from the architecture used in [15] where they used encoder-decoder style, while our model is stack of three ConvLSTM layers followed by a 3D Convolution layer. This is a significant deviation from the earlier model and may be considered as our contribution. To keep the resolution same as that of the input frame, we pad the frames during every convolution operation across the entire network. The three ConvLSTM layers use kernel sizes of 5×5, 3×3, 1×1 and produce tensor feature maps in the domain of $\mathbb{R}^{w \times h \times 128}$, $\mathbb{R}^{w \times h \times 64}$, $\mathbb{R}^{w \times h \times 64}$ respectively. Then we use 3D convolution layer to map the output feature maps to the one time step shifted frames. Sigmoid activation function is used in the final layer to squash the output to 0-1 range. Our model and implementation is different from [15] in the following ways. We trained by feeding the 64×64 grayscale image as it is, while they transformed the image into a tensor of $16 \times 16 \times 16$ by taking patches of 4×4. We dropped *peepholes* [3] in ConvLSTM cells. They concatenated hidden states from all the decoder layers per time step and then perform convolution to predict the output image where as we considered only the hidden states in the last ConvLSTM layer. We choose to take n previous hidden states from this layer to predict the next frame. We kept this n value small in range $[1, 3]$ as a frame could be more relevant to the frames from recent past but not to the frames far in the past. We can choose to convolve only one previous hidden state but having context of few previous frames makes the prediction better is the idea. Note that using context of more previous frames increases the computation time.

4.2 Training

4.2.1 Moving-MNIST Dataset

For evaluating the performance of our proposed network for video frame prediction, we have applied our model on a benchmark data set, namely Moving-MNIST Dataset, suitable for this purpose. Moving-MNIST Dataset [1] contains 10000 video samples. Each sample video consists of 20 frames of size 64×64 having two handwritten digits moving randomly. This dataset is synthetically generated using 500 digits in MNIST dataset [8].

4.2.2 Loss Function

We have used binary cross-entropy loss as criterion function between ground truth frame T and the predicted frame P. So our loss function based on binary cross-entropy is defined as

$$\mathcal{L}_{BCE} = -\sum_{i,j,k} T_{i,j,k} \log P_{i,j,k} + (1 - T_{i,j,k}) \log (1 - P_{i,j,k}) \tag{4.1}$$

4.2.3 Parameter Tuning and Results

The network is trained to minimize the binary cross-entropy loss. Adam optimizer is used with learning rate 10^{-3}. The model is trained using 2000 samples with 1800 samples used for actual training and remaining 200 for validation. We train the model for 25 epochs with a batch size of 5 video samples per iteration. In total the network parameters are allowed to be updated trained for 9000 iterations.

4.2.4 Results

In this section we present the experimental results for video frame prediction on Moving-MNIST dataset and evaluate the results both visually and also objectively.

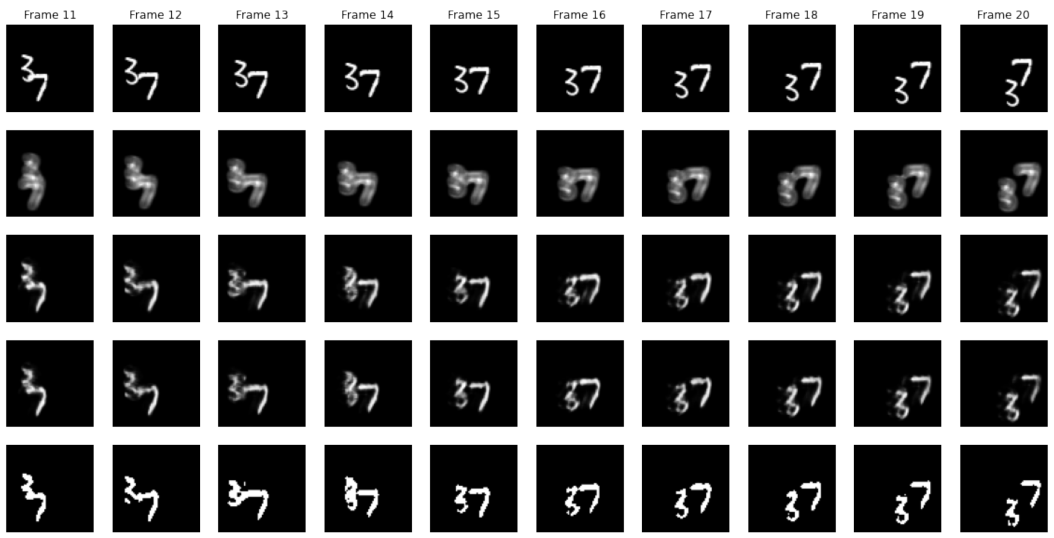

Figure 4.2: From top to bottom: Target ground truth sequence; Next frame predictions using previous 1 frame, 5 frames, 9 frames; final row are the images obtained by applying thresholding to the frames predicted in the 3rd row.

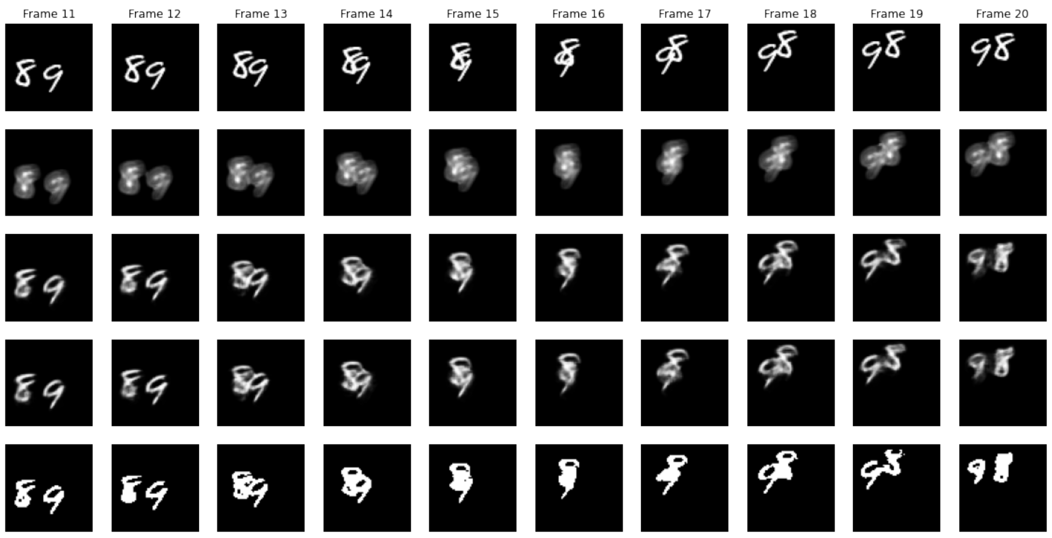

Figure 4.3: From top to bottom: Target ground truth sequence; Next frame predictions using previous 1 frame, 5 frames, 9 frames; final row are the images obtained by applying thresholding to the frames predicted in the 3rd row.

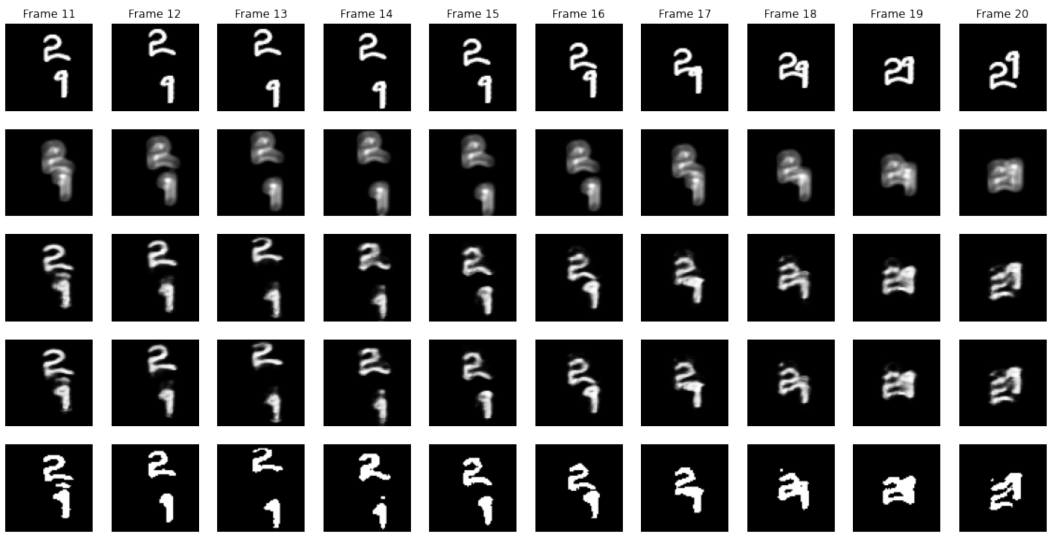

Figure 4.4: From top to bottom: Target ground truth sequence; Next frame predictions using previous 1 frame, 5 frames, 9 frames; final row are the images obtained by applying thresholding to the frames predicted in the 3rd row.

For the sake of evaluation, we have used 200 samples from the Moving-MNIST dataset different from what we used for training and validation. Figures 4.2, 4.3 and 4.4 show the output of three random samples from the dataset. The model is made to predict last 10 frames, i.e., frames 11-20, by taking corresponding previous k frames. We have varied the value of k from 1-10. To make it clear, suppose we are predicting the 15^{th} frame and the value of k is 4. We choose frames 11-14 in the input sample to make the prediction and for predicting 16^{th} we use frames 12-15 and so on. In this way we predict frames 11-20 using corresponding 4 previous frames. Thus, we get 10 sets of predicted frames for each of the samples where each set corresponds to a value of k (i.e., using k previous frames) where k lies in range 1-10. Top row of Figure 4.2 shows the groundtruth of the predicted frames (i.e., actual frames #11 to #20 of the original video) which are the targets of predicting network to reach. 2nd, 3rd and 4th rows from the top show the predicted frames for $k = 1$, $k = 5$ and $k = 9$ respectively. It is visually seen that there is almost no difference the corresponding frames in 3rd and 4th row. In other words, there is no qualitative difference in results for $k = 5$ and $k = 9$, We will justify this also in terms of objective image quality measures. Figures 4.3 and 4.4 can also be described in a similar way.

We have quantitatively evaluated the predicted frames using structural similarity index (SSIM) [19] and peak signal-to-noise ratio (PSNR). The results are summarized in Table 4.1 for $k = 1$ to 10.

Table 4.1: Evaluation results of predicted frames for different number of previous frames

# of Past Frames	**Mean SSIM**	**Mean PSNR**
1	0.6800	16.0881
2	0.7819	18.2370
3	0.8027	18.7172
4	0.8133	18.9025
5	0.8163	18.9492
6	0.8183	18.9656
7	0.8205	18.9734
8	0.8221	18.9703
9	0.8234	18.9643
10	0.8246	18.9598

In Table 4.1 mean values of PSNR and SSIM computed over 200 samples are presented and we see that for $k > 5$ values of PSNR and SSIM do not change much. Like visual analysis, these objective criteria also suggest that previous 5 frames are sufficient to predict the future frame. Data of Table 4.1 is presented graphically in Figure 4.5 and

Figure 4.6.

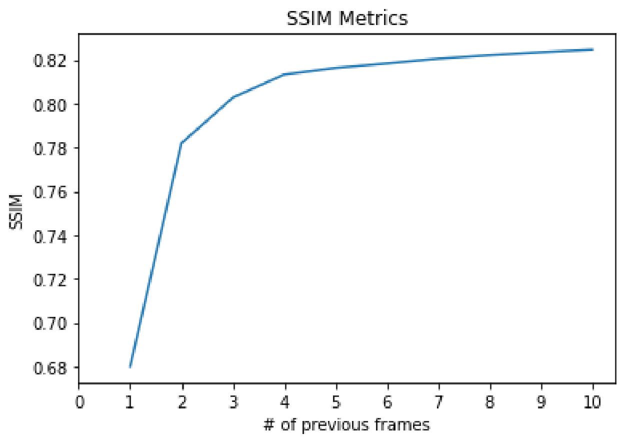

Figure 4.5: Mean SSIM values versus number of previous frames used for prediction

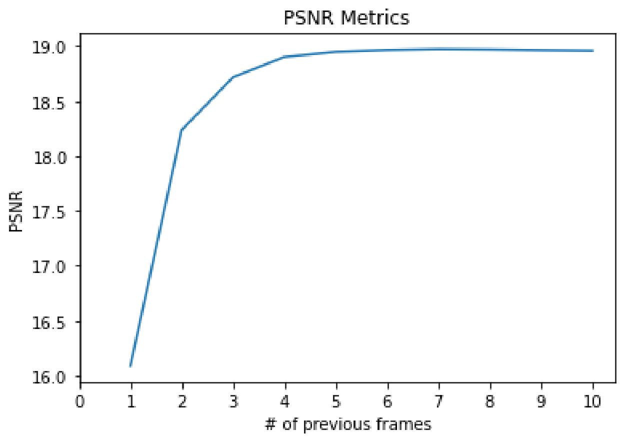

Figure 4.6: Mean PSNR values versus number of previous frames used for prediction

We observed that next frame predictions using fewer (1-2) past frames are blurry. But as we increase the number of previous frames, frames were getting better. Figures 4.2, 4.3 and 4.4 indicate that from rows 3 and 4, there isn't much visual difference from the frames obtained by predicting using past 5 frames and 9 frames. It's also evident from the plots 4.5 and 4.6 that improvement starts to cease around 4-5 previous frames which is an interesting finding.

4.3 Software

We have used some standard software for implementing our proposed method and to obtain experimental results. Important ones are listed below.

- Python 3.7.10
- Tensorflow 2.5.0 and Keras 2.4.3
- OpenCV-Python 4.1.2.30

Chapter 5

Conclusion

5.1 Summary and Future Work

In this thesis, we have presented the details of our model which generates one frame into future. We can tweak our models to generate a sequence of frames by feeding in the frames that were predicted in the previous time steps. However, the quality of the future frames has worsened rapidly. Frames predicted beyond 3-4 time steps started to show up blobs and are blurry. This model might not be sufficient to give future frames with a decent accuracy. Choice of the loss function is critical in producing frames of satisfying quality. The loss function that we used is pixel-based function. This is the reason for the blur in the predictions as it tends to minimise the error by averaging out the possible future frames. It's appropriate to use feature-based loss functions by capturing and estimating the motion of the objects in the scene and then use this information to predict the next frame. Such a loss function is helpful in applications needing to forecast the frames far into the future(10 timesteps and beyond). Also the frames predicted weren't sharp. GANs [4] are proven effective in producing clear images[9]. Villegas et al [18] used movement information of skeletal structure of objects in the frames and GAN based model to generate quality next frames. It has shown good results in long-term prediction(128 future frames) and is one of the possible future directions to pursue. A thorough framework to evaluate the similarity/cost of the predicted frames based on high level visual similarity is essential and it is an active area of research.

Bibliography

[1] Moving-mnist dataset http://www.cs.toronto.edu/~nitish/unsupervised_video/

[2] Bhattacharyya, A., Fritz, M., Schiele, B.: Long-term on-board prediction of people in traffic scenes under uncertainty. In: CVPR. pp. 4194–4202. IEEE Computer Society (2018)

[3] Gers, F.A., Schraudolph, N.N., Schmidhuber, J.: Learning precise timing with lstm recurrent networks. J. Mach. Learn. Res. 3(null), 115–143 (Mar 2003), https://doi.org/10.1162/153244303768966139

[4] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (2014)

[5] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (Nov 1997), https://doi.org/10.1162/neco.1997.9.8.1735

[6] Hu, A., Cotter, F., Mohan, N., Gurau, C., Kendall, A.: Probabilistic future prediction for video scene understanding. CoRR abs/2003.06409 (2020), https://arxiv.org/abs/2003.06409

[7] Jiang, Y., Dong, H., El-Saddik, A.: Baidu meizu deep learning competition: Arithmetic operation recognition using end-to-end learning OCR technologies. IEEE Access 6, 60128–60136 (2018), https://doi.org/10.1109/ACCESS.2018.2876035

[8] LeCun, Y., Cortes, C., Burges, C.J.: The mnist database of handwritten digits http://yann.lecun.com/exdb/mnist/

[9] Ledig, C., Theis, L., Huszar, F., Caballero, J., Aitken, A.P., Tejani, A., Totz, J., Wang, Z., Shi, W.: Photo-realistic single image super-resolution using a generative adversarial network. CoRR abs/1609.04802 (2016), `http://arxiv.org/abs/1609.04802`

[10] Liu, Z., Yeh, R.A., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: ICCV. pp. 4473–4481. IEEE Computer Society (2017)

[11] Makansi, O., Ilg, E., Çiçek, Ö., Brox, T.: Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. CoRR abs/1906.03631 (2019)

[12] Mukherjee, A., Shukla, B.P., Chanda, B., Mukherjee, D.P.: A novel neural network based meteorological image prediction from a given sequence of images. In: 2nd International Conference on Emerging Applications of Information Technology (EAIT) (2011)

[13] Mukherjee, A., Shukla, B.P., Chanda, B., Pal, N.R., Mukherjee, D.P.: Prediction of meteorological images based on relaxation labeling and artificial neural network from a given sequence of images. In: International Conference on Computer Communication and Informatics (ICCCI) (2012)

[14] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (Oct 1986), `https://doi.org/10.1038/323533a0`

[15] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., Woo, W.: Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: NIPS. pp. 802–810 (2015)

[16] Srivastava, N., Mansimov, E., Salakhutdinov, R.: Unsupervised learning of video representations using LSTMs. In: ICML (2015)

[17] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. CoRR abs/1409.3215 (2014), `http://arxiv.org/abs/1409.3215`

[18] Villegas, R., Yang, J., Zou, Y., Sohn, S., Lin, X., Lee, H.: Learning to generate long-term future via hierarchical prediction. CoRR abs/1704.05831 (2017), `http://arxiv.org/abs/1704.05831`

[19] Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 13(4), 600–612 (2004)

[20] Zhao, B., Fei-Fei, L., Xing, E.P.: Online detection of unusual events in videos via dynamic sparse coding. In: IEEE Computer Vision and Pattern Recognition (CVPR) (2011)

[21] Zhao, H., Gallo, O., Frosio, I., Kautz, J.: Loss functions for neural networks for image processing. CoRR abs/1511.08861 (2015), `http://arxiv.org/abs/1511.08861`