Team 2 - Final Project Report

Pushkal Mishra
EE20BTECH11042
ee20btech11042@iith.ac.in

Perambuduri Srikaran
AI20BTECH11018
ai20btech11018@iith.ac.in

Sourabh Somnath Gholap
EE20BTECH11047
ee20btech11047@iith.ac.in

Ankur Kumar
EE20BTECH11057
ee20btech11057@iith.ac.in

Abstract

With the rising trend of autonomous vehicles and intelligent agents, the crucial part in their functioning is their ability to make precise decisions. For this, sometimes it is helpful to predict anomalies in the near future before hand for error-free operations. One of the ways we can do it is by predicting the future frames from the past video frames. Many models exist which perform the above task but blurry predicted frames is a major issue. To solve this problem, we aim to explore image deblurring techniques using deep learning models in addition to the current state of the art prediction models to increase its performance. Furthermore, the problem of relevance arises when predicting more number of frames; so we will also explore methods of using input frames together with our predicted frames.

1. Introduction

In a broad sense, the prediction and anticipation of future events is a key component of intelligent decision making systems which we humans can perform seamlessly. But from the machine's point of view, the task of predicting future frames is extremely challenging due to factors such as occlusions, camera movement, lighting conditions, cluttering, or object deformations.

The problem of video frame prediction has become highly intriguing in recent times due to its importance in a wide variety of computer vision applications such as autonomous vehicles, intelligent agents [3], precipitation nowcasting [5], and many more. Also, recent advances in deep learning has improved the performance of video prediction as it is naturally a good self-supervised learning task.

Various deep learning techniques exist which extract meaningful spatio-temporal correlations from video data in a self-supervised learning fashion, such as in CNN-RNN-CNN based architectures the models generate predictions frame by frame with the previous output for capturing temporal evolution of frames. This paper [2] implements a completely CNN based architecture that generates predictions in a one-shot manner and potentially employ UNET connections between the convolutional layers.

The major problem we observed in the current models is that the predicted frames are slightly blurry for dynamic objects in the video which causes inaccuracies in reconstruction. In this project, we explore image deblurring techniques for the predicted frames to improve the performance of existing models.

2. Problem Statement

Current video frame prediction models use the predicted images to generate new predictions. It was observed that the model outputs slightly blurry images. This causes a cascading effect which makes the future frames to be blurrier.

To solve this problem, we look to deblur the predicted images before sending it to the model for predicting the next frames. We deblur the images using deep learning techniques.

But, predicting too many frames in continuum will cause loss of relevance in the original video. We look to explore the approach of using the predicted images and the input images together to predict future frames accurately.

This would be especially useful in autonomous driving. We would be predicting the future positions of obstacles accurately and also maintain the relevance by taking inputs from the environment at appropriate times. This also calls for having models with lesser inference time for this approach to work.

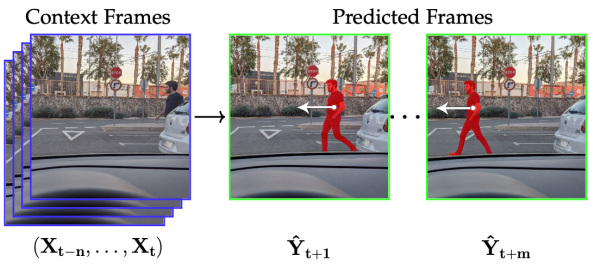


Figure 1. Video Frame Prediction for Autonomous driving

3. Literature Review

3.1. Video Frame Prediction

3.1.1 Next frame prediction using ConvLSTM

This paper [1] introduced a method for predicting future frames based on a series of prior input frames using ConvLSTM. The input video was first segmented into frames which are stored in a 2D array. Then each frame is sent one at a time to the ConvLSTM module which extracts the featues, stores it and then takes in the next input frame. Using these learned features in the LSTM memory cells they make a prediction for the next frame. This predicted output is then compared with ground truth data.

The ConvLSTM model is divided into two sections: LSTM Encoder and Decoder. The first section reads the input sequence, extracts a **fixed-length** motion vector and a predicted frame with the help of previous reconstructed frame. Then the residual frame and motion vectors are passed to the LSTM decoder wherein a frame is predicted using these vectors and previous reconstructed frame. The final system's output is the sum of residual frame and predicted frame. To assess the prediction accuracy, this output frame is compared to the ground truth data.

This paper uses SSIM index and Perceptual Similarity as parameters to evaluate the model's performance.

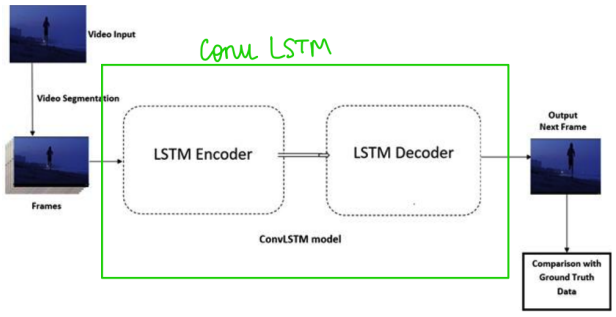


Figure 2. Architecture of the ConvLSTM model

3.1.2 VPTR

This [7] is a transformer block for video future frames prediction based on an efficient local spatial-temporal separation attention mechanism. To avoid prediction of similar frames, a contrastive feature loss is applied to maximize the mutual information between predicted and ground-truth future frame features.

This proposed transformer block is efficient for spatio-temporal feature learning by combining spatial local attention and temporal attention in two steps. The new Transformer block successfully reduces the complexity of a standard Transformer block with respect to same input spatio-temporal feature size, specifically, from $\mathcal{O}(T^2H^2W^2)$ to $\mathcal{O}(H^2W^2P^2 + T^2)$.

It has been demonstrated that the VPTR models, VPTR-NAR and VPTR-FAR, which utilize a straightforward attention mechanism, have the capability to achieve better results than the state-of-the-art VFFP models based on ConvLSTM that are more complex.

A comparison was formally carried out between two variants of VPTR, and the findings indicate that VPTR-NAR has a quicker inference rate and a lower error accumulation during inference compared to its counterpart. However, the process of training VPTR-NAR is more challenging. To address this issue, they utilized a contrastive feature loss that aims to increase the mutual information of the predicted and ground-truth future frame features.

VPTR-NAR is different from ConvTransformer with respect to the fundamental attention mechanism. ConvTransformer proposed a custom hybrid multi-head attention module based on convolution, but VPTR-NAR uses the standard multi-head dot product attention.

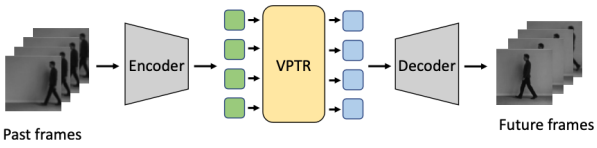


Figure 3. Overall framework of VPTR

3.1.3 Temporally Consistent Video Transformer

Temporally Consistent Video Transformer (TECO) [6] is a vector-quantized latent dynamics video prediction model that learns compressed representations to efficiently condition on long videos of hundreds of frames during both training and generation. TECO works in two broad steps: first the model learns temporal representations using encoder and temporal transformers and then it predicts future frames using some dynamics prior and a decoder. This model was

applied to 3D environments such as DMLab, Minecraft and Habitat.

To spatially compress the video data, the encoder of the VQ-GAN encodes the current frame x_t conditioned on the previous frame by channel-wise concatenating x_{t-1} and then quantizes the output using codebook C to produce $z_t = E(x_t, x_{t-1})$. Then a single strided convolution was applied to downsample each discrete latent z_t to reduce the losses and lower the spatial resolutions. Afterwards, a large transformer model was learned to model temporal dependencies, and then apply a transposed convolution to upsample the representation back to the original resolution z_t. Then a decoder was used which is an upsampling CNN that reconstructs $\hat{x}_t = D(z_t, h_t)$. Lastly to model the dynamics prior, they used MaskGit which allows for faster and higher quality sampling compared to an autoregressive prior.

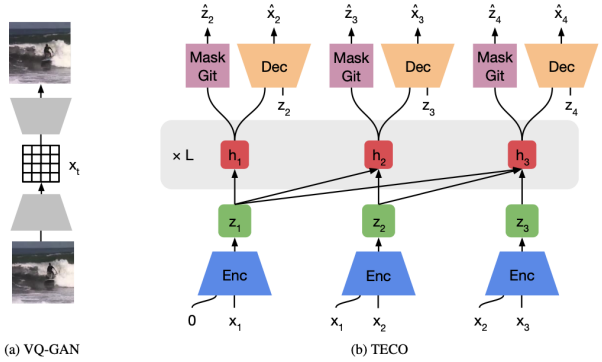


Figure 4. Architecture design of TECO

3.1.4 SimVP

SimVP [2] is a simple yet effective spatio-temporal predictive learning model using CNNs for frame prediction. It aims to design an autoencoder-like architecture that inputs the past frames and outputs the future frames while preserving the temporal dependencies. The CNN autoencoder is used to reconstruct a frame by borrowing voxels from nearby frames.

SimVP learns a mapping $\mathcal{F}_\theta : \mathcal{X}^{t,T} \rightarrow \mathcal{Y}^{t+1,T'}$ to encode the past frames $\mathcal{X}^{t,T}$ and decode the future frames $\mathcal{Y}^{t+1,T'}$.

The **spatial encoder** is employed to encode the high dimensional past frames into the low-dimension latent space. The **translator** learns both spatial dependencies and temporal variations from the latent space. The **spatial decoder** ultimately decodes the latent space into the predicted future frames.

A mapping is introduced between the spatial encoder and the spatial decoder to preserve the spatial features.

The training time and efficieny is better than ConvLSTM, PredRNN, PredRNN++, MIM, E3D-LSTM and PhyDNet. The predicted images are much clearer than other models.

The error is relatively large when the scene changes dramatically but remains low in most cases. SimVP accurately predicts the future trend to a large extent and unexpectedly finds the sudden traffic jam from the observations of placid transportation

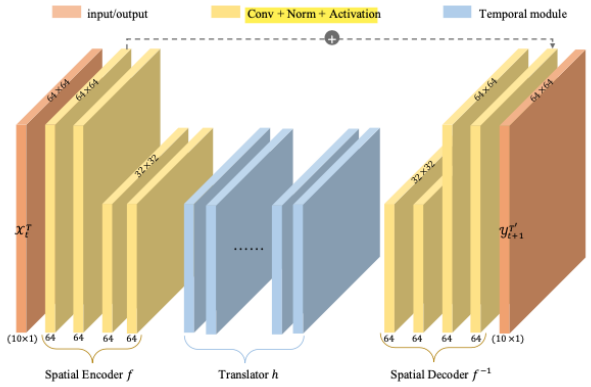


Figure 5. Architecture of SimVP

3.2. Image Deblurring

3.2.1 Efficient Transformer for Image Restoration

This paper [8] developes an efficient Transformer model that can handle high-resolution images for restoration task by introducing key design changes to the multi-head SA layer and a multi-scale hierarchical module that has lesser computing requirements than a single-scale network.

Say $H \times W$ is the spatial dimension and C is the number of channels, so firstly the model applies a convolution to obtain low-level feature embeddings $F_0 \in \mathbb{R}^{\mathbb{H}\times\mathbb{W}\times\mathbb{C}}$. Then these shallow features F_0 pass through a 4-level symmetric encoder-decoder and transformed into deep features. From the input, the encoder hierarchically reduces spatial size, while expanding channel capacity whereas the decoder takes low-resolution latent features as input and progressively recovers the high-resolution representations. To assist the recovery process, the encoder features are concatenated with the decoder features via skip connections which is followed by a 1×1 convolution to reduce the channels at all levels.

Further, the deep features were enriched in the refinement stage operating at high spatial resolution which yields quality improvements. Finally a convolution layer is applied to the refined features to generate a residual image which will be added to the original degraded image to obtain the refined image. To validate their model, they

performed various experiments for image processing tasks such as image deraining, image deblurring, defocus deblurring and image denoising which showed some promising results.

3.2.2 Blurry Video Frame Interpolation (BIN)

In this [4] paper, they proposed Blurry video frame Interpolation (BIN) method which includes a pyramid module and an inter-pyramid recurrent module. The structure of this pyramid module resembles a pyramid that consists of multiple backbone networks. Based on the pyramid structure, they have proposed an inter pyramid recurrent module which effectively exploits the time information. Specifically, the recurrent module adopts ConvLSTM units to propagate the frame information across time. They have formulated the joint blur reduction and frame rate up-conversion problem as maximizing a posteriori of the output frames conditioned on the blurred inputs. They formulated the joint blur reduction and frame rate up-conversion problem as maximizing a posteriori of the output frames conditioned on the blurred inputs.

$$\mathcal{F}^* = \max_{\mathcal{F}} p(\hat{I}_{1:1:2N-1}|B_{0:2:2N})$$

where $B_{0:2:2N}$ denotes the low-frame-rate blurry inputs starting from index 0 to 2N with a time step of 2, $\hat{I}_{1:1:2N-1}$ represents the restored and frame rate up-converted results, and refers to the optimal joint space-time enhancement model. $\mathcal{F}^*$ refers to the optimal joint space-time enhancement model. They have proposed to use trainable neural networks to approximate the above optimal model. So as a minimization of the loss function L over dataset S.

$$\begin{gathered}\min_{\mathcal{F}(.;\Theta)} \sum_{s\in\mathcal{S}} \mathcal{L}(\hat{I}_{1:1:2N-1}|I_{1:1:2N-1}) \\ \text{subject to } \hat{I}_{1:1:2N-1} = \mathcal{F}(B_{0:2:2N})\end{gathered} \tag{1}$$

where $I_{1:1:2N-1}$ denotes the ground-truth frames in the video sample $s \in \mathcal{S}$, and $\mathcal{F}(.;\Theta)$ refers to the proposed BIN with network parameters Θ.

Pyramid Module: It takes N + 1 frames $B_{0:2:2N}$ as input, and outputs the deblurred and the interpolated frames $\hat{I}_{1:1:2N-1}$

$$\hat{I}_{1:1:2N-1} = \mathcal{F}(B_{0:2:2N}) \tag{2}$$

The backbone network $\mathcal{F}_b$ interpolates an intermediate frame using two consecutive inputs:

$$\hat{I}_1 = \mathcal{F}_b(B_0, B_2)$$

Architecture of pyramid module is shown in figure 1.

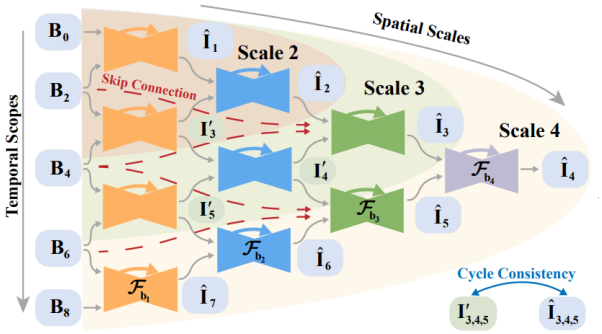


Figure 6. Pyramid Module

Inter-Pyramid Recurrent Module: This module enforce temporal motion consistency between neighbouring frames. It contains multiple ConvLSTM unit. Each ConvLSTM unit uses hidden states to propagate previous frame information to the current pyramid module. Illustration for BIN with scale 2 :
Here we generate two intermediate frames $\hat{I}_1^t$ and $\hat{I}_3^t$ by feed forwarding network $\mathcal{F}_{b1}$ twice:

$$\begin{aligned}\hat{I}_1^t &= \mathcal{F}_{b1}(B_0^t, B_2^t), \\ \hat{I}_3^t &= \mathcal{F}_{b1}(B_2^t, B_4^t)\end{aligned} \tag{3}$$

Then use this synthesized intermediate frames $\hat{I}_1^t$,$\hat{I}_3^t$ and the hidden state H^{t-1} to synthesize the deblurred frame $\hat{I}_2^t$.

$$\hat{I}_2^t = \mathcal{F}_{b2}(H^{t-1}, B_0, B_2)$$

Along with synthesizing the target frames, the ConvLSTM module also requires to maintain its cell state for temporal recurrence. Formulated the updating equation of the inter-pyramid recurrent module by:

$$H^t, C^t = \mathcal{F}_c(\hat{I}_3^t, H^{t-1}, C^{t-1})$$

where $\mathcal{F}_c$ refers to the ConvLSTM unit, C^{t-1} and C^t are previous cell state and current cell state,H^t refers to the current hidden state, and $\hat{I}_3^t$ denotes the current input. At time t and t+1, we obtain $\{\hat{I}_1^t, \hat{I}_2^t\}$ and $\{\hat{I}_1^t, \hat{I}_2^t\}$,respectively. By extending the iteration to time T, we can synthesize all the deblurred and interpolated frames $\hat{I}_{1:1:2N}$.

4. Implementation of ConvLSTM

In this section we show our progress and results obtained for video frame prediction using the Convolutional LSTM network. We referred the code from here.

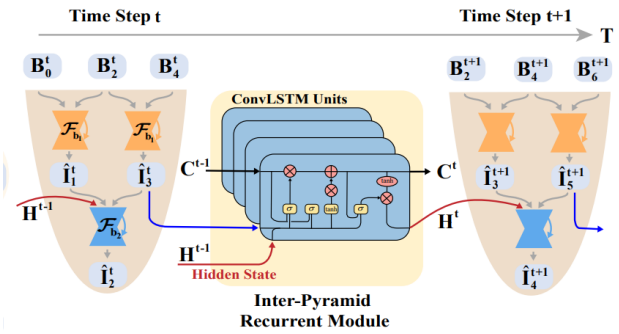


Figure 7. Computation Flow of BIN_2

4.1. Network Architecture

The architecture was inspired from this paper [1] which uses LSTM based encoder and decoder. The encoder reads the input sequence of images, extracts a **fixed-length motion vector** and a predicted frame with the help of previous reconstructed frame and current predicted frame. Then the residual frame and motion vectors are passed to the LSTM decoder wherein a frame is predicted using these vectors and previous reconstructed frame. The final system's output is the sum of residual frame and predicted frame. To assess the prediction accuracy, this output frame is compared to the ground truth data.

This is better understood from the below image:

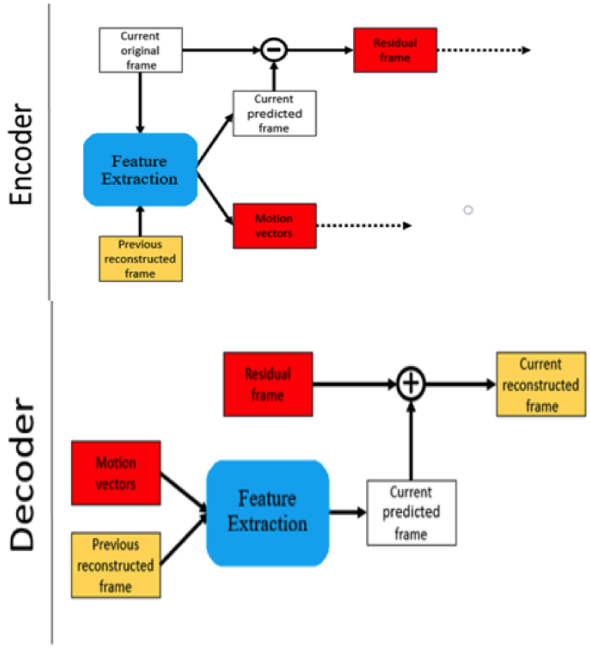


Figure 8. Encoder and Decoder network

4.2. Preliminary Results

4.2.1 Dataset

We train the model on two datasets: Moving MNIST[1] and Walking action from the KTH action dataset[2].

4.2.2 Training methodology

We trained the model on 60 input frames and predicted the next 10 frames and ran over the entire dataset for 50 epochs. We used a 3x3 size convolutional filter with 64 kernels and a learning rate of 10^{-4}. The model uses Adam optimizer for Binary Cross Entropy loss as we set our target pixels to be 0 or 1 and ReLU activation.

4.2.3 Results

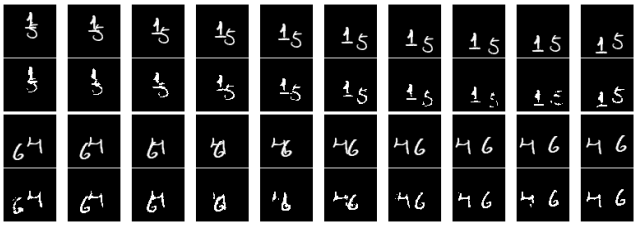


Figure 9. Output on Moving MNIST dataset. The first and third rows are the ground truth labels and the second and fourth rows are the predicted frames

Figure 10. Output on KTH Action dataset. The odd-numbered rows are the ground truth labels and the even-numbered rows are the predicted frames

The average of MSE loss for all the predicted frames is 299, 205 and 302 respectively.

5. Image Deblurring

We used a pre-trained Restormer [8] model to deblur the predicted frames. Our code is referred from this notebook

[1]Moving MNIST Dataset link
[2]KTH Action Dataset link

5.1. Preliminary Results

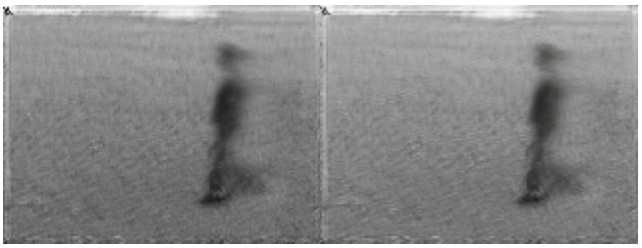


Figure 11. The deblurred frames after passing it through Restormer

6. Final Results

Firstly, we will describe the problem statement and explain it with an example. Then, we will go in detail about the ways we approached to solve the problem. We used KTH Action dataset for our problem.

6.1. Example of the Problem Statement with Explanation

Next frame prediction models predict the immediate next frame with a very good accuracy. But, predicting a series of frames with the given initial position and movement returns blurry frames.

For example, let the problem be that, we are given 40 frames of a video and we have to predict the next 5 to 10 frames accurately. We predict 1 frame using the given data. After that, we use the predicted frame and the given video such that there are 40 frames again, and predict the next frame. We do this process of removing the oldest frame from the given video and append the predicted frame till we get the required number of frames.

By solving the problem as above, we do not lose the temporal or spatial information. The frames predicted later are more blurry because the errors of the initially predicted frames are propagated. We can demonstrate this effect with the following figures.

To solve the problem, we motion deblur and/or denoise the predicted frames before predicting the next frames.

6.2. ConvLSTM Training Procedure

6.2.1 Initial Training Procedure - From MTR

Previously to train the ConvLSTM model during the Mid-Term report, we followed the next frame prediction procedure and passed the entire dataset to the model. But this caused some effects in the surroundings of the person (in the background) as seen in the image below.

After exploring the dataset we found out that there were two types of videos, one was shot at home in front of a white wall and the other was shot on a beach. Due to mixing of this training data, we can see some special effects around the legs of the subject.

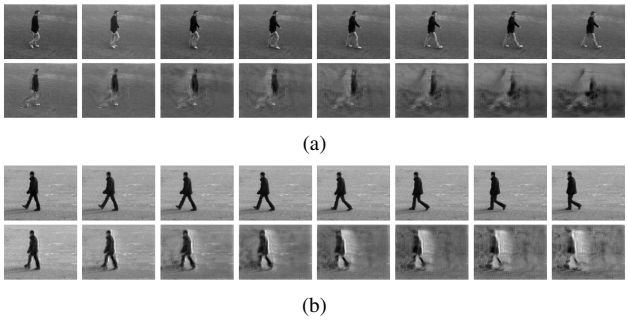


Figure 12. Demonstration of the cascading effect of the next frame prediction models. The first row is the ground truth frames. The second row is the predicted frames.

Figure 13. Side effect of training on the entire dataset.

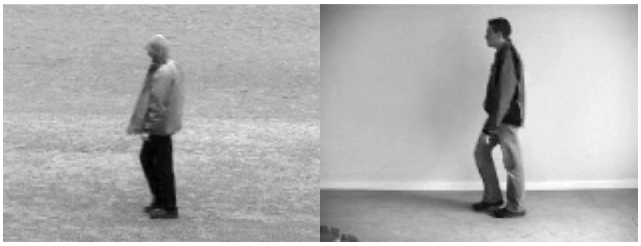


Figure 14. Two types of videos present in the dataset

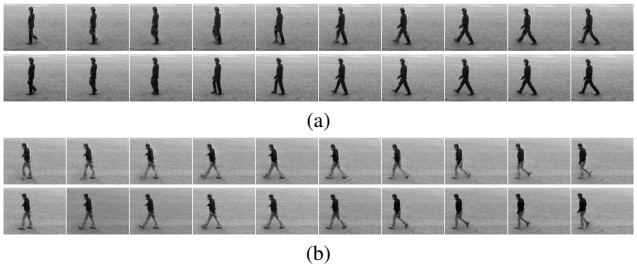
To avoid this issue, we segregated the dataset into two parts with respect to the background they were shot in.

6.2.2 Final Procedure

Now we train the ConvLSTM model on the videos shot at the beach and again using the next frame prediction concept.

We also used the weights obtained from the previous report for transfer learning due to shortage of datapoints (generally LSTMs are notoriously hard to train and require huge amounts of data to train).

The results of the next frame prediction are as follows-



(a)

(b)

Figure 15. Next frame prediction of ConvLSTM

6.3. Motion Deblurring Training Procedure

One can clearly observe from the above image pairs that the predicted images have undergone some blurring in the subject's leg which can be attributed to the inaccuracies in prediction of the ConvLSTM model. Visually, we can approximately say that the legs have blurred due to motion. Hence we use the motion deblurring model.

To train this model, we generated multiple pairs of next frame predicted images for different time instances of the video and also for different videos. Then, we fed this predicted frame and ground truth frame to a pre-trained restormer model.

6.4. Denoising Training Procedure

After performing motion deblur to the image, we observed that the noise in the background of the image was being amplified, and so we decided to denoise the image again using the restormer model set in denoising mode.

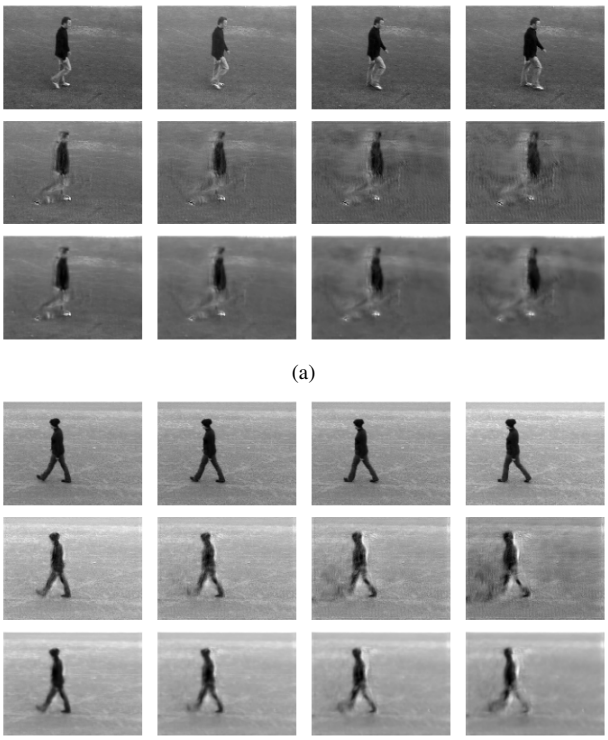
The training procedure for this method is exactly the same as the one we followed for motion deblurring.

6.5. Overall pipeline of the approach

We will summarize our overall approach.

- The input to the next frame prediction model (ConvLSTM) would be a video.
- We predict a new frame.
- This new frame is passed to the motion-deblurring and/or the denoising models.
- The resulting frame is appended to the end of the input video. And, the first frame of the input video is removed. The resulting video is fed to the model and the above steps are repeated till we predict the required number of frames.

The following images are the final predictions of our approach:



(a)

(b)

Figure 16. Final predictions of our approach. The first row is the ground truth frames. The second row is the frames predicted without deblurring and denoising. The third row is the set of frames predicted using our approach.

As we can see, the blurriness in the last frames has been reduced.

7. Future Work

Since the models of ConvLSTM and Restormer were large, it was difficult to train them. There are certain approaches which could be tried out if more computational power were present:

- **Ensembling the denoising and the motion-deblurring models**: We can denoise and deblur the models before passing it to the model for next frame prediction. The denoising model is a low-pass filter and the motion-deblurring model is a high-pass filter. This would act as a band-pass filter.
- **Trying out a different training methodology for the denoising and deblurring models**: In our current setup, we had trained the denoising model and the motion-deblurring models separately. Instead, we can

train both of them together. This may improve the performance of individual models.

8. Conclusion

We believe our approach to solve the problem of the cascading effect of blurriness in the continuous video frame prediction is a step in the right direction. With the right techniques used for training, the accuracy of the predictions can be increased.

References

[1] Padmashree Desai, C Sujatha, Saumyajit Chakraborty, Saurav Ansuman, Sanika Bhandari, and Sharan Kardiguddi. Next frame prediction using ConvLSTM. *Journal of Physics: Conference Series*, 2022. 2, 5

[2] Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z. Li. SimVP: Simpler yet better video prediction, 2022. 1, 3

[3] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection – a new baseline, 2017. 1

[4] W. Shen, W. Bao, G. Zhai, L. Chen, X. Min, and Z. Gao. Blurry video frame interpolation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5113–5122. IEEE Computer Society, jun 2020. 4

[5] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting, 2015. 1

[6] Wilson Yan, Danijar Hafner, Stephen James, and Pieter Abbeel. Temporally consistent video transformer for long-term video prediction, 2022. 2

[7] Xi Ye and Guillaume-Alexandre Bilodeau. VPTR: Efficient transformers for video prediction, 2022. 2

[8] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration, 2021. 3, 5