Lab Vision Systems: WS22: Video Prediction

Abstract. Prediction of future frames when presented with an initial sequence of images extracted from a video sequence is inherently referred to as Video Prediction. We construct an encoder-decoder style model architecture consisting of residual connections from the encoder to the decoder. To predict the future frames, Convolutional LSTMs are utilised. The encoder-decoder structure makes use of ResNet style blocks to achieve better results. The model is trained using Mean squared error, Mean average error and a combined loss with Mean squared error and Structural similarity index measure on the synthetically generated Moving-MNIST dataset as well as the KTH action dataset. The results are inspected using MSE, MAE, PSNE, SSIM and LPIPS metrics.

1 Introduction

With the emergence of various decision making systems and the increased emphasis on predicting what the future holds, Video prediction has gained immense attention from the scientific community. The most popular application to video prediction is autonomous driving. While the task may seem trivial and too simple to humans, it is increasingly difficult for a machine to predict the next state of the environment. Given a sequence of N frames extracted from a video sequence, the task is to predict the next M frames as precisely as possible. This is achieved by learning representations from previous video frames using an encoder-decoder style network along with Convolutional LSTMS in order to generate the next frames. The results depend on how good the learned representations are and how much has the model learned during training. In this report we try and successfully solve the task of Video Prediction as mentioned. In Section two, we discuss in detail our model architecture and the enhancements to it. Section 3 then goes on to talk about the method used for training, the datasets used and the various experiments that were conducted. Finally, Section 4 and 5 summarize the results obtained and the conclusion.

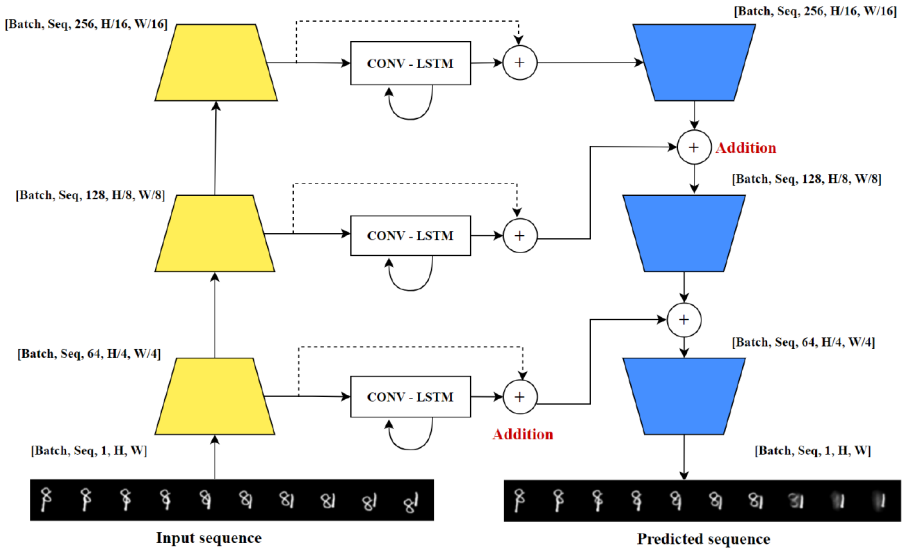


Fig. 1: Model Architecture

2 Model Architecture

Our model mainly has three components, the encoder architecture, the predictor and the decoder architecture. The encoder and decoder architectures consists of three encoder and decoder blocks respectively. The predictor consists of Convolutional LSTMs. The model has a three layer architecture, where each layer is connected sequentially as follows - an encoder takes an input image, downsamples it and passes the feature embeddings to a Convolutional LSTM which then passes the predicted embedding to the decoder in that specific layer. The decoder then up samples the output image. Each layer is connected to the next through encoders and decoders. Each encoder block sends the feature embedding to the next encoder block and to a Convolutional LSTM in its layer. The feature embedding from the previous decoder, the predicted embeddings from the Convolutional LSTM and an additional skip connection from the encoder to the decoder carrying the context frame[1] are added together to provide the input for the decoder. The decoder then processes the input and passes them on to the next decoder in the next layer.

The initial sequence of frames extracted from the videos are of length 20. These are fed into the encoder, Convolutional LSTMs and decoders sequentially to generate our desired output which also consists of 20 frames, wherein the first 10 are ground truth and the next 10 are predicted frames. The model is explained in detail below. Figure 1 gives a general overview of our model architecture.

2.1 Encoder

The integrated encoder starts with a 3x3 convolution followed by batch normalization and a ReLU activation which increases the channel length from one to 16. This convolutional layer is then followed by three encoder blocks where the first block has twice the capacity as the other blocks. Each encoder block has three convolutional layers. The first two of these convolutional layers are 3x3 convolutions with stride one and kernel size three, these convolutions are each followed by 2D batch normalization and a ReLU activation function. A skip connection carries over the initial input and adds it to the output of the second batch normalization. A ReLU activation is then applied. This overview of the encoder is depicted in figure 2a.

Finally a third 1x1 convolution is applied with kernel size as one and stride as two to downsample the image and increase the channels as required. It is noteworthy to mention that since the first encoder has twice the capacity as the other encoders it essentially has six convolutions followed by batch normalization and ReLU activations (without any ReLU activation after the 1x1 convolutions). The first encoder increases the channel information by a factor of four - first to 32 and then to 64. The second and third encoders increase the number of channels to 128 and 256 respectively.

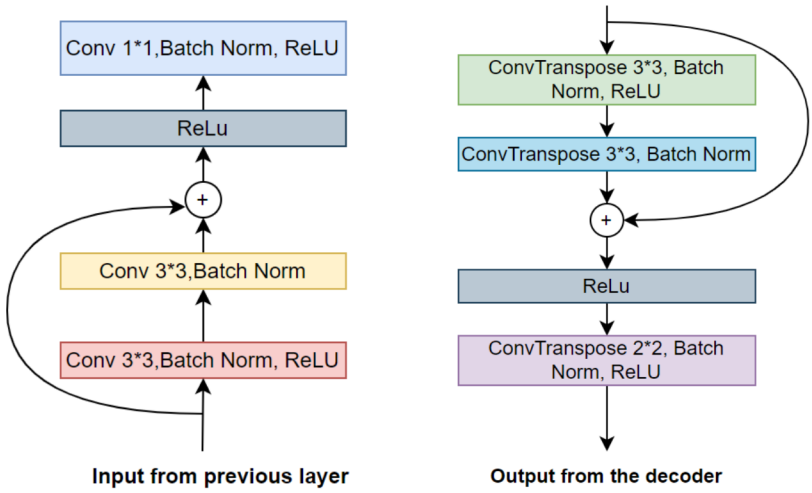
Each block downsamples feature embedding by a factor of two while the first block downsamples by a factor of four. The size of the feature embeddings are of size 16x16 after the first encoder, 8x8 and 4x4 after the second and third encoder respectively. These blocks are residual blocks inspired from the Resnet architecture. The output of the integrated encoder returns a list of outputs from each encoder.

2.2 Predictor

The predictors used in the model are Convolutional LSTMs. LSTM or Long Short Term Memory is a recurrent neural network that passes a previous hidden state of a sequence to the next step of the sequence. It uses information that it has gathered previously to make decisions. A Convolutional LSTM consists of cells that have hidden and cell states and uses convolutions to calculate them. The equations needed to calculate the hidden and cell states for the Convolutional LSTM are as follows [5].

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o) \\ h_t &= o_t \circ \tanh(c_t) \end{aligned} \tag{1}$$

where i_t : Input gate, f_t : Forget gate, c_t : Cell state, o_t : Output,h_t : Hidden state, $\circ$: Hadamard Product



(a) Encoder architecture

(b) Decoder architecture

Fig. 2: Encoder-Decoder detailed structure.

All convolutional LSTMs in the model have two layers and a kernel size of 3x3. The input from the encoder is added to the hidden states of the ConvLSTM to generate predictions. The input that is passed to the ConvLSTM is a 5D tensor of shape [batch size,seq length, channels, width, height]. At each iteration, all the first frames (ground truth) of a batch are passed to the ConvLSTM to predict the next frame and then the second frames of a batch are passed and so on. For the first ten frames (the ground truths) in the sequence of 20, we pass the frame as the input to the ConvLSTM and calculate the hidden and cell states. For the next 10 frames which are to be predicted by the ConvLSTM ,the previous hidden state is passed as the input to the ConvLSTM to calculate the current hidden and cell states. The outputs of the three Convolutional LSTMs are stored in a list format again.

2.3 Decoder

The decoder is a mirrored version of the encoder which also contains three decoder blocks. Each decoder block upsamples the image by a factor of two except the last decoder block which upsamples by a factor of four as it has twice the capacity of the other blocks. Each decoder block has three Convolutional layers followed by batch normalization and ReLU activation functions as shown in figure 2b

The first two Convolutional layers apply 3x3 transposed convolutions with kernel size as three and stride as one, these convolutions are each followed by 2D

batch normalization and a ReLU activation function. Similar to the encoders, a skip connection carries over the initial input and adds it to the output of the second batch normalization . Finally a third 2x2 convolution is applied with kernel size as two and stride as two to upsample the image and decrease the channels as required. Again, as in the case of encoders, the last decoder has twice the capacity of the other decoders. The first decoder has an input with channel size of 256 as increased by the last encoder. The channel size is then decreased by the first, second and third encoder to 128, 64 and 1 respectively. The image size is also increased by a factor of two for the first two decoders and by a factor of four for the last decoder. The input to the first decoder is an image of size 4x4 which is then upsampled by the first, second and third decoder to sizes of 8x8, 16x16 and 64x64 respectively.

2.4 Context Frame Addition

To achieve better results and taking inspiration from [1], we proceeded to add skip connections from the encoder to the decoder. The output from the encoder at each layer of the model is added to the ConvLSTM predictions and the previous decoder outputs to achieve better results. Significant improvements were noticed after the addition of the context frames. The reasoning behind this addition is that these skip connections help in recognizing the digits faster and further speed up the processing. This is explained in section 4.1.

3 Experimental details

In this section we inspect the datasets used in the project and the training procedure. We also look at changes made to the baseline model and analyse how these changes helped our task of video frame prediction. We further evaluate the models using quantitative and qualitative metrics. Some of the quantitative metrics used are : MSE, MAE, PSNR, SSIM and LPIPS.

3.1 Datasets

Moving MNIST: For developing our model, we create our own synthetic Moving MNIST dataset which consists of two random digits moving in a 64x64 layout consisting of 20 frame sequences.

KTH Dataset: The KTH dataset [4] consists of 25 subjects performing six type of actions namely walking, jogging, running, boxing, hand waving and hand clapping. There are a total of 2391 video sequences. The sequences have a resolution of 160x120 pixels which were further transformed to 64x64 in our case in order to ease computations.

3.2 Training

The entire architecture is implemented with Pytorch [2]. All models are trained for 100 epochs with the Adam optimizer and an initial learning rate of $1 * 10^{-3}$. We use the ReduceLRonPlateau scheduler which down samples the learning rate by a factor of 0.1 when it sees no improvement for 10 epochs. This helps us in learning when the model reaches a plateau stage and there is no improvement seen on further epochs. We also make use of the ExponentialLR for our experiments.
All experiments performed with the combined loss are trained using learning rate warmup. This helps the model to avoid overfitting and helps the optimiser to calculate the gradients correctly instead of jumping in the wrong direction at the start. Learning rate warmup starts with a very low value and increases linearly for a some defined epochs.In our experiments the learning rate warmup starts from 0 and increases linearly in the initial 10 epochs (warm up state) and then goes to the original learning rate of $1 * 10^{-3}$.
We feed in a sequence of 20 frames as input which are of the shape [batch size, 20, 1, 64, 64] from which the first 10 frames are used as our ground truth. The sequence first goes through the encoder to generate embedded frames, these embedded frames are then passed onto the ConvLSTM to generate predictions. This sequence is then passed on to the decoder initially with the shape [batch size, 20, 1, 4, 4], the decoder decodes it at various levels to generate outputs of the shape [batch size, 20, 1, 64, 64] where the first 10 frames are the ground truth and the next 10 are the predicted. The loss is then calculated using Mean squared error and back propagated.

3.3 Criterion/Loss for training

We essentially use mean squared error for training in the beginning which is given by equation 2 where N is the number of samples used and y_i and $\hat{y_i}$ are target and predicted pixel values respectively. We then go on to experiment using mean average error (L1 loss). A downside to using MSE loss is that it assumes pixel independence. To the human eye, structural differences are more valuable than pixel differences which the MSE computes.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{2}$$

To try and mitigate this, we combine mean squared error (MSE) with structural similarity index measure (SSIM). Structural similarity index measure makes use of mainly three components to capture similarity, namely luminance, structure and contrast [6]. Since, these three characteristics make up to what we as humans perceive, it makes sense to use this metric as a loss criterion to train our models. Equation 3 show how SSIM is calculated, where $l(x, y)$, $c(x, y)$ and $s(x, y)$ correspond to the luminance, contrast and structural properties respectively. The

constants α, β and γ correspond to the parameters to adjust which property to give importance to.

$$SSIM(x, y) = [l(x, y)]^{\alpha}.[c(x, y)]^{\beta}.[s(x, y)]^{\gamma} \tag{3}$$

Since this is a similarity metric, we make use of equation 4 to compute the Loss.

$$L_s(x, y) = 1 - SSIM(x, y) \tag{4}$$

In order to combine MSE with SSIM as depicted in equation 5, the value of λ is chosen to be 0.001 for it to be comparable to the MSE loss. We made use of the PIQA [3] library to compute the loss.

$$L(x, y) = MSE(x, y) + \lambda * L_s(x, y) \tag{5}$$

3.4 Experiments

In this section the various experiments conducted are summarized. The experiments are conducted by changing a few key parameters in the training of the model. The key parameters that we change are the loss functions that are used in training, the learning rate schedulers and learning rate warmup. The experiments have been conducted for all possible combinations of MSE, MAE, MSE+SSIM, MAE+SSIM for the training loss with ExponentialLR and ReduceLROnPlateau as schedulers and using learning rate warmup.
Ablation Study: Two experiments have also been conducted as part of an ablation study on the Moving MNIST dataset. The first of these experiments was to remove the last context frame addition to the model from the encoder to the decoder. The next experiment was to remove the skip connections inside the decoder/encoder blocks (using vgg like blocks instead of resnet blocks).

4 Results

This section summarises the results of the above experimental setting both qualitatively and quantitatively.

4.1 Qualitative Analysis

Moving-MNIST Results : We experiment on this dataset using different losses and schedulers and come up with a best performing model. To explain and justify our results well, figure 3 depicts a snippet of one of our results on a sequence on different models. However, it is important to note that this is only a part of the results obtained, and there were many instances in which our best performing model predicted both images correctly and precisely as shown in section 4.3.
Using MSE loss with ReduceLROnPlateau schedulers gives us satisfactory results till the the 5th predicted frame. As the digits touch and cross over each

other, it looses one of the digits and doesn't predict it well.
To remove this issue that we encountered we used a combined loss function combining SSIM with MSE as explained in equation 5. This helps us in achieving at least one digits structure, position and movement well. Hence, we deduce that adding a combined loss function helps us achieve better results.
Experimenting with Mean average error(MAE) loss gave us sharper results. The

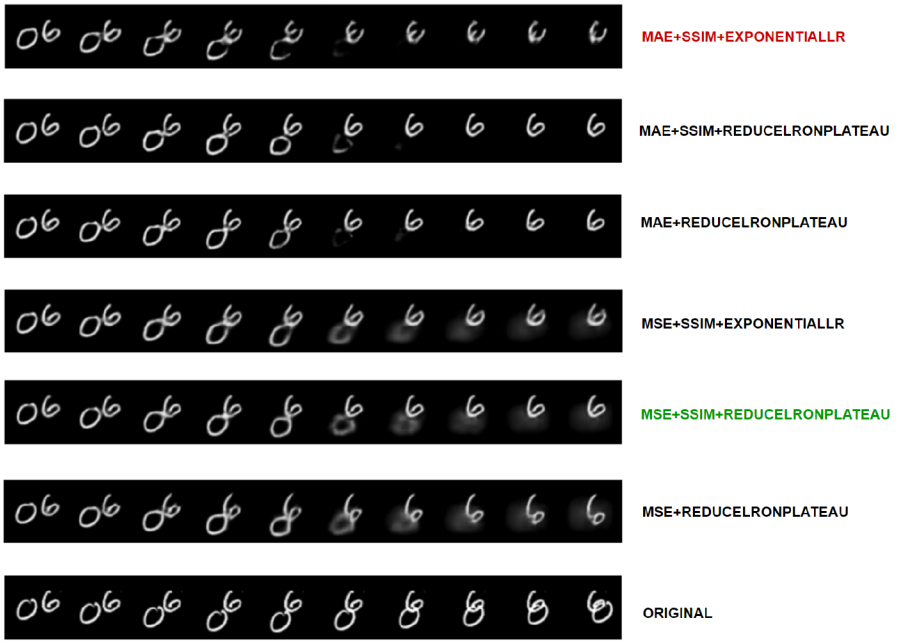


Fig. 3: Moving-MNIST Results. Results marked in green and red are the best and worst performing models respectively.

digits were more defined as shown in figure 3. However, it fails to acknowledge the fact that a second digit even exists towards the end and predicts a completely black background. With MSE, the model at least would acknowledge that a second digit exists by having a blurry grayish background. Combining MAE loss with SSIM makes very little difference here in the images obtained.
We go on to prove that using ReduceLROnPlateau scheduler proves to be the best as opposed to using ExponentialLR scheduler. The latter gives us very blurry and distorted images. This is because ExponentialLR scheduler changes the learning rate for each epoch by multiplying the previous learning rate by a factor (gamma). A gamma with value of 0.7 was used in the experiments so the learning rate decreases by a huge factor as the number of epochs increases and so the model learns very slowly in these epochs.

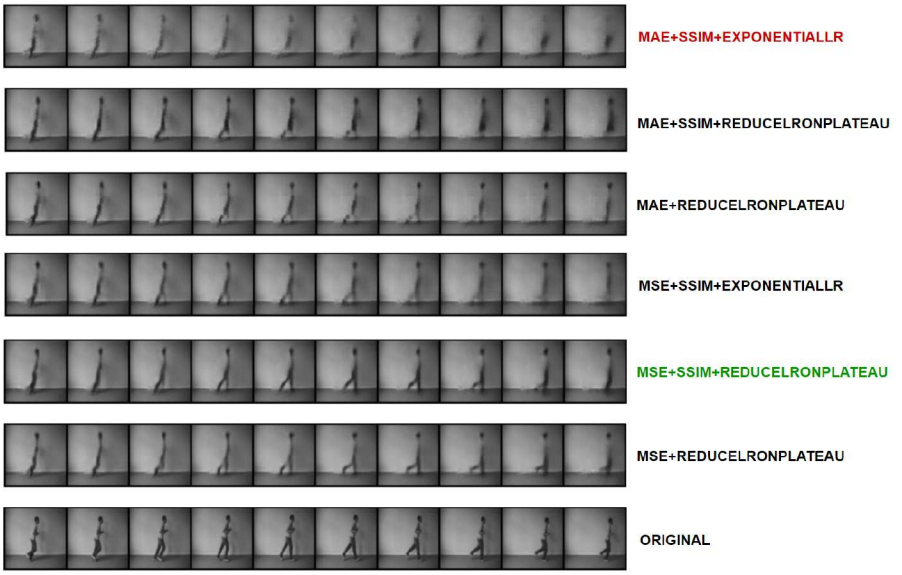
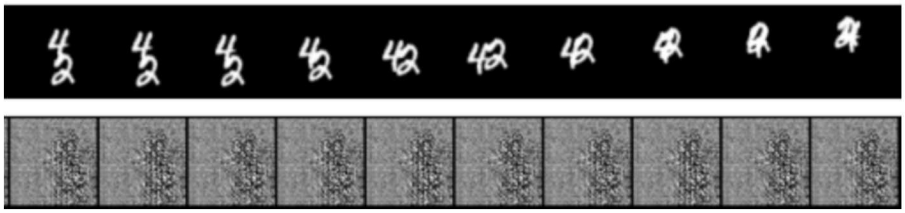


Fig. 4: KTH Results. Results marked in green and red are the best and worst performing models respectively.

KTH Action results: As done with the case of Moving-MNIST dataset, we make use of different loss functions and schedulers to get to the best performing model. This dataset is complicated as compared to the other as it had many frames which were blank in which no actions were being performed.
With the use of MSE loss in this case, it predicts the movement of the person well as depicted in figure 4. This is a difficult task to achieve and the fact that the model is able to predict leg movements till the last frame accurately is commendable. However, if we go on to add SSIM loss to MSE, we obtain much sharper and better quality results. The most significant improvement can be seen in the colour of the pants of the person moving. It appears to be closer to the original target sequence in case of luminance and contrast properties of images. Since SSIM primarily works on luminance, contrast and structure, using it proves to be of use to us.
MAE loss also works well and obtains close results, however, it fails to predict the leg movements in the last three frames. Also, as opposed to the case of Moving-MNIST dataset, MAE does not yield sharper results.
As the case of Moving-MNIST, ExponentialLR fails and produces very blurry and distorted images due to the same reasons given above.

Context Frame Addition: With the addition of context frames, that is adding the last context frame obtained from the encoder to the output of the ConvLSTM for each layer and then feeding it to decoder helps in achieving the localization of



(a) Moving-MNIST results after first epoch with the addition of context frames.

(b) Moving-MNIST results after first epoch without the addition of context frames.

Fig. 5: Moving-MNIST with and without context frames. The first line depicts the ground truth and the second line corresponds to the predicted frame results.

the digits faster. Figure 5a shows the results after the 1st epoch with the addition of context frames whereas Figure 5b shows the results without the addition of context frames after 1st epoch. There is clearly a distinction between the two as with the addition of context frames, after the first epoch itself the model does learn that the digits exist and hence we go on to include context frames in all of our further experiments.

4.2 Quantitative Analysis

Moving-MNIST dataset Analysis Table 1 refers to results obtained on training the model with different loss criterion and schedulers on the Moving-MNIST dataset for a total of 50 epochs. The best results are obtained when the model is trained with a combined loss, i.e. MSE with SSIM loss along with using the ReduceLROnPlateau Scheduler that decreases the loss by a factor of 0.1 if the model plateaus for a total(patience level) of 10 epochs which helps in improving results if the model reaches a plateau stage.

Mean Squared Error and Peak Signal to Noise Ratio (PSNR) are closely related as PSNR is the logarithmic inverse of MSE. This relationship can be observed as highest PSNR value was observed for models that obtained the lowest MSE. SSIM measures the structural similarity of the predicted frames. The highest observed is 0.792 while the lowest measured is 0.614. Our best model achieves low SSIM, but this can be explained by the example given in figure 3. Frames predicted with MAE as loss criterion disregard the second digit sometimes which leads to perfect black backgrounds as opposed to the ones generated using MSE

Moving-MNIST Results					
Models	**MSE**	**MAE**	**PSNR**	**SSIM**	**LPIPS**
MSE+ReduceLROnPlateau	0.029	0.065	15.715	0.614	0.214
MSE+SSIM+ReduceLROnPlateau	**0.028**	0.061	**15.806**	0.650	**0.194**
MSE+SSIM+ExponentialLR	**0.028**	0.063	15.792	0.631	0.210
MAE+ReduceLROnPlateau	0.034	**0.046**	15.043	0.788	0.418
MAE+SSIM+ExpScheduler	0.034	0.047	14.900	**0.792**	0.294
MAE+SSIM+ReduceLROnPlateau	0.034	0.046	14.909	0.790	0.4276

Table 1: Summary of results for Moving-MNIST Dataset

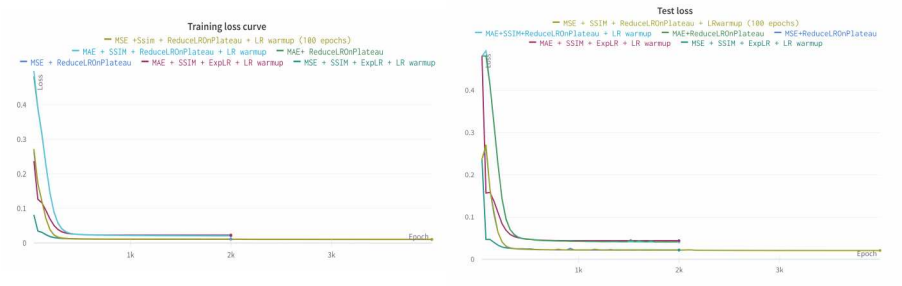
loss that have grayish noise in the image. We suspect this to be the reason that SSIM metric performs well with MAE loss in the case of Moving-MNIST dataset.

Lastly we compare the results from table 1 with the results for the ablation study conducted on the Moving MNIST dataset trained with MSE and SSIM loss functions with ReduceLROnPlateau Scheduler with LR Warmup. For the experiment without context frame addition we found that the results for the PSNR metric is 15.677 and SSIM metric is 0.641. And for the second experiment we found that the result for PSNR is 14.306 and SSIM is 0.594. It is noteworthy to mention that the second experiment gives us the lowest SSIM values in all our experiments. We also see that the same model (second row) in Table 1 has much better results when it has context frame addition and skip connections. This effectively justifies our decision to use context frame addition and skip connections inside encoder/decoder in our model architecture.

KTH Results					
Models	**MSE**	**MAE**	**PSNR**	**SSIM**	**LPIPS**
MSE+ReduceLROnPlateau	0.035	0.156	15.240	**0.77**	0.241
MSE+SSIM+ReduceLROnPlateau	0.043	0.179	14.373	**0.77**	0.239
MSE+SSIM+ExponentialLR	0.037	0.168	14.732	0.754	0.249
MAE+ReduceLROnPlateau	**0.033**	0.146	**15.426**	0.73	**0.225**
MAE+SSIM+ExponentialLR	**0.033**	**0.143**	15.310	0.70	0.248
MAE+SSIM+ReduceLROnPlateau	0.039	0.166	14.561	0.71	0.242

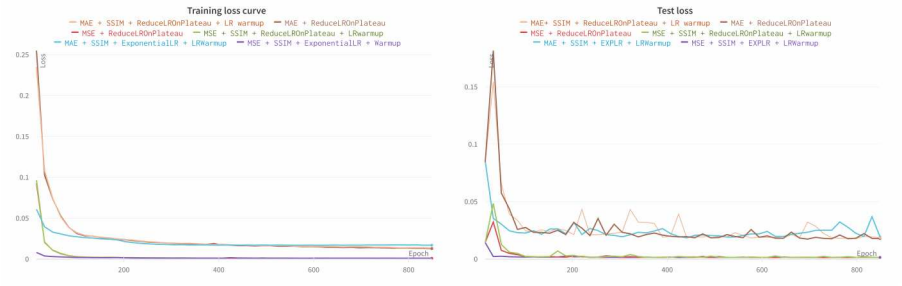
Table 2: Summary of results for KTH Dataset

KTH Action dataset Analysis Table 2 depicts the quantitative results on the KTH dataset. The best performing model is trained using MSE and SSIM as a loss function along with ReduceLROnPlateau scheduler. This model achieves the highest SSIM measure of 0.77 and high a high PSNR value too, although not the highest. LPIPS value is also the second best (0.239) for this model. Qualitatively, this model gives us best results. Models trained on MAE produce low MSE values, which is surprising to us. Although using ExponentialLR scheduler gives us low MAE and MSE values, qualitatively these produce the worst results.



(a) Training Loss Curve (b) Test Loss Curve

Fig. 6: Summary of train and test plots for various models on the Moving-MNIST dataset



(a) Training Loss Curve (b) Test Loss Curve

Fig. 7: Summary of train and test plots for various models on the KTH dataset

The training and test loss curves for the various experiments conducted on the Moving-MNIST dataset are shown in Figure 6. We see that there is a significant difference in the train and test loss curves for models trained on MSE as well as SSIM criterion as compared to models trained on the MAE criterion.
Loss curves for the KTH action dataset are depicted in figure 7. A very low loss on both training and test data is observed here as compared to Moving-MNIST results. This can be attributed towards the similar pixel values (grays) in the KTH action dataset as compared to Moving MNIST which had clear black or white colour pixels.

4.3 Best Model Results

The model that performs best in all our experiments for both datasets is the model trained with MSE+SSIM loss with ReduceLROnPlateau Scheduler and LR warmup. For the Moving MNIST dataset the model predicts the shape and location of both digits in the frame for the case when the digits never overlap with each other. In case overlap occurs the model usually loses track of one digit

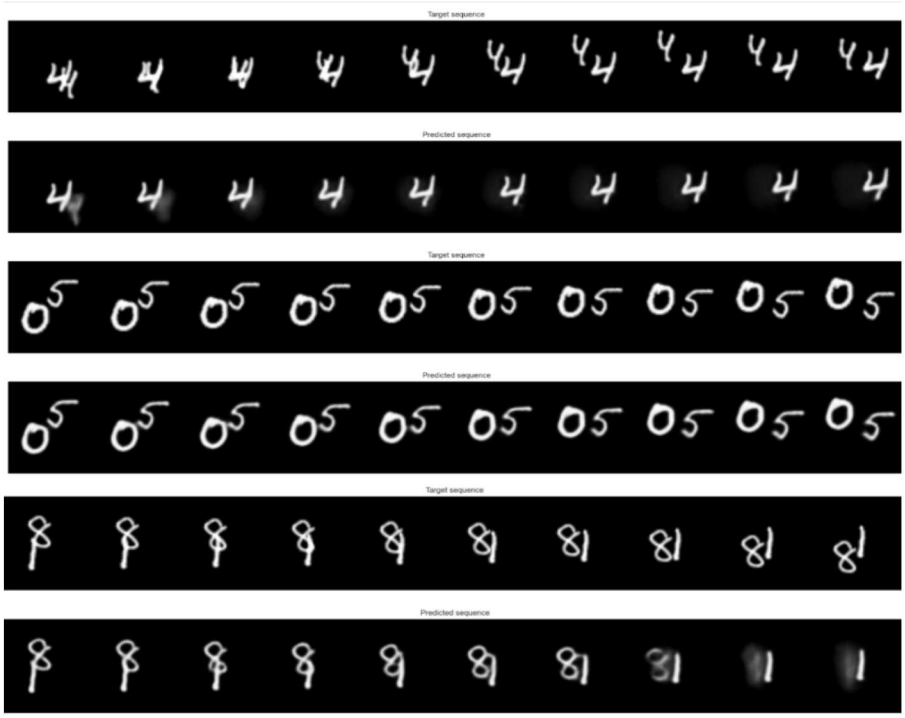


Fig. 8: Moving MNIST best model results. The first and second line depict the target and predicted sequences respectively.

and is able to predict only one of the digits perfectly. In some cases, the second digit is sometimes seen as a white blur. Hence, we deduce that our model is not able to detect two digits well after an overlap as overlapped structure of two digits is very different to the individual digits that the model usually predicts. The results are depicted in figure 8.

For the KTH dataset the model consistently delivers good results as depicted in figure 9. It is able to predict the location of a person in a frame and their posture perfectly. However in instances where the person is performing an action using their arms (e.g. hand waving,boxing, hand clapping), the model predicts changes in the movement of the arms, however it is not able to predict the length and size of the arms very clearly, they appear to be a bit blurred. This is due to the fact that the model calculates the predicted frames pixel-wise and so the model is not able to predict the arms that occupy much less pixels when compared to the rest of the person's base figure. It should also be noted that the further the arms are extended the further they are from the person's base figure and the model finds it hard to predict parts of the person that are so far away from their base figure against the background which has very different intensity values for the pixels.

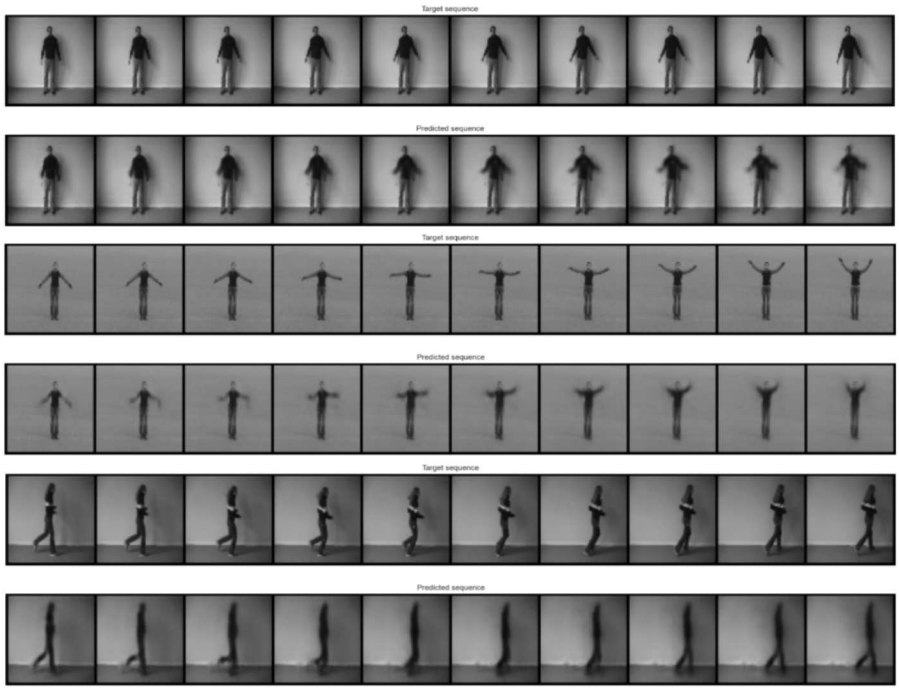


Fig. 9: KTH Action best model results. The first and second line depict target and predicted sequences respectively.

5 Conclusion

A model that properly predicts spatial arrangement of digits and actions of humans was implemented successfully. The exact baseline model with encoder-decoder architecture and Convolutional LSTMs as predictors was used. We further added skip connections that emerge from the encoder to the decoder carrying the last context frame which helps in modeling the localization of the digits faster. We also use ResNet like residual blocks in the encoders as well as the decoders. The model is successfully trained on the synthetically generated Moving-MNIST dataset as well as on the KTH Dataset. The model performs quite well on both datasets with some exceptions. For the Moving-MNIST dataset it sometimes fails to distinguish between two digits, especially when the digits overlap as for the KTH Dataset, it fails to recognize hand movements when they are further away from the ground truth. (e.g if the hand action in the ground truth starts at the 17th frame, its difficult for the model to realise that action). This failure is mostly noticed for frames that contain hand movements. Leg movements are usually predicted correctly. In either cases, our model has far too many instances where it predicts correctly and as precisely as possible for both datasets.

6 Contributions

The code for model architecture were mostly written by Vardeep and Sugan. Dataloaders were created by Aysha. Discussions and brainstorming for ideas was done regularly (at least twice a week) in which all three of us participated. Some of the code was jointly written by the three of us.
Conducting the experiments and creation of the report were done together by the three of us. Generation of figures in the report are done by Aysha.

References

1. Francesco Cricri, Xingyang Ni, Mikko Honkala, Emre Aksu, and Moncef Gabbouj. Video ladder networks. *CoRR*, abs/1612.01756, 2016.
2. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf*, pages 8024–8035, 2019.
3. François Rozet. Piqa. `https://github.com/francois-rozet/piqa`.
4. C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36 Vol.3, 2004.
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. *https://arxiv.org/abs/1506.04214*, 2015.
6. Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Trans. Img. Proc.*, 13(4):600–612, apr 2004.