VIDEO PREDICTION: MITIGATING BLURRED PREDICTIONS

Team Members: 1. Shubham Wasnik (17894)
2. Palash Kamble (17853)

1. ABSTRACT

Video prediction is the task of predicting future frames given past frames. For given sequence of frames, we wish to build such a model which can learn to give hidden representation of this sequence, and then this hidden representation can be used to predict future frames. Using standard Mean Square Error, blurred predictions are obtained. So we aim to handle blurry predictions by doing some experiments with loss function. We tried adding regularization in the MSE loss which still gave blurry predictions, then we tried using cross entropy loss, which resulted in lesser blurry predictions. The model architecture we used is sequence to sequence model containing ConvLSTM cell.

2. TECHNICAL DETAILS

The model type we used is called seq2seq model which is typically used for NLP or time-series applications. Since video prediction comes under the category of time-series modelling (we need to model future frames), we choose to work with seq2seq model architecture. The typical implementation of any time-series task includes LSTM architecture suitable for sequences such as words, letters. But since we are working with video containing sequence of frames/images, the better architecture would be convLSTM.

2.1. Seq2Seq Model Architecture

Seq2seq model takes a sequence of frames as input and outputs another sequence of frames which we call it as predicted frames.
We separate this model into three parts:

- Encoder: Encodes given sequence of frames where each frame is fed to the convLSTM cell.
- Encoded Embedding Vector: Hidden representation obtained after passing all sequence of frames to the Encoder.
- Decoder: The encoded embedding vector is fed to the convLSTM cell architecture to predict/decode future frames

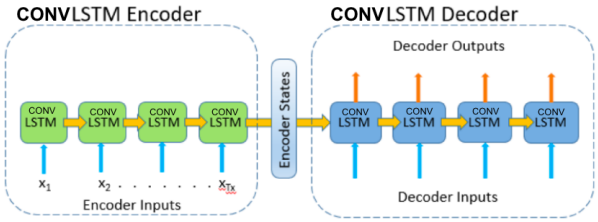

Fig. 1. Seq2Seq model architecture

2.2. ConvLSTM Cell Architecture:

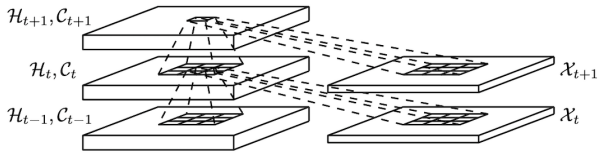

Fig. 2. ConvLSTM Architecture

The ConvLSTM cell has three gates:
1. Input Gate: i_t
2. Forget Gate: f_t
3. Output Gate: o_t

The equations for these gates are given below

- $i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i)$
- $f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f)$
- $C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c)$
- $o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o)$
- $H_t = o_t \circ \tanh(C_t)$

Video is a spatiotemporal sequence that is it has both spatial and temporal correlations we need such an architecture to model these both correlations. Since LSTMs are used to

model temporal relations in any given sequence, spatial relations are handled by convolution operations, we use convLSTM architecture which is capable of handling both spatial and temporal correlations for given video sequences. The most important featur of convLSTM is that it can handle long range dependencies between frames/sequences.

2.3. Dataset

We worked on ucf101 dataset which contains short video clips across many topics, ex. Cricket, Boxing, Badminton, etc. Since, the dataset size is huge (more than 7 GB), and to avoid time taken to train model, we used the top 5 videos from categories - Cricket, Boxing, PlayingCello, Tennis, and ShavingBeard.

2.3.1. Data Pre-processing

- From each video, we took the first 20 frames.
- Each frame size was resized to 128x128.
- For each video, we created X (set of input frames) and y (set of true frames) tuple. Since, we are doing first frame prediction, we assigned first 19 frames to X, and the remaining last frame to y.
- The tuple (X,y) is now ready to be fed to the model.

3. EXPERIMENTS AND RESULTS

3.1. Experiment 1

In this experiment, we tried using **sgd** optimizer, and standard MSE loss. We trained the model for following parameters:

bSize	hDim	lRate	Epochs	Optimizer	Loss
8	16	0.01	50	sgd	MSE

Here, bSize is batch size, hDim is hidden vector dimension, and lRate is learning rate.
Following is the loss plot and first frame predictions on test dataset (shown upto 5 random videos from test set).

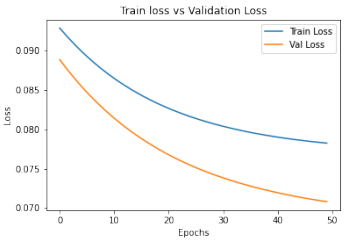

Fig. 3. Exp1: Loss plot

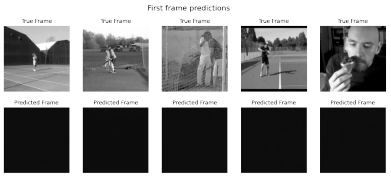

Fig. 4. Exp1: First frame predictions

As we can see from figure 4, the top row is the 20th true frame for five videos from test dataset, and the bottom row is their corresponding predicted frame. We see that we didn't even get the predictions.

3.2. Experiment 2

In this experiment, we tried using **adam** optimizer, and standard MSE loss. We trained the model for following parameters:

bSize	hDim	lRate	Epochs	Optimizer	Loss
8	16	0.01	50	adam	MSE

Here, bSize is batch size, hDim is hidden vector dimension, and lRate is learning rate.
Following is the loss plot and first frame predictions on test dataset (shown upto 5 random videos from test set).

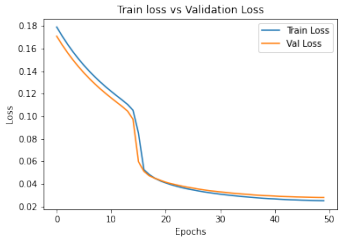

Fig. 5. Exp2: Loss plot

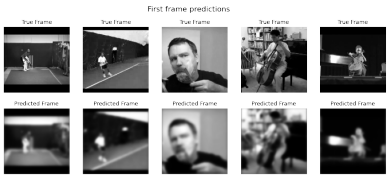

Fig. 6. Exp2: First frame predictions

As we can see from figure 6, we started getting predictions and as expected the predictions were blurry

Following is the ssim score obtained for this experiment:

AverageSSIM	MinimumSSIM	MaximumSSIM
0.43	0.16	0.72

3.3. Experiment 3

In this experiment, we tried using **adam** optimizer, and standard MSE loss with L2 Regularization. We trained the model for following parameters:

bSize	hDim	lRate	Epochs	Optimizer	Loss
8	16	0.0001	100	adam	MSE+L2

Here, bSize is batch size, hDim is hidden vector dimension, and lRate is learning rate.
Following is the loss plot and first frame predictions on test dataset (shown upto 5 random videos from test set).

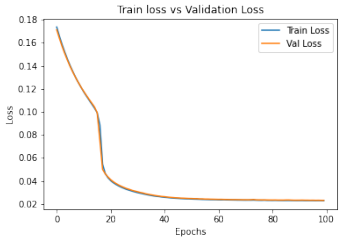

Fig. 7. Exp3: Loss plot

Fig. 8. Exp3: First frame predictions

As we can see from figure 8, we started getting predictions and as expected the predictions were blurry but better than the experiment 2

Following is the ssim score obtained for this experiment:

AverageSSIM	MinimumSSIM	MaximumSSIM
0.63	0.23	0.88

3.4. Experiment 4

In this experiment, we tried using **adam** optimizer, and cross entropy loss. We trained the model for following parameters:

bSize	hDim	lRate	Epochs	Optimizer	Loss
8	16	0.0001	100	adam	CSELoss

Here, bSize is batch size, hDim is hidden vector dimension, lRate is learning rate, and CSELoss is cross entropy loss.
Following is the loss plot and first frame predictions on test dataset (shown upto 5 random videos from test set).

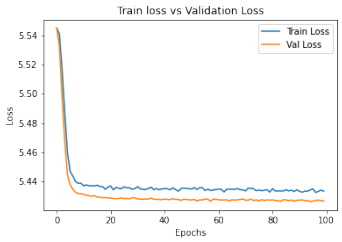

Fig. 9. Exp4: Loss plot

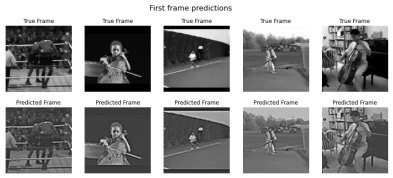

Fig. 10. Exp4: First frame predictions

As we can see from figure 10, we got much better predictions compared to previous experiment with very less blur.

Following is the ssim score obtained for this experiment:

AverageSSIM	MinimumSSIM	MaximumSSIM
0.91	0.73	0.98

4. CONCLUSION

We have seen that Mean Square Error Loss(MSE) gives blurry results. We tried MSE loss with stochastic gradient descent optimizer which did not give any predictions, In the next experiment we tried MSE loss with Adam optizimer, it started giving output and as we expected, the predictions were blurry. In the next experiment we added L2 regularizer with MSE loss keeping the same optimizer(Adam), it resulted into better predictions than previous experiment, with some blur. Finally we changed the loss function and switch to cross entropy loss which resulted in much better predictions with very less blur. Link to codes code