Predicting and Generating Video Sequences Using Deep Learning

Ali Ashraf (21I-0605)
Department of Computer Science
FAST NUCES
Islamabad, Pakistan
i210605@nu.edu.pk

Muhammad Umais (20I-0854)
Department of Computer Science
FAST NUCES
Islamabad, Pakistan
i200854@nu.edu.pk

***Abstract*—This project focuses on developing deep learning models to predict and generate future frames in video sequences using the UCF101 dataset. Through short input sequences, the models learn to simulate ongoing actions and thus produce coherent and realistic continuations of videos. This project develops three deep learning approaches toward this goal: ConvLSTM, PredRNN, and a Transformer-based model. The produced frames are compiled into video segments, and a user interface is developed to represent the input, predictions, and full sequences. The performance of the models is tested using metrics like Mean Squared Error (MSE) and Structural Similarity Index (SSIM).**

I. Introduction

A. Problem Definition

The project develops a system able to predict the subsequent video frames based on some short input sequence of video frames. The input frames would give some glimpse of the continuation of the action, and what the system is to work out would be to extract the patterns of motion present in those frames to come up with the continuation of the video. These predicted frames would seamlessly continue the input sequence, almost naturally continuing the action.

B. Significance

One important application of the ability to predict future video frames is enhancing the process of video compression through filling-in missing frames, manual animation effort reduction in video production and editing, and more futuristic uses such as in an autonomous vehicle where future movement is predicted to take proper decisions.

This capability further enhances the quality of video in stream media and enables real simulation scenarios in virtual environments. Generating coherent, realistic video sequences is hence how deep learning can enable bridging the understanding gap of human-like visual perception by replicating it.

C. Existing Approaches

Different approaches have been designed to overcome the issue of video frame prediction; they are all about exploiting spatial and temporal dependencies, though with different levels.

Convolutional LSTMs (ConvLSTMs): This approach is designed to capture the spatial and temporal patterns simultaneously. They are a combination of convolutional operations and Long Short-Term Memory units, thereby making them excellent for short-term predictions but not so for the long-term dependencies because of the sequential nature.

PredRNN: PredRNN enhanced mechanisms for the improvement of the time modeling abilities of the convLSTMs introduced and spatiotemporal memory flow. That results in a better fitting over longer sequences of intricate motions.

Transformer-Based Architectures: Transformers architecture has been transferred into the application of video frame prediction where, it has been quite nicely suited to the task and can model long-range dependencies at hand with efficiency even in complex cases of extended sequences and intricately moving objects.

D. Organization

This paper is organized as follows: Section II describes the proposed approach and implementation details, including dataset preprocessing and augmentation. Section III outlines the CNN architecture and training setup. Section IV presents performance evaluation metrics and experimental results. Finally, Section V concludes the report and provides references.

II. Proposed Approach/Implementation Details

A. Overview of Approach

We implement and compare three deep learning models: ConvLSTM, PredRNN, and Transformer-based architectures. The UCF101 dataset is used for training and evaluation.

B. Loading and Preprocessing the Dataset

1) Preprocessing Steps: We used the UCF101 video dataset for the model's training and we made the following changes: 64x64 frames to reduce input size for shorter training and prediction times. Frames were gray scaled to reduce channels from 3 to 1 and to reduce input size. These changes were made because input sizes of 256x256 and 3 channels were taking 10+ hours training times for PredRnn and LSTM, while 64x64 and 1 channel took 20 minutes to an hour, due to having limited resources we went the smaller input size.

2) Data Augmentation: Data augmentation techniques, such as rotation, flipping, and cropping, are applied to increase the diversity of the data set and improve the robustness of the model.

III. Model Architectures

A. LSTM Model Architecture

The architecture figure for LSTM is as below:

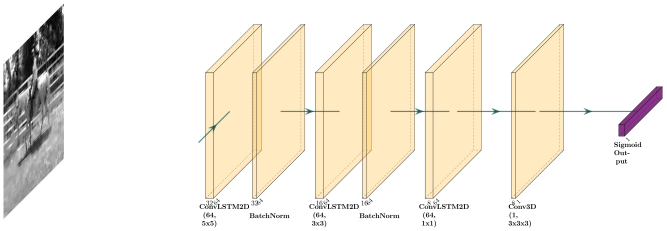

Fig. 1: LSTM Model Architecture

LSTM has the following layers: Input Layer: Shape = (2, 19, 64, 64, 1)

ConvLSTM2D Layer 1: 64 Kernels (5x5 Filter)
ConvLSTM2D Layer 2: 64 Kernels (3x3 Filter)
ConvLSTM2D Layer 3: 64 Kernels (1x1 Filter)
ConvLSTM2D Layer : 1 Kernel (3x3x3 Filter)
Sigmoid Activation Layer
Output

B. PredRNN Model Architecture

The architecture figure for PredRNN is as below:

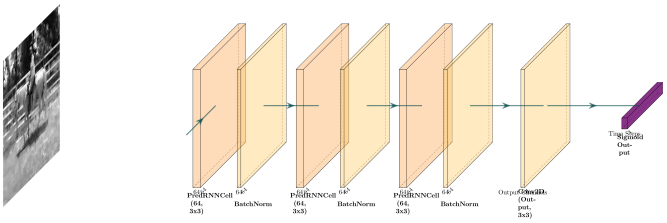

Fig. 2: LSTM Model Architecture

PredRNN has the following layers: Input Layer: Shape = (2, 19, 64, 64, 1)

PredRNN Cell Layer 1: 64 Kernels (3x3 Filter)
PredRNN Cell Layer 2: 64 Kernels (3x3 Filter)
PredRNN Cell Layer 3: 64 Kernels (3x3 Filter)
ConvLSTM2D Layer: 1 Kernel (3x3 Filter)
Sigmoid Activation Layer
Output

PredRNN is a custom recurrent unit with the following properties:

A custom recurrent unit designed for spatiotemporal data. Uses two convolutional layers (conv-x and conv-h): conv-x processes the input at each timestep. conv-h processes the hidden state from the previous timestep. Both layers have configurable filters, kernel size, stride, and ReLU activation. Outputs a new hidden state (h-next) by adding the results of conv-x and conv-h and applying ReLU activation. If no previous hidden state is provided, initializes it as zeros matching the input dimensions.

C. Transformer Model Architecture

The architecture figure the Vison-Transformer is as below:

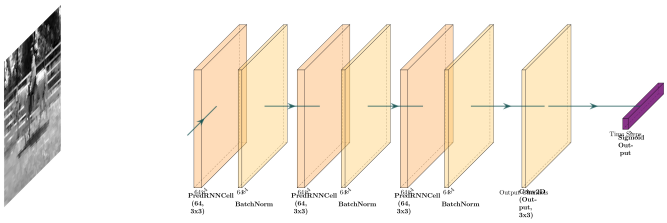

Fig. 3: LSTM Model Architecture

Transformer has the following layers:

A bunch of transformer block layers and a reconstruction layer.

Transformer Layers are consist of the following:

Multi-Head Attention: Captures relationships between patches across space and time. Feedforward Network (FFN): Processes the output of the attention mechanism. Layer Normalization: Stabilizes training. Dropout: Reduces overfitting. Residual Connections: Helps preserve information flow.

Reconstruction Layer:

Converts the transformed patch embeddings back into image frames using: Convolutional layers for feature processing. Transposed convolution for upscaling and reconstructing the original spatial dimensions.

D. Training Setup

The models are trained using the Adam optimizer and a learning rate of 0.001. Training is conducted for 20 epochs (LSTM), 50 epochs (PredRNN) and 60 epochs (Vision-Transformer) with early stopping and a batch size of 8.

IV. Performance Evaluation

A. Evaluation Metrics

The models are evaluated using Mean Squared Error (MSE) and Structural Similarity Index (SSIM) to measure prediction accuracy and visual quality.

V. Results

A. LSTM Model Performance

LSTM had the best result for our dataset due to it being the simplest model to train and implement and most suited for our dataset (short-form videos):

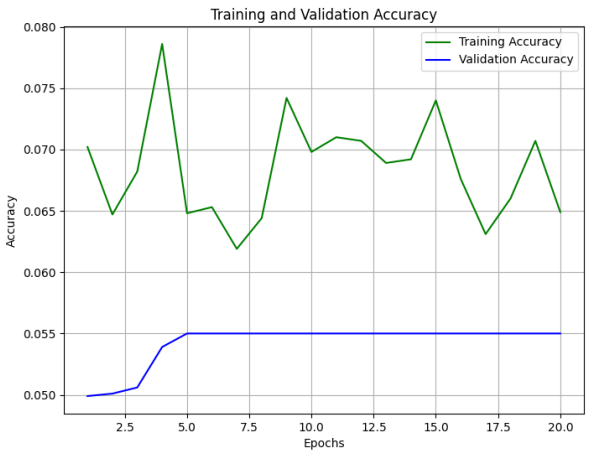

Fig. 4: LSTM Accuracy

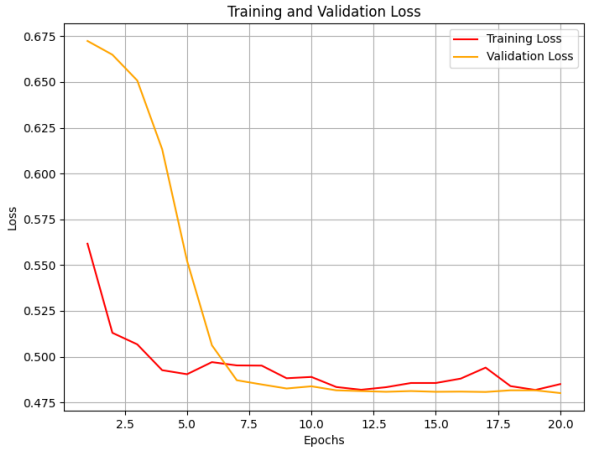

Fig. 5: LSTM Loss

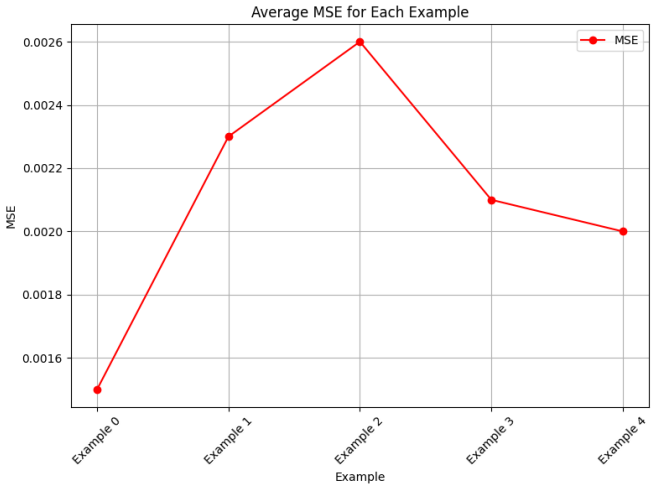

Fig. 6: LSTM MSE

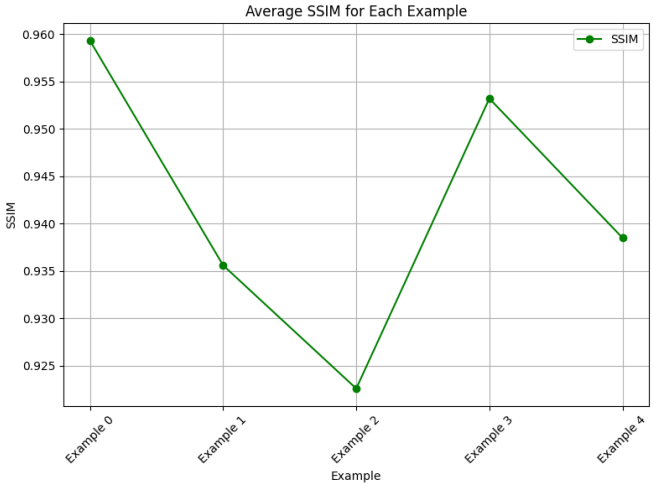

Fig. 7: LSTM SSIM

B. PredRNN Model Performance

PredRNN achieves better temporal modeling compared to ConvLSTM, producing smoother transitions between frames.

C. Transformer Model Performance

The Transformer-based model excels in capturing long-term dependencies, offering the most visually coherent results.

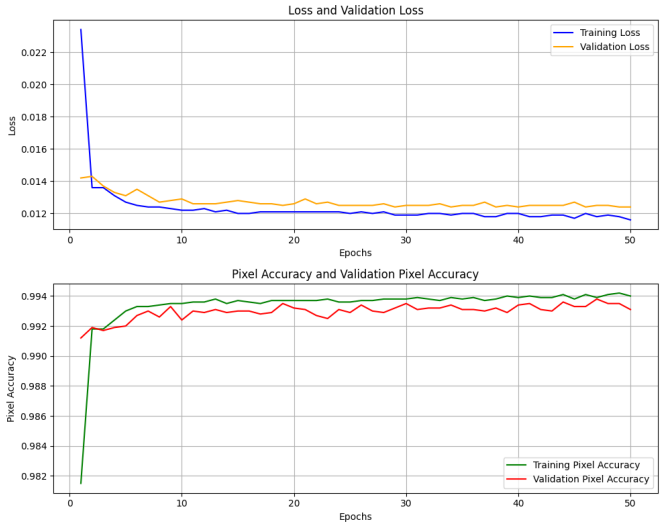

Fig. 8: PredRnn Results

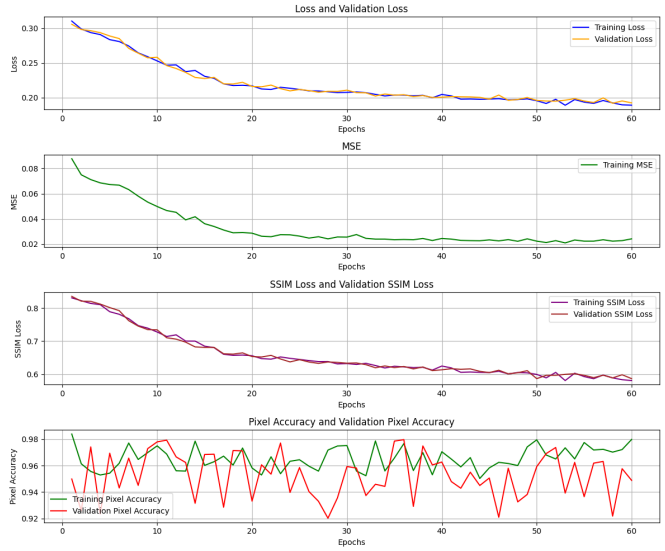

Fig. 9: Transformer Results

D. Comparison of Results

LSTM had the best result because the model was trained and implemented properly, followed by PredRnn with also visibly satisfying results. Transformers had blurry frames because training such an architecture takes hours (or days) worth of training and sizable resources.

VI. Conclusion

This project demonstrates the potential of deep learning models in video frame prediction. The comparative analysis shows that LSTM architectures are most effective for generating realistic video continuations (for short form videos and limited training time and resources).

References

1. J. Amogh, "Keras documentation: Next-Frame Video Prediction with Convolutional LSTMs," Keras.io. [Online]. Available: https://keras.io/examples/vision/conv$_lstm/$.

2. Y. Wang, Z. Gao, M. Long, J. Wang, and P. S. Yu, "PreDRNN++: Towards a Resolution of the Deep-in-Time Dilemma in Spatiotemporal Predictive Learning," arXiv.org, Apr. 17, 2018. [Online]. Available: https://arxiv.org/abs/1804.06300.

3. iamrakesh28, "Video-Prediction: Implementation of Transformer Encoder Decoder Architecture for Video Predictions," GitHub. [Online]. Available: https://github.com/iamrakesh28/Video-Prediction.