Next in Motion: Comparative Analysis of Deep Learning Models for Video Frame Prediction

Duaa Fatima
Dept. of Artificial Intelligence and Data Science
National University of
Computing and Emerging Sciences Islamabad, Pakistan
fatimaduaa053@gmail.com

Syeda Mahum Raza
Dept. of Artificial Intelligence and Data Science
National University of
Computing and Emerging Sciences Islamabad, Pakistan
mahum.raza4@gmail.com

Nayyera Wasim
Dept. of Artificial Intelligence and Data Science
National University of
Computing and Emerging Sciences Islamabad, Pakistan
nayyerawasim@gmail.com

Abstract—This work proposes to build a video prediction model that can generate frames that follow an input sequence of limited length from the UCF101 dataset, which includes various human activities recorded in video format. By using deep learning methods, the project builds and evaluates three state-of-the-art models: ConvLSTM, PredRNN, and Transformerbased architectures for forecasting multiple subsequent frames, replicating continuous motion. The errors and the performance of the models are assessed by Mean Squared Error (MSE) and Structural Similarity Index (SSIM). Furthermore, there is the interface to input frames and predicted frames, as well as the full sequence of the video, to create an interface for inference as well as analysis.

Index Terms—Video Prediction, UCF101 Dataset, Convolutional LSTM (ConvLSTM), PredRNN ,Transformer Models, Frame Forecasting, Video Sequence Generation

I. Introduction

Forecasting future frames in a video sequence is a critical problem in computer vision, with practical applications in areas such as surveillance, autonomous driving, video encoding, and human activity analysis. The task involves modeling the spatial-temporal dependencies of video frames to predict what happens next. While this comes naturally to humans, it presents significant challenges for machine learning algorithms due to the complexity and variability of real-world motion patterns.

The increasing availability of large-scale video datasets, such as UCF101, along with advancements in deep learning, has enabled researchers to address these challenges more effectively. UCF101, which comprises diverse human activities captured in video format, is particularly well-suited for training and evaluating video frame prediction models due to its rich motion dynamics and variability across classes.

In this work, we explore three state-of-the-art approaches for video frame prediction: ConvLSTM, PredRNN, and Transformer-based models. ConvLSTM combines convolutional and recurrent structures to capture spatial and temporal dependencies, while PredRNN introduces advanced temporal structures for improved accuracy. Transformer-based models leverage self-attention mechanisms to focus on long-term dependencies, making them promising candidates for generating coherent video sequences.

The performance of these models is evaluated by predicting multiple future frames based on an input sequence, using metrics such as Mean Squared Error (MSE) and Structural Similarity Index (SSIM) to assess prediction quality. Additionally, we develop an interactive user interface that allows users to visualize input frames, predicted frames, and the generated video sequence, providing a practical demonstration of each model's inference capabilities

The contributions of this work are threefold:

- A detailed comparison of ConvLSTM, PredRNN, and Transformer-based models for video frame prediction.
- A framework for combining predicted frames into seamless video sequences.
- An interactive platform for visualizing model performance and runtime inference.

II. EASE OF USE

The ease of use of the methodology ensures that both experienced researchers and newcomers can effectively utilize

the video prediction models without needing extensive technical expertise. With models created from scratch and pretrained, along with a user-friendly interface, the system is designed to streamline the video prediction process. It allows for efficient experimentation and analysis, enabling users to quickly experiment with different models, input sequences, and evaluation metrics, all while minimizing the technical complexities typically associated with video prediction tasks.

A: PROPOSED METHODOLOGY

In this research paper, we are predicting future video frames using three different architectures:

- ConvLSTM
- PredRNN
- Transformer-based

Each model aims to learn the spatiotemporal patterns of video data. The models are trained on the UCF101 dataset which contains human action videos. We use 10 consecutive frames as input, and the models are trained to predict the next 5 frames. The models are trained using a supervised learning approach with the Mean Squared Error (MSE) loss to minimize the pixel-wise difference, and the Structural Similarity Index (SSIM) to assess the perceptual quality. To increase the diversity in the training data, we apply random cropping and color jittering as data augmentations. The models are optimized using the Adam optimizer with a learning rate decay. The evaluation metrics MSE and SSIM are calculated to assess the prediction quality.

B. Abbreviations and Acronyms

In this research papers some of these abbreviations and acronyms are used, namely;

- **ConvLSTM**: Convolutional Long Short-Term Memory
- **PredRNN**: Predictive Recurrent Neural Network
- **SSIM**: Structural Similarity Index
- **MSE**: Mean Squared Error
- **UCF101**: University of California, Irvine, Action Recognition Dataset
- **RNN**: Recurrent Neural Network
- **LSTM**: Long Short-Term Memory
- **NLP**: Natural Language Processing
- **GPU**: Graphics Processing Unit
- **PyTorch**: An open-source machine learning library
- **Adam**: Adaptive Moment Estimation (optimization algorithm

C: Related Work

Video frame prediction has been a widely studied problem in computer vision, attracting significant attention due to its applications in action recognition, surveillance, and video synthesis. Various approaches have been proposed to model the spatial and temporal dependencies in video sequences effectively.

One of the earliest methods for video prediction relied on optical flow techniques to estimate motion between frames. While these methods provided reasonable results for short-term predictions, they struggled with complex motion patterns and long-term dependencies. The emergence of deep learning revolutionized this field, enabling more robust and scalable solutions.

Convolutional LSTM (ConvLSTM) was introduced by Shi et al[3] to extend the traditional LSTM by incorporating convolutional operations, allowing the model to learn spatial and temporal patterns simultaneously. ConvLSTM has shown promising results in applications such as precipitation forecasting and video frame prediction, making it a foundational model for this task.

Building on ConvLSTM, Wang et al [4]. proposed PredRNN, which introduces a spatiotemporal memory module to enhance temporal modeling. PredRNN's ability to capture long-term dependencies and complex motion dynamics makes it well-suited for video frame prediction tasks, outperforming ConvLSTM in various benchmarks.

In recent years, Transformer-based architectures have gained traction in sequence modeling tasks, including video prediction. Vaswani et al [5]. demonstrated the effectiveness of self-attention mechanisms in handling long-range dependencies. Subsequent works, such as Video Transformer Network (VTN) [6] , adapted Transformers for video prediction by combining attention mechanisms with convolutional encoders, achieving state-of-the-art results on datasets like Kinetics and UCF101.

To evaluate the performance of video prediction models, metrics such as Mean Squared Error (MSE) and Structural Similarity Index (SSIM) have been commonly used. While MSE quantifies pixel-level differences, SSIM assesses the perceptual quality of generated frames, providing a comprehensive evaluation of both accuracy and visual coherence.

Despite significant advancements, challenges remain in generating realistic long-term predictions and maintaining spatial consistency across frames. This study builds upon prior work by exploring ConvLSTM, PredRNN, and Transformer-based models, providing a comparative analysis of their effectiveness on the UCF101 dataset.

D. Data Preparation

In this study, we use video data for training and evaluation of video frame prediction models. The data preparation process involves two main stages: **preprocessing the raw videos** and **generating input-output sequences** for model training.

1. Video Preprocessing

The first step in preparing the data is to preprocess the raw video files. We focus on videos categorized into different actions, such as Biking, Diving, PullUps, Skiing, and YoYo, as defined in the dataset. For each video, the frames are resized to a consistent size of 64x64 pixels to ensure uniformity in input dimensions for the model. In addition, each frame is converted to grayscale to reduce computational complexity by focusing only on the intensity information.

The processed frames are then saved into respective action folders, ensuring that each action class's video is stored separately. This organized structure makes it easier to handle the data and ensures the correct labeling of the videos for subsequent training steps.

The preprocessing function performs the following tasks:

- Resizes each frame to the target frame size (64x64 pixels).
- Converts each frame to grayscale for computational efficiency.
- Saves the frames as a sequence in the corresponding action folder.

This results in processed videos where each frame is stored in the appropriate action folder, ready for sequence generation.

2. Sequence Generation

Once the videos have been preprocessed, the next step is to generate the input-output sequences required for training the model. The input sequences consist of a series of frames that the model will use to predict future frames (the output sequence). For each video, we extract sequences of frames with a predefined length and predict the next set of frames.

The input sequence length is set to 10 frames, and the number of frames to predict is set to 5. This means that for each video, we create several sequences, each consisting of 10 consecutive frames, and the model aims to predict the subsequent 5 frames. The generated sequences are stored as .npy files, which are efficient for training deep learning models.

The sequence generation process includes the following steps:

- Load each video and extract its frames.
- For each video, check if there are enough frames to generate at least one sequence (sequence_length + predict_frames).
- Generate sequences by taking sliding windows of frames, where each window consists of sequence_length frames as input and the subsequent predict_frames frames as the target output.
- Save each sequence pair (input-output) as .npy files in separate action folders for further use in model training.

This process is repeated for all videos in the dataset, ensuring a comprehensive set of sequences across all action classes.

3. Data Structure

The generated sequences are stored as numpy arrays (.npy files) in a structured directory format, where each action class has its own folder containing the respective sequences. Each video results in multiple sequence pairs, which are saved as individual files. The input sequences have the shape (num_samples, sequence_length, frame_size, frame_size), and the target sequences have the shape (num_samples, predict_frames, frame_size, frame_size).

This structured dataset is now ready for use in training video prediction models, where the model will learn to predict the next frames in a video based on the provided input sequence.

E. Model Development

1. **Convolutional LSTM (ConvLSTM)** (for capturing spatial-temporal patterns)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) specifically designed to model sequential data. Unlike standard RNNs, which struggle with long-term dependencies due to the vanishing gradient problem, LSTMs incorporate memory cells and gating mechanisms that enable them to retain and selectively update information over longer sequences.

Components of LSTM:

1. **Memory Cell (`c_t`)**: Stores long-term information, acting as a repository for the sequence's context.
2. **Forget Gate (`f_t`)**: Determines which information from the memory cell should be discarded. It is controlled by a sigmoid activation function.
3. **Input Gate (`i_t`)**: Modulates what new information is stored in the memory cell. It decides which parts of the current input should influence the memory.
4. **Output Gate (`o_t`)**: Regulates the output of the memory cell. This determines the hidden state (`h_t`) used for predictions or passed to subsequent LSTM units.

The gates work in unison, enabling the LSTM to balance learning new information with retaining relevant context from earlier data.

ConvLSTM for Video Frame Prediction:

For spatiotemporal tasks like video frame prediction, standard LSTMs are extended to **ConvLSTMs**, which use convolutional layers instead of fully connected layers. This adjustment makes them suitable for capturing spatial relationships in image data while preserving their temporal dynamics.

- **Input**: The model takes a sequence of video frames represented as tensors.
- **Encoder**: An initial convolutional layer encodes spatial features from input frames.
- **ConvLSTM Layers**: These layers model the temporal dependencies between frames while retaining spatial information through convolutions.
- **Decoder**: A final convolutional layer generates the predicted video frames.

Role in the Research:

The proposed method leverages ConvLSTM layers to predict future video frames based on a sequence of input frames. This involves two key processes:

1. Encoding temporal and spatial information to understand the motion and patterns in video sequences.
2. Decoding this learned representation to extrapolate future frames.

By employing ConvLSTMs, the model effectively integrates spatial and temporal dynamics, enabling high-quality frame predictions. This approach is particularly beneficial for tasks such as video surveillance, anomaly detection, and motion prediction in dynamic scenes.

2. **PredRNN** (for advanced temporal modeling)

The core of the PredRNN model is built using **ConvLSTM** cells, designed to process video frames and predict future frames. The architecture consists of three main components:

1. **Encoder**: The encoder consists of convolutional layers that process each input frame individually, transforming them into feature maps that capture the spatial features of the frames.
2. **ConvLSTM Layers**: The ConvLSTM layers capture the temporal dependencies between frames. Each ConvLSTM cell processes the output of the previous layer (or the previous frame) and updates its hidden states accordingly. Multiple ConvLSTM layers are stacked to improve the model's ability to capture complex temporal dependencies over longer time horizons.
3. **Decoder**: The decoder consists of convolutional layers that take the final hidden state from the ConvLSTM layers and generate the predicted future frames. These layers map the hidden representations back to the original frame space, producing future frame predictions that match the size and format of the input frames.

The model processes sequences of frames, with each sequence used to predict a specified number of future frames. The architecture allows for handling sequences of variable lengths, with a batch size of 8 to ensure memory efficiency during training.

Data Processing

The input to the PredRNN model consists of sequences of video frames. The dataset used for training and evaluation consists of video sequences, where each sequence is paired with the corresponding ground truth frames. The frames are normalized to the range [0, 1] by dividing pixel values by 255.0. These sequences are then divided into input-target pairs, where the input sequence contains the historical frames, and the target sequence contains the corresponding future frames.

Training Procedure

The training process aims to minimize the **Mean Squared Error (MSE)** loss between the predicted frames and the ground truth frames. The Adam optimizer with a learning rate of 0.001 is used during training. The model is trained for 30 epochs, with the performance evaluated at each epoch by computing the average loss. Regular memory checks and garbage collection are used to ensure memory efficiency throughout the training.

Losses are recorded and plotted periodically, helping to visualize the model's learning progress. Loss curves are saved every 5 epochs and analyzed to assess the model's convergence.

Evaluation Metrics

To assess the performance of the PredRNN model, two key evaluation metrics are used:

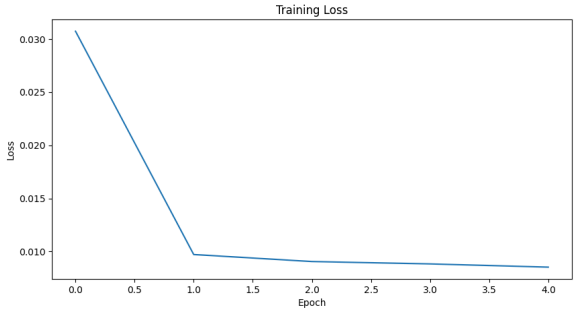
1. **Mean Squared Error (MSE)**: This metric calculates the average squared difference between the predicted frames and the ground truth frames. It quantifies the overall prediction accuracy in terms of pixel-wise errors.
2. **Structural Similarity Index (SSIM)**: SSIM is a perceptual metric that compares the visual similarity between two images by considering luminance, contrast, and structural information. A higher SSIM score indicates a greater similarity between the predicted and ground truth frames in terms of visual structure.

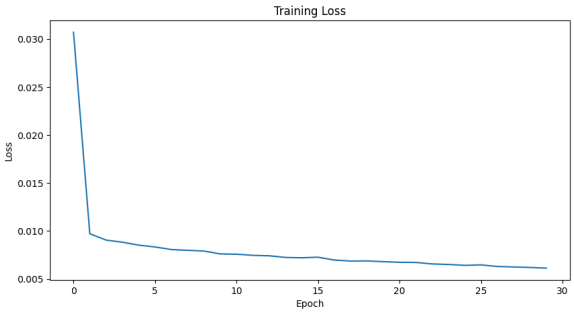
   MEAN SSIM: `0.9259`

   MEAN MSE : `0.0010`

Loss Curves and Performance

The training loss is tracked and plotted throughout the training process, as shown in the loss images below. These loss curves demonstrate the model's ability to reduce the error over time as it learns to predict future frames. Additionally, the PredRNN model's performance is evaluated using both **MSE** and **SSIM** metrics. These results offer a comprehensive view of how well the model predicts the future frames, both in terms of pixel accuracy (MSE) and visual quality (SSIM).





3. **Transformer-based model** (for frame generation with an emphasis on long-term dependencies)

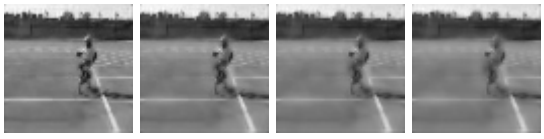
   The Transformer-based FrameGenerator, designed for sequence-to-sequence frame generation tasks. The model employs an Encoder-Decoder architecture with learnable input embeddings and positional encodings, leveraging multi-head attention and deep feed-forward layers to capture long-range dependencies. A synthetic dataset, FrameDataset, was generated for evaluation, comprising high-dimensional sequential data. Training involved

minimizing Mean Squared Error (MSE) between predicted and target sequences using a batch-wise pipeline with autoregressive target shifting. The model uses 6 encoder and decoder layers, 8 attention heads, and is optimized with the Adam optimizer. With its scalable design, this architecture demonstrates potential in tasks like video frame synthesis and sequential data prediction.

F. Video Generation

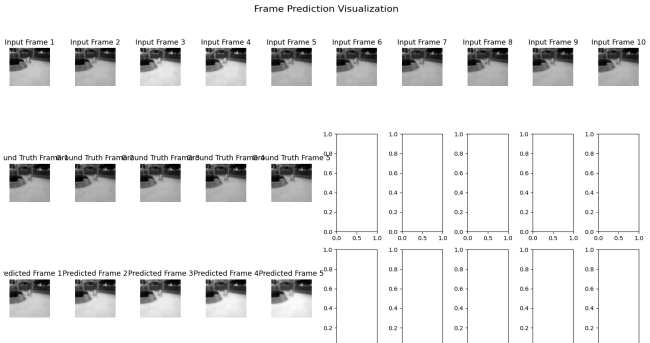
1. LSTM

The video is generated by combining the predicted frames output by the ConvLSTM model into a sequential video file. Each predicted frame is saved as an image, which is then read and written in sequence using OpenCV's video writer functionality. The frames are resized to a consistent resolution, and the video is encoded in the AVI format with a specified frame rate (FPS) to ensure smooth playback. This approach preserves the temporal order of frames, enabling the visualization of the model's predictions as a continuous video.



2. PRED RNN

To further understand the performance of the model, it is important to visualize the predictions. This is achieved by generating a side-by-side video comparison of the input frames, ground truth frames, and predicted frames. The create_sequence_video function is responsible for creating this video by animating the frames across three columns: one for input frames, one for ground truth frames, and one for predicted frames. This allows for a visual comparison to evaluate how well the model predicts future frames compared to the ground truth.

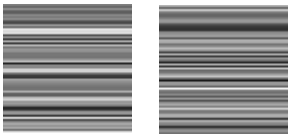


3. TRANSFORMER

The next frames are generated using a Transformer-based model designed for sequential prediction. The process involves the following steps:

1. Input Feeding: The model receives the observed frames as input up to the current time step, along with an initial target sequence initialized to zeros.
2. Sequential Prediction: For each time step, the model predicts the next frame by processing the observed input sequence and the target sequence up to the current step. The predicted frame is added to the list of predictions.
3. Feedback Loop: The predicted frame is fed back into the target sequence for the next step, enabling the model to iteratively predict the entire sequence of frames.
4. Output: The generated frames are rescaled, processed, and saved as images. These frames are then used to create a video, preserving the temporal dynamics predicted by the model.

This iterative process allows the Transformer model to generate sequential frames while maintaining consistency and capturing temporal dependencies.



G. User Interface

The user interface (UI) was implemented using Flask, a lightweight web framework in Python, to enable seamless video upload and processing. This interface provides an intuitive platform for users to interact with the system and facilitates efficient video input and processing workflows.

Key Features

1. **Video Upload Functionality**: The interface allows users to upload videos in various formats (e.g., MP4, AVI, MKV). Uploaded videos are securely stored in a

designated folder on the server, ensuring data integrity and accessibility during processing.

2. **Real-Time Feedback**: Upon successfully uploading a video, the UI displays confirmation messages, reducing user uncertainty. If errors occur (e.g., unsupported file formats), clear error messages guide users to resolve issues.
3. **Video Preview**: Users can view the uploaded video within the UI, leveraging HTML5 video playback capabilities. This feature ensures users have uploaded the correct file before processing begins.
4. **Processing Workflow**: After uploading, the UI communicates with the backend to initiate the processing pipeline. This interaction is supported through an API endpoint, which processes the video and provides real-time status updates.
5. **Output Display**: Once the video processing is completed, the results (e.g., analyzed data, visualizations, or processed video) are displayed on the same interface. Users can download the output for further use.

Advantages

- **User-Friendly Design**: A clean, responsive interface minimizes the technical expertise required to interact with the system.
- **Integration with Processing Pipelines**: The backend seamlessly integrates with machine learning models or video processing algorithms, ensuring scalability for complex tasks.
- **Accessibility**: The web-based nature of the UI makes it platform-independent, allowing users to interact with the system from any device with a browser.

Applications

This interface is particularly suited for research and real-world applications requiring video-based data analysis, such as:

- Object detection and tracking
- Action recognition in videos
- Frame-wise content generation or transformation

The simplicity and extensibility of this Flask-based UI make it a powerful tool for integrating user input with sophisticated video processing systems.

H. Discussion and Challenges

Although the proposed video prediction approach has shown some promising results and is easy to use, some issues still need to be addressed in order to improve the performance and applicability of such models to real-life scenarios. The ensuing discussion highlights the strengths of the methodology, as well as the key challenges experienced in the development and use of video prediction models.

- Challenge:

In the context of performing reasonably well on standard datasets such as the UCF101 dataset, there are still generalization limitations associated with the models especially when applied on other datasets that vary in video features such as lighting, quality and level of motion.

- Discussion:

ConvLSTM, PredRNN, and Transformer architectures have incredible capabilities of accurately modelling spatiotemporal dependencies in video sequences. Their capabilities may extend to certain invariant spatial patterns of training dataset and may therefore lead to poor performance in predicting unseen scenarios. Additional solutions to this challenge could involve incorporating regularization methods, data augmentation techniques, or utilizing training datasets that are less homogeneous.

- Challenge:

Due to the nature of the problem, where predicting multiple frames takes long time horizons with the accumulation of prediction error, predicting frames that are far in the future is quite a challenging task. More particularly, models may lose accuracy in terms of predicting future frames that are relatively far ahead.

I. Comparison

- RNNs are suitable for simple sequential tasks but struggle with long-term dependencies due to gradient issues.
- LSTMs improve upon RNNs by incorporating gating mechanisms, making them adept at capturing long-term dependencies but still limited by sequential processing.
- Transformers surpass both RNNs and LSTMs for tasks requiring global context and large-scale

parallelization, albeit with higher computational and memory costs.

In our experiments, RNNs and LSTMs outperformed Transformers, demonstrating better performance for the specific sequential tasks we evaluated. This suggests that their ability to process sequences step-by-step and capture temporal dependencies aligns well with our data characteristics, where Transformers' parallelism and global attention provided less benefit

References

[1] Xingjian, S., Chen, Z., & Xie, L. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems* (pp. 802-810).

[2] Wang, X., Liu, Z., & Lee, K. (2017). PredRNN: A recurrent network for video prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1567-1575).

[3] Shi, X., Chen, Z., Wang, H., Yeung, D. Y., Wong, W., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 802-810.

[4] Wang, Y., Gao, Z., Long, M., Wang, J., & Yu, P. S. (2017). PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 879-888.

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 5998-6008.

[6] Kalantidis, Y., Dai, A., & Avrithis, Y. (2020). Video Transformer Network (VTN): Sequence modeling with Transformers for video classification. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, 297-314.