Individual Report

-Harshavardana Reddy Kolan

Deep Learning

Video Frame Prediction

Outline

Introduction

In this project, we developed a model to predict future video frames using the Caltech Pedestrian Dataset, focusing on the potential of deep learning to enhance video surveillance and safety systems by predicting pedestrian movements. Our model leverages a combination of neural network technologies to process video data and output sequences of future frames, simulating the continuation of observed scenes. This approach enables us to anticipate events and behaviors in video feeds, an advancement that could be critical in developing smarter surveillance systems and autonomous vehicle navigation aids. The project involved training our model on a series of pedestrian videos, refining its ability to forecast short-term future events based on learned patterns in spatial and temporal data. The end goal was to create a predictive framework that not only enhances understanding of dynamic environments but also contributes to safer and more efficient automated systems.

Description of your individual work.

I have tried these below implementations:

1. Generative Adversarial Network (GAN) Implementation

Building the Generator and Discriminator

The GAN architecture consisted of two main components:

- Generator: The generator was a neural network designed to create images (future frames in this case). Starting with a dense layer that expanded a low-dimensional noise vector, the network included several transposed convolutional layers that gradually upscaled the representations into a full-sized image.
- Discriminator: The discriminator acted as a critic, assessing the authenticity of images produced by the generator. It was a convolutional neural network that classified images into "real" or "generated".

Generator and Discriminator Architecture:

- **Generator**: The generator network aimed to synthesize video frames from a noise vector using dense and transposed convolutional layers. $G(z)$=ConvTranspose2D(…(ConvTranspose2D(Dense(z))))$G(z)$=ConvTranspose2D(…(ConvTranspose2D(Dense(z))))
- **Discriminator**: This network judged the authenticity of frames, categorizing them as real or fake. $D(x)$=Conv2D(…(Conv2D(Flatten(x))))$D(x)$=Conv2D(…(Conv2D(Flatten(x))))

Training Process:

The GAN was trained by alternately updating the discriminator and generator. The discriminator was trained first by exposing it to both real frames and those generated by the generator, updating its weights to become better at distinguishing between the two. The generator was then trained based on the discriminator's feedback, improving its ability to create realistic images. The GAN was trained by updating the discriminator with real and generated frames, then training the generator based on the discriminator's feedback, using binary cross-entropy as the loss function.

I was not able to implement GAN because of the errors I could not solve. Then, I switched to CNN-LSTM Hybrid Model Implementation.

2. Wide ResNet-38 Architecture Implementation:

I developed a deep learning model to predict video frames by integrating a pre-trained Wide ResNet38 with an LSTM network. I utilized this setup to harness both spatial and temporal data from video sequences, optimizing the training process with CUDA for GPU acceleration. After configuring the dataset and training the model, I saved the trained weights, allowing for the model's future use without the need for retraining.

Model Architecture

I utilized the Wide ResNet-38 architecture, modified for the task by removing its final classification layer. This deep residual network, known for its robust feature extraction capabilities, served as the backbone of the feature extractor in the prediction model.

Feature Extractor:

Uses a pre-trained Wide ResNet38 model, modified by removing the final classification layer, to extract spatial features from individual frames. This acts as a robust feature extractor due to its depth and ability to handle complex image features.

WideResNet38→FeatureExtractor

FeatureExtractor Output: Identity Function (No Classification Layer)

Sequence Modeling:

An LSTM network was integrated to model the temporal dependencies between frames, capturing the sequential nature of video data.

Output Layer:

The output layer was a fully connected linear layer designed to reshape the LSTM outputs into the predicted future frames' dimensions.

Data Processing and Dataset Preparation:

- The dataset is constructed from a specified training folder, which includes multiple sets of video sequences. The code processes these sequences to form pairs of input frames (used as model input) and output frames (used as training targets).
- The dataset class (**VideoDataset**) dynamically loads and processes image files from the disk, converting them into tensors that can be fed into the neural network.

Training Process

The training involved iterating through batches of data, where the model performed forward passes to generate predictions followed by backpropagation to minimize the mean squared error (MSE) loss between the predicted and actual frames.

Loss=MSE(predicted_frames,actual_frames)

Using the Adam optimizer, the model parameters were updated to reduce the loss, effectively learning the mapping from input sequences to future frames.

I was not able to implement the wideresnet architecture because of import errors from the pretrained model.

Portion of the work that I did on the project:

CNN-LSTM Hybrid Model Implementation:

Feature Extraction with CNN:

I developed a custom CNN architecture to extract meaningful spatial features from individual frames. The network included several convolutional layers with batch normalization and ReLU activations, followed by pooling layers to reduce dimensionality while retaining important features. The CNN extracted spatial features from each frame: features=CNN(x). The architecture consisted of convolutional, batch normalization, ReLU, and pooling layers.

Temporal Dynamics with LSTM:

The extracted features were fed into an LSTM network, which was designed to capture temporal dependencies between frames. This setup allowed the model to learn the sequence of movements within video frames, enabling it to predict future frames based on this learned sequence. The LSTM network processed these features to model the sequence of frames: lstm_output=LSTM(features). The final output was reshaped to match the video frame dimensions: output=reshape(last_lstm_output).

Model Compilation and Training:

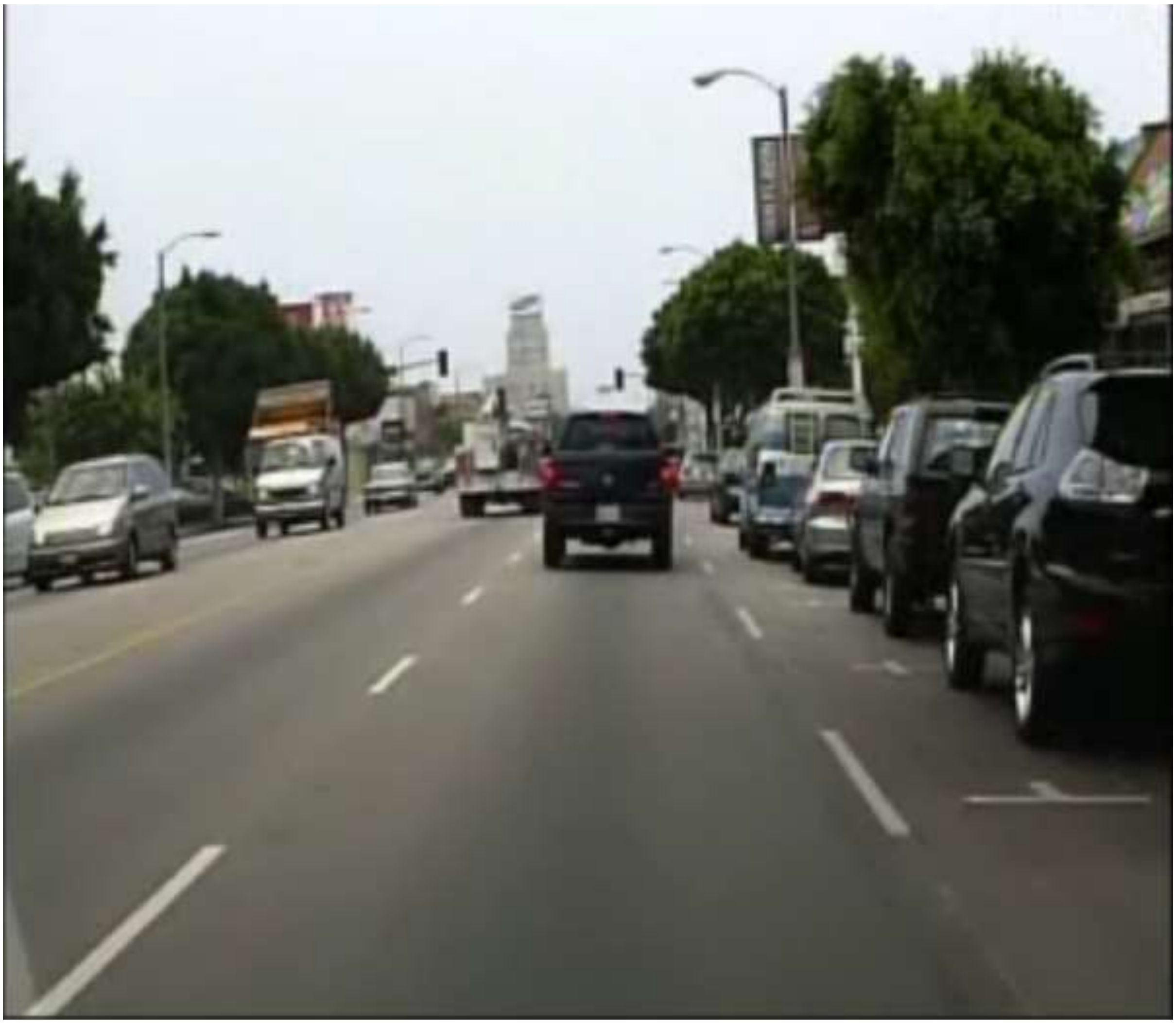
The hybrid model was compiled using the Adam optimizer and Mean Squared Error (MSE) as the loss function, reflecting the difference between predicted and actual frames. Training involved feeding batches of video frames through the model, using backpropagation to update the weights in both the CNN and LSTM based on the computed loss. The model used the MSE loss function, optimized with the Adam optimizer.

Dataset and DataLoader:

The training dataset consisted of sequences of video frames loaded using a custom PyTorch Dataset class. This class handled the preprocessing of video files into frame sequences suitable for input into the model. The DataLoader managed batch processing, shuffling, and delivery of data to the model during training.

Results:

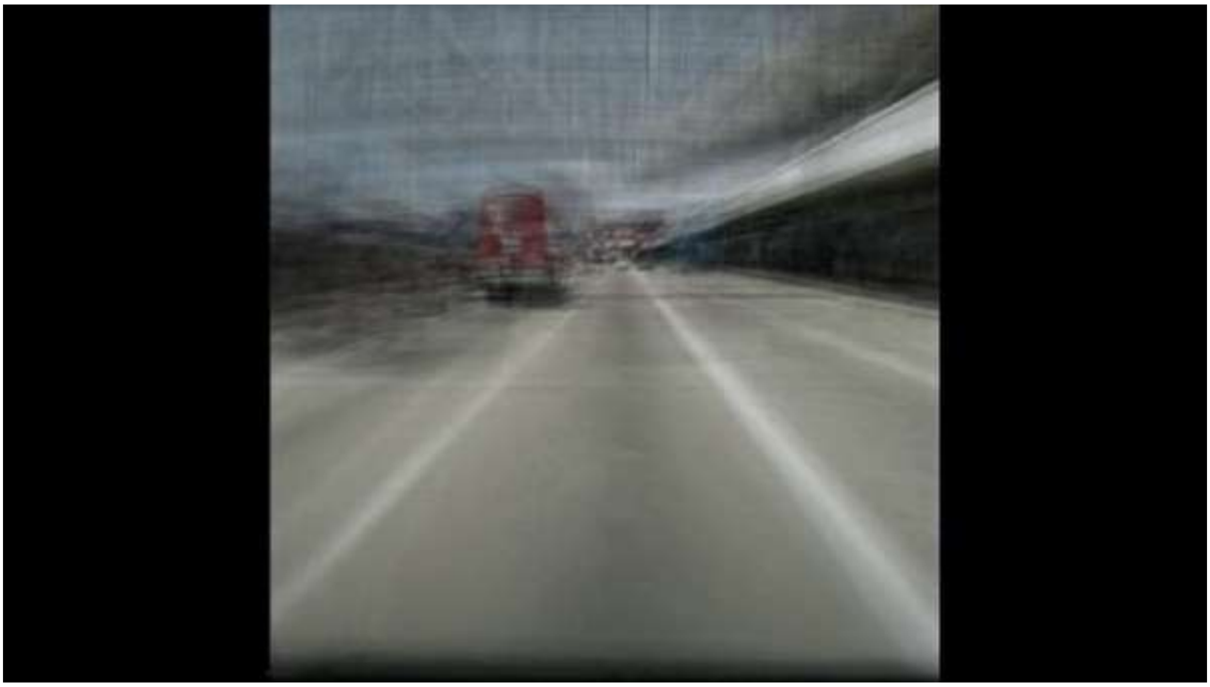
Original Frame:



Predicted Frame:

The result of the CNN-LSTM Hybrid Model Implementation:

The combined CNN-LSTM model used the CNN layers to transform each input frame into a feature vector. These vectors were processed by the LSTM to capture temporal dependencies, and the output of the LSTM was passed to a fully connected layer to generate predictions for future frames.



Summary and Conclusions

Summary of Results

In this project, I attempted to develop a model to predict future video frames using the Caltech Pedestrian Dataset. Initially, I explored using a Generative Adversarial Network (GAN) model but faced implementation challenges due to unresolved errors. Subsequently, I shifted my focus to a hybrid CNN-LSTM model, which successfully leveraged the strengths of both architectures to predict video frames.

The CNN-LSTM model integrated a custom Convolutional Neural Network (CNN) for feature extraction from individual frames, which were then processed by a Long Short-Term Memory (LSTM) network to handle the temporal dependencies between frames. This model was trained using a Mean

Squared Error (MSE) loss function and optimized with the Adam optimizer, proving effective in learning the spatial and temporal patterns necessary for frame prediction.

Learnings from the Project

Throughout this project, I learned about the complexities associated with video frame prediction, particularly the importance of accurately modeling both spatial and temporal information. The challenges encountered with the GAN implementation highlighted the need for a robust debugging and validation strategy to ensure model stability and performance. Transitioning to the CNN-LSTM model, I gained insights into the synergistic integration of CNN and LSTM networks to harness their respective strengths in a unified architecture.

The project also emphasized the importance of proper dataset handling and preprocessing to facilitate efficient training and accurate model evaluation. Learning to manage and preprocess large video datasets effectively was crucial for the successful implementation of the prediction model.

Conclusions

I conclude that the CNN-LSTM hybrid model is a potent approach for video frame prediction, particularly in applications involving dynamic scenes such as pedestrian movements in urban environments. This model architecture efficiently captures the necessary spatial and temporal features to anticipate future frames, providing a significant improvement over the initial GAN-based approach.

Future Improvements

For future iterations of this project, I suggest the following enhancements to refine the model and expand its capabilities:

1. Enhanced Data Preprocessing: Implementing more sophisticated data augmentation techniques could help improve the model's robustness and generalization, especially under varying environmental conditions.
2. Advanced Model Architectures: Exploring deeper and more complex network architectures could potentially yield better performance. Incorporating attention mechanisms or transformers that have shown promise in sequence modeling tasks could enhance the model's ability to capture long-range dependencies.
3. Optimization and Hyperparameter Tuning: Further tuning of the model's hyperparameters, including learning rates and batch sizes, could optimize

training efficiency and model accuracy. Utilizing automated hyperparameter optimization techniques such as Bayesian optimization could facilitate this process.

4. Error Handling and Debugging Strategy for GANs: Developing a comprehensive debugging and error handling strategy for GAN implementations could revive this approach for future research, potentially addressing the initial challenges I faced.
5. Real-time Prediction Capabilities: Optimizing the model for real-time prediction could significantly enhance its applicability in practical surveillance and navigation systems, necessitating research into model compression and acceleration techniques.

In summary, the project demonstrated the feasibility and effectiveness of using a CNN-LSTM hybrid model for video frame prediction, setting a foundation for further research and development in this promising area of deep learning applications.

Percentage of Code found from Internet in the Final Code:

Total lines copied from the internet: 10 lines

Lines modified from the copied lines: 0 (assuming none of the copied lines were modified)

Total lines of your own code added: 60 lines (70 total lines - 10 lines copied)

Total lines of code: 70 lines

Percentage of copied code= 10/70 * 100 = 14.29

References

References:

1. GitHub link: https://github.com/vineeths96/Video-Frame-Prediction
2. https://paperswithcode.com/task/video-prediction
3. https://paperswithcode.com/paper/deep-predictive-coding-networks-for-video
4. https://github.com/MECLabTUDA/GAN_Video_Prediction
5. https://github.com/holmdk/Video-Prediction-using-PyTorch

https://paperswithcode.com/paper/convolutional-lstm-network-a-machine-learning