Project Report

Deep Learning

Group 6

Video Frame Prediction

Outline

Introduction

Dataset Description

Description of Algorithm

Experimental Setup

Results

Conclusion

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 sequences, 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.

Dataset Description

Source: https://data.caltech.edu/records/f6rph-90m20/files/data_and_labels.zip?download=1

For the project we have used the Caltech pedestrian dataset from the Caltech library website. The dataset contains the sequence files(.seq). The dataset is around 11.8Gb which consists of 'Train' and 'Test' folders. The dataset contains captures from a moving vehicle, the dataset features video footage characterized by low-resolution imagery and frequent occlusions of pedestrians, presenting a realistic and challenging environment for predictive modeling. The dataset's complexity is further highlighted by its critical assessment of commonly used evaluation metrics. This dataset provided a robust framework for testing our video frame prediction model, particularly in simulating and understanding pedestrian dynamics in urban settings.

Description of Algorithm

Our model architecture for predicting video frames uses a combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks.

1. CustomCNN (Feature Extractor):

CustomCNN serves as the feature extractor within our model architecture. Its configuration is as follows:

- **Convolutional Layers:** Each convolutional layer applies filters to the input data to create feature maps that capture spatial hierarchies for different aspects of the input images. The first layer uses 64 filters of size 7x7, followed by layers with 128 and 256 filters of size 3x3, respectively. These layers are equipped with a stride of 2 and padding to maintain the size consistency across layers.
- **Batch Normalization:** This is used after each convolutional operation to stabilize learning and normalize the input layer by re-centering and re-scaling.

- **ReLU Activation:** The Rectified Linear Unit (ReLU) activation function is used to introduce non-linear properties into the network, helping it learn more complex patterns.
- **Pooling Layers:** Max pooling is applied following the first convolutional layer to reduce the spatial size of the representation, thereby reducing the number of parameters and computation in the network. Adaptive average pooling is used at the end to reduce each feature map to a single value, flattening the output in preparation for the LSTM layer.

2. VideoPredictionModel (Temporal Processing):

- **LSTM Network:**

  The LSTM layers are designed to handle sequences of data (in this case, the feature vectors from consecutive frames). It processes the temporal information and is particularly effective in avoiding the vanishing gradient problem common in traditional recurrent neural networks, thus better capturing long-term dependencies.

- **Output Layer:**

  A fully connected linear layer converts the LSTM outputs to the required size to reconstruct the predicted frames. This output mimics the number of pixels multiplied by the number of channels for each frame in the sequence.

Training of Algorithm:

1. Data Preprocessing: Frames from the Caltech Pedestrian Dataset are preprocessed through resizing to 320x320 pixels, conversion from BGR to RGB format (as OpenCV reads images in BGR format), and normalization to scale the pixel values between 0 and 1.

2. Model Compilation: The model is compiled using the Adam optimizer, which adjusts the learning rate dynamically. It uses Mean Squared Error (MSE) as the loss function, which is appropriate for regression tasks where the goal is to minimize the error between predicted and actual frames.

3. Training Loop: Training involves feeding batches of frame sequences into the model, where the network predicts the next frames based on previous frames. The

model's weights are updated to minimize the loss between the predicted and actual frames.

4. Evaluation and Adjustment: Throughout the training process, the model's performance is evaluated, and adjustments are made to the hyperparameters, or architecture as needed based on the observed performance metrics.

Experimental Setup

Data Utilization

The dataset used for this project is pre-divided into training and testing subsets and consists of JPEG images depicting urban scenes with pedestrians. These conditions are ideal for training a model to predict future video frames. The training subset enables the network to learn the task of frame prediction by adjusting its parameters according to the past data, while the testing subset evaluates the model's ability to generalize this learning to new, unseen data.

Network Implementation in PyTorch

The network is implemented using PyTorch, a flexible and powerful framework suitable for handling sequences and dynamic computation graphs needed for video data processing.

Network Architecture:

- **CustomCNN:** Acts as a feature extractor, using convolutional layers to capture spatial hierarchies.
- **LSTM Network:** Captures temporal dependencies between frames, crucial for accurate future frame prediction.
- **Output Layer:** Transforms LSTM outputs to the necessary dimensionality to reconstruct the video frames.

Training Procedure

- **Minibatches Usage:** The model training employs minibatches, which are crucial for managing memory usage efficiently and speeding up the training process. By averaging the gradient updates over minibatches, the learning updates are stabilized.

- **Minibatch Size:** The chosen batch size=1 , primarily due to the high resolution of the frames. This setup implies the use of stochastic gradient descent, which can help the model escape local minima during training.
- **Training Parameters:**
    - **Learning Rate and Optimizer:** The learning rate is set adaptively by the Adam optimizer, known for adjusting the learning rate based on the training dynamics. This helps in mitigating the need for manual adjustments.
    - **Epochs:** The training is set to run for five epochs, balancing between computational resources and the need for sufficient training to achieve effective learning outcomes.

Performance Evaluation

To assess the model's performance, we use the Mean Squared Error (MSE) for quantifying prediction accuracy and Peak Signal-to-Noise Ratio (PSNR) for evaluating the quality of reconstructed frames:

- **MSE:** Lower MSE values indicate more accurate predictions.
- **PSNR:** Higher PSNR scores suggest better quality reconstructions, comparing favorably against standard compression artifacts.

Overfitting Prevention

Our model does not explicitly implement mechanisms like early stopping or dropout in the provided code, focusing instead on robust training through the careful setup of epochs and learning rates within the scope of what is programmatically defined. However, in practice, techniques such as early stopping could be considered to halt training when validation performance degrades, thus preventing overfitting.

This structured approach ensures that our model not only performs well on the training data but is also robust enough to handle new, unseen scenarios, aligning with real-world application needs for video frame prediction.

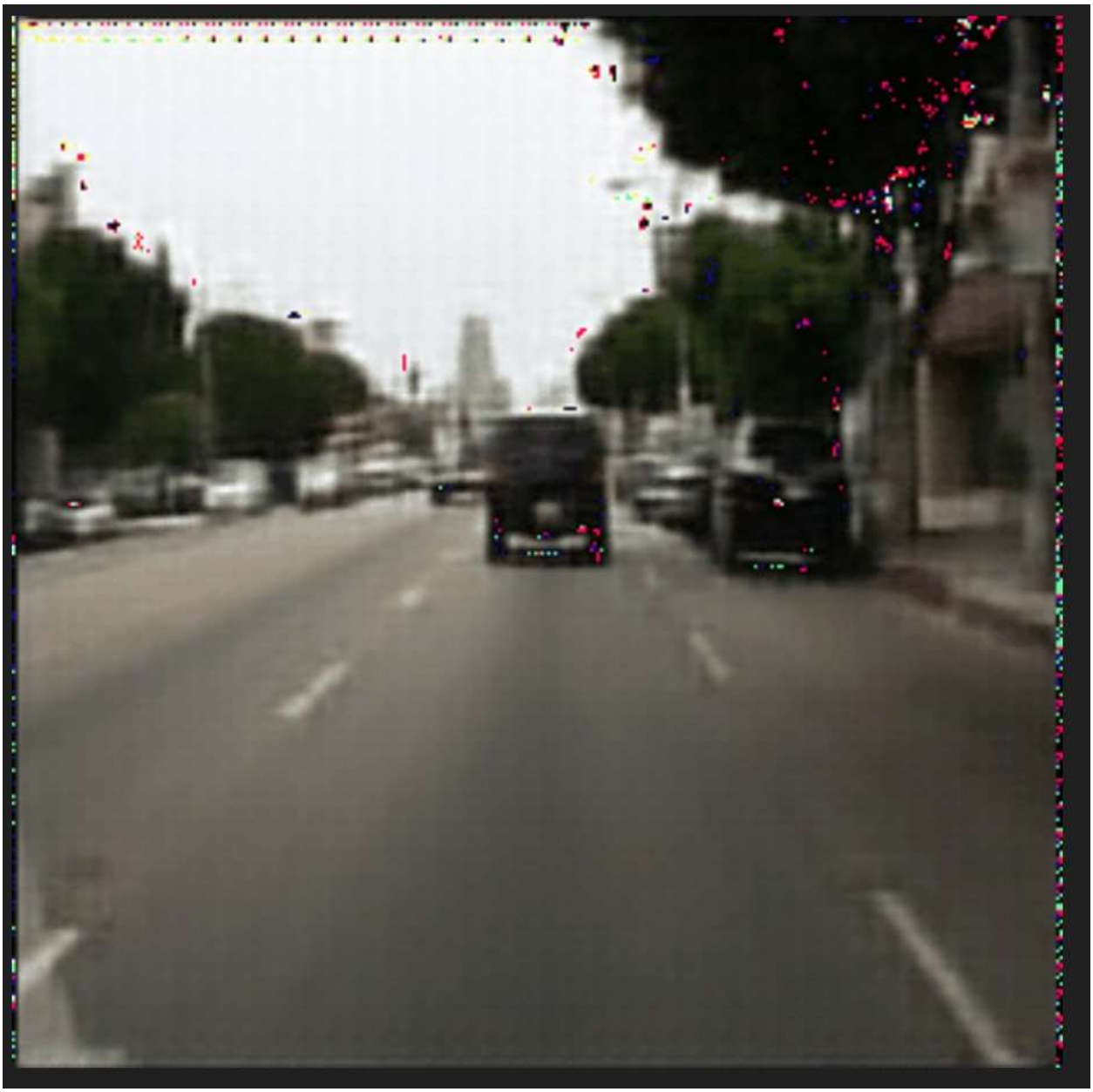
Result

Original Frame:

Predicted Frame:

Initially we used only Convolution Network, which has an encoder and decoder in it. The encoder converts the sequence of images to a lower dimension and gives it as output to the decoder. Then, the decoder converts the output of encoder to a higher dimension sequence of frame. We realized that this approach was wrong. Then, we decided to transition to LSTM, integrating temporal dynamics.

Learning from Limitations: The shortcomings of the CNN-only model highlighted the need for capturing temporal relationships in the video data, a task for which LSTMs are particularly well-suited. LSTMs can process data sequences and maintain hidden states, allowing them to store information about previous frames that is crucial for predicting future frames. So, we decided to implement LSTM in our model.



Improved Model Predicted Frame:

Model Enhancement with LSTM: We introduced an LSTM layer to work in conjunction with our existing CNN architecture. The CNN continued to serve as a feature extractor, encoding spatial information of each frame into a condensed feature vector. These vectors were then fed sequentially into an LSTM network, allowing it to learn and remember temporal dynamics across frames.

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.

Improvements:

Quantitative Improvements:

The introduction of LSTM led to a substantial improvement in our model's performance. While testing the model, we got an MSE around 0.05. The MSE decreased significantly, indicating more accurate frame predictions. Similarly, the PSNR (Peak Signal-to-Noise Ratio) score improved, which meant that the predicted frames not only matched the ground truth more closely in terms of pixel values but also exhibited enhanced image quality and fidelity from a structural and perceptual viewpoint. While testing the model, the value of PSNR was around 14. This enhancement in PSNR underscores the model's ability to reconstruct high-quality frames that are visually more similar to the actual video sequences.

Visual Assessment:

Visually, the frames predicted by the CNN-LSTM model were sharper and more coherent over time compared to those generated by the CNN-only model. The

LSTM-equipped model managed to preserve object continuity and dynamics much better, demonstrating its ability to understand and predict the movement of pedestrians within the frames.

Summary and Conclusions

Summary of Results

In our project, we developed a video frame prediction model using the Caltech Pedestrian Dataset. Our approach utilized a hybrid neural network architecture combining Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. The CNN component acted as a feature extractor, translating raw video frames into a condensed spatial feature set, while the LSTM module processed these features to capture temporal relationships and predict future frames.

The integration of the LSTM was a pivotal enhancement over our initial CNN-only model, which struggled with temporal dynamics inherent in video sequences. By adopting LSTM, we significantly improved the predictive accuracy and quality of the reconstructed frames. Our final model achieved a Mean Squared Error (MSE) of approximately 0.05 and a Peak Signal-to-Noise Ratio (PSNR) of around 14, indicating both high accuracy in frame prediction and substantial quality in the reconstructed images.

Learnings from the Project

Through this project, we learned the critical importance of modelling both spatial and temporal dependencies in video data for frame prediction tasks. The CNN's spatial processing capabilities were crucial but insufficient alone, as video frame prediction also requires understanding the sequence and continuity of movements, which the LSTM effectively provided.

Moreover, the project underscored the challenges of working with video data, particularly issues related to low-resolution imagery and occlusions, common in real-world datasets like the Caltech Pedestrian Dataset. These challenges necessitated careful consideration of model architecture and training procedures to ensure robustness and reliability.

Conclusion

We conclude that the hybrid CNN-LSTM architecture is highly effective for predicting video frames in scenarios involving dynamic environments such as pedestrian movements in urban settings. The model not only improved in quantitative metrics like MSE and PSNR but also demonstrated qualitative enhancements in the visual coherence and continuity of predicted frames.

Future Improvements

For future iterations of the project, several enhancements could be considered:

1. **Data Augmentation**: Implementing data augmentation techniques such as random cropping, rotations, and horizontal flipping could help improve the robustness of the model against varied and unforeseen conditions.
2. **Model Complexity and Regularization**: Introducing regularization techniques such as dropout or L2 regularization within the LSTM could help mitigate overfitting, especially with more extensive datasets or longer training durations.
3. **Advanced Architectures**: Exploring more sophisticated neural network architectures like GANs (Generative Adversarial Networks) or deeper LSTM networks might yield better performance, particularly in handling more complex scenarios with multiple interacting objects.
4. **Real-Time Processing**: Optimizing the model for real-time prediction could expand its applicability in practical surveillance and safety systems, requiring efficient processing and low latency.
5. **Extended Training and Tuning**: Increasing the number of training epochs, experimenting with different learning rates, and employing techniques like learning rate schedules or early stopping could further refine model performance.

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