Individual Report

Deep Learning

Video Frame Prediction

Introduction

In this project, we developed a deep learning model aimed at predicting future frames in video sequences, a critical capability for applications in surveillance, autonomous driving, and video compression. Our work primarily utilizes the Caltech Pedestrian Dataset, which offers a diverse array of urban driving scenarios captured through vehicle-mounted cameras, making it an ideal resource for training our predictive models. Originally, we planned to also incorporate the KITTI dataset to compare model performance across different traffic and pedestrian environments. However, technical difficulties prevented us from processing and utilizing the KITTI data effectively within our project's scope.

GitHub Repo: https://github.com/sriramsathwik59/Deeplearning-Group6-

Dataset

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

The dataset is taken from the Caltech data library. The dataset contains moving objects such as cars, trucks, and some. The dataset is in the form of the .seq file. The dataset is around 11GB.

Description of Individual Work

As there are many networks such as ConvLstm, CNN&LSTM, GAN, PredNet type of architecture to build the video frame prediction. I have tried using GAN network.

Background information of Algorithm:

GAN Architecture

For the video frame prediction, I employed a Generative Adversarial Network (GAN) framework. GANs are composed of two main components: the Generator and the Discriminator. The Generator aims to create video frames that are indistinguishable from real sequences, while the Discriminator evaluates their authenticity.

Generator

The Generator is designed to forecast future frames given past frames. It learns to synthesize potentially infinite future frames from a given sequence.

Discriminator

The Discriminator evaluates the authenticity of both real sequences and the synthetic frames produced by the Generator. Its goal is to differentiate between the 'real' frames from the dataset and 'fake' frames generated by generator.

Portion of work

As there are many datasets, first I wanted to implement moving MNIST. For moving MNIST I have used a code to generate the dataset I got the code on Internet there are some changes needs to do versions as such so and after getting the data I have implemented the convlstm on the dataset it worked. So based on that I wanted to implement it on the KITTI dataset. Making an account on the website is very hard on it due to some errors. After downloading the dataset from the KITTI. I tried to use the KITTI.

Preprocessing:

It was very hard to preprocess the dataset at first as the folder contains the images, related time stamps which is a text document which has the time stamp for each image and there are points in the documents where it tells us the pedestrians and the car is. I tried to map the points from the text document from the time stamp and map it with each of the pictures and put that into to a sequence before training the data. I have used the internet to give the code. But there are errors while doing it. So, we shifted to Caltech. Caltech dataset is in seq file. The dataset has 2 folders which are train and test. Preprocessed the dataset with python code which can be found in our GitHub repo ‘sequence.py’ that takes the seq files and converts them into sequence frame of images to their specific folders.

For the model I have used GAN to implement the frames. The models functions follows below following: Generator: The Generator's role is to create the next frame in a video sequence from a given frame. It consists of convolutional layers that upsample the input frame, producing a frame as output. This model uses a simple architecture with a Conv2d layer followed by a ReLU activation and a ConvTranspose2d layer that outputs the transformed frame with a Tanh activation to scale the pixel values between -1 and 1.

Discriminator: The Discriminator evaluates whether a given frame is real (from the dataset) or fake (generated by the Generator). It is constructed using Conv2d layers followed by a LeakyReLU activation to prevent vanishing gradient issues, culminating in a AdaptiveAvgPool2d to reduce each feature map to a single scalar, and a Sigmoid activation to output a probability.

Training process:

Loss Functions**:** For the loss function we used BCELoss in both training the generator and discriminator.

Optimization: Used Adam optimizer for both models.

Loop: The training involves alternating updates between the discriminator and the generator. The discriminator learns to better classify frames as real or fake, while the generator improves its ability to create realistic fake frames that can fool the discriminator.

Tweaking:

Learning Rate: Experimented with different type of learning rate. When the learning rate was 0.0001(for both generator and discriminator) the model is killed before even gets started. To start the code I tried to

increase learning rate to 0.01 and for the Num workers which is basically the number of CPUs assigned is 8 which basically giving the all the CPUs for the work.

Loss Function: For the loss function I started with BCE loss function, but after that I tried with Wasserstein Loss which I got from the internet. I tried this loss function with different learning rate but the model was killed instantly.

The model started working when the learning rate= 0.02 and batch size =10. But after some time, it got killed as well. If we increase the GPU, we can train a greater number of batches.

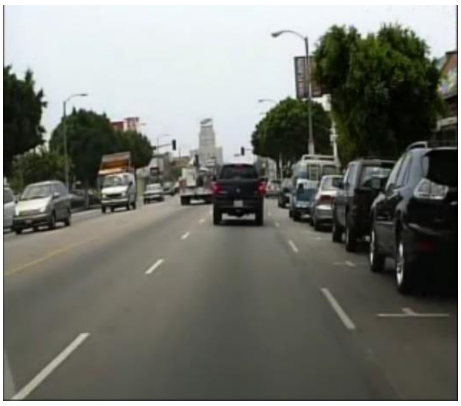
Shifted to CNN& LSTM model:

So, we have shifted to the model to Custom model of CNN and LSTM. Which gave us a pretty good result. I have worked on preprocessing the dataset and test file. The preprocessing is the same way I have implemented for the GAN. The testing files are written in such a way we can take as many as folder for the testing. We have testing folders from set006 to set010. We can choose only one set as well if we do not have enough GPU.

Results

For CNN and LSTM

Predicted Frame:



Original Frame:

Conclusion

In this project, I aimed to predict future video frames using GAN. The GAN, comprising a Generator to create video frames and a Discriminator to assess their authenticity, Initially it started showing some progress but it got killed. I transitioned to a hybrid CNN and LSTM model, which provided better performance and stability, handling temporal data more effectively.

Lesson Learned:

Dataset Complexity and Preprocessing: Handling real-world datasets like Caltech and KITTI revealed the complexities involved in preprocessing.

Model Sensitivity: The project underscored the sensitivity of GANs to hyperparameters like the learning rate and batch size.

Future Improvements:

Expanded Dataset Usage: Although technical issues limited our use of the Caltech dataset, future projects should aim to integrate multiple datasets to ensure that models are well-tested across different scenarios.

Implementing ConvLSTM: Implementing a ConvLSTM model could greatly enhance the handling of spatial-temporal dynamics, which is crucial for accurately predicting video frames in complex scenarios.

Percentage of code:

I have used 300 lines of code from internet, modified 80 lines of code, added 40 lines of code:

(300-80/300+40) *(100)=64%

References

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