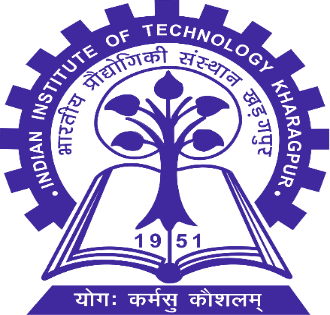
****

**INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR**

**AI60201: Generative and Graphical Models for ML**

**Topic: Unconditional Video Generation using**

**Deep Generative models**

**Term Project Report by**

**Sumanth Yegurla (18AE30028)**

1. **Introduction:**

Deep Generative models have gained significant attention recently for their usability in computer vision tasks. These models learn deep feature representations in an unsupervised manner and can also generate novel data. As steady progress toward better image generation is made, studying the video generation problem is also essential. However, the extension from generating images to generating videos is very challenging, although the generated data has just one more dimension – the time dimension. Video generation is much harder for the following reasons:

* A video is a spatio-temporal recording of visual information of objects performing various actions, a generative model needs to learn the plausible physical motion models of objects in addition to learning their appearance models.
* The time dimension brings in a huge number of variations. Consider the amount of speed variations that a person can have when performing a squat movement. Each speed pattern results in a different video, although the appearances of the human in the videos are the same.
* Human beings have evolved to be sensitive to motion, motion artifacts are particularly perceptible.

We can employ different paradigms for video generation task which differ mainly based on the presence or absence of input conditions. The authors of the paper ‘*Video Generative Adversarial Networks: A Review*’ provides the following classification of Deep Generative models for Videos:

* Unconditional Video Generation
* Conditional Video Generation
  + Image to Video
  + Semantic map to Video
  + Text to Video etc.,

1. **Problem Statement:**

     Among different paradigms mentioned above for Generative modelling of videos, we will be mainly focusing on **Unconditional Video Generation,** which represents a class of models that learn in the absence of input conditions. Producing videos without prior information is more challenging as the model needs to capture the data distribution without help from the input signal which can help to narrow the target space. For this project, we will be implementing the **VideoGAN** architecture, a convolutional GAN model, as presented in the paper titled, ‘*Generating Videos with Scene Dynamics*’ [2] by training it on **UCF11** dataset.

* 1. **Dataset:**

UCF11 dataset contains 1600 videos based on different activities like basketball shooting, biking/cycling, diving, golf swinging, horseback riding, soccer juggling, swinging, tennis swinging, trampoline jumping, volleyball spiking, and walking with a dog. This data set is very challenging due to large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc. For the purpose of our experiments, we mainly used videos with more than 32 frames (we only removed three videos which have less the 32 frames, making the dataset size to 1597).

1. **Model Architecture**:

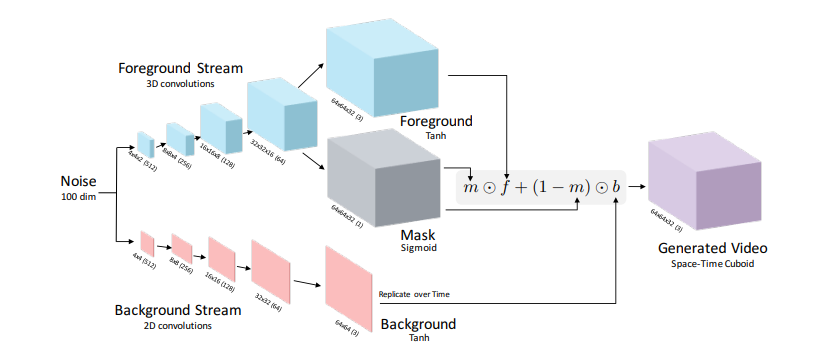
To capture some of the temporal knowledge contained in large amounts of unlabelled video, the authors present an approach that learns to generate tiny videos with realistic dynamics and motions. The model pipeline is based on standard Generative Adversarial Networks. VideoGAN is a convolutional GAN model which contains a two-stream generative model that explicitly models the foreground separately from the background, which allows us to enforce that the background is stationary, helping the network to learn which objects move and which do not. As with any other GAN model, the VideoGAN architecture contains a Generator and a Discriminator Network, both of which are Convolutional Neural Networks. Here, time is considered as a third dimension. The generator network G tries to produce a video, and a discriminator network D tries to distinguish between “real“ videos and “fake” generated videos. These are trained using an alternative mini-max game where they optimize the below objective function contrastively. Here as we do not know the true data distribution p(x), we can estimate the expectation by drawing from the dataset.

1. **Generator Network:**

The generator network consists of two convolutional networks: the first a 3D spatio-temporal convolutional network that captures moving objects in the foreground (blue stream as shown in figure), while the second is a 2D spatial convolutional model for the static background (red stream in the figure). The idea is such that, 0 ≥ m(z) ≥ 1 can be viewed as a spatio-temporal mask that selects either the foreground f(z) model or the background model b(z) for each pixel location and timestep.



To enforce a background model in the generations, b(z) produces a spatial image that is replicated over time, while f(z) produces a spatio-temporal cuboid masked by m(z). By summing the foreground model with the background model, we can obtain the final generation



1. **Discriminator Network:**

The discriminator must be able to classify realistic scenes from synthetically generated ones, and recognize realistic motion between frames. The authors use a five-layer spatio-temporal convolutional network. The design of the architecture is made to be reverse of the foreground stream in the generator, replacing fractionally strided convolutions with strided convolutions (to down-sample instead of up-sample), and replacing the last layer to output a binary classification.

1. **Experimental Setup:**
2. **Dataset pre-processing**:

First, we converted limited 32 frames only, and each frame is of dimension 64 x 64 x 3. Out of 1600 videos, 3 videos have less than 32 frames, so we discarded them. We make use scikit-video library for video processing.

1. **Model Implementation**:

We used PyTorch for the implementation. For the generator network, we considered Conv3DTranspose layers for foreground stream and Conv2DTranspose layers for the background stream. We also used some batch norm and Leaky ReLU activation layers. For the discriminator network, we considered normal Conv3D layers as the feature map need to be down-sampled for the 0/1 classification.

1. **Hyper Parameters used:**

Number of Epochs = 15;

Input Video Dim = [3,32,64,64];

Noise Vector Dim = 100;

Batchsize = 32;

Learning Rate = 0.0002;

1. **Training Loop**:

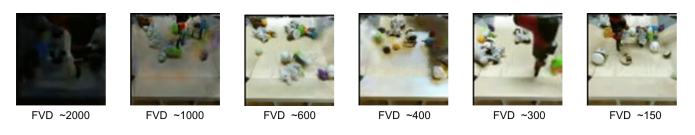
In the training loop, for every iteration we do the following,

* Update Discriminator network: maximize log(D(x)) + log(1 - D(G(z)))
  + The first term is obtained by feeding a batch of real videos to Discriminator.
  + The second term is obtained by first sampling noise for generating batch size number of videos (we took dim of noise as 100) using generator network and feeding them to discriminator and taking log of 1 – (the discriminator output).
* Update Generator network: maximize log(D(G(z))) (same as minimizing the discriminator objective)
  + The term is obtained by taking the log of output of Discriminator when fake videos are fed into it.

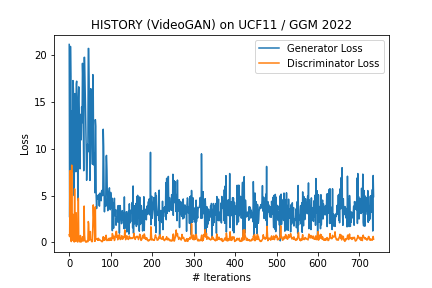
1. **Plots and metrics used**:

We plot the loss for generator and discriminator networks. To assess the quality of generated videos, we used a metric called **Fréchet Video Distance (FVD)** to evaluate the visual quality of generated videos, which is a temporal counterpart of Fréchet Inception Distance (FID). Specifically, FVD computes Fréchet Distance between real-world video distribution PR and fake video distribution PG under the condition that PR and PG are multivariate Gaussian. FVD is defined as :

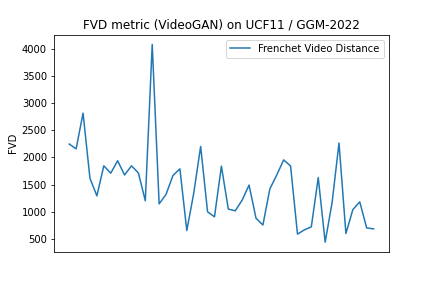
Usually values of FVD range between 0 – 2000. As we can see from the below image taken from original FVD paper, the lower the metric the better is the quality of generated videos.



1. **Results:**
2. We plotted the loss for generator and discriminatorby storing them at each iteration. We ran the model for 15 epochs. We can see that the loss values for both the networks seems to decrease over the iterations. Also, we can see the spikes went down with iterations. As for any other GAN model, this model is hard to converge due to competitive optimization alternatively. The loss plot is as shown below,

****

1. We also calculated the Fréchet Video Distance (FVD) between fake and real video batches after every 20 iterations. The plot for the same can be seen in the figure below:

****

We can see that FVD goes down with iterations. The average FVD obtained over iterations is **1444.52** and the best FVD obtained over iterations is **440.40.**

1. We generated videos using generator at regular intervals for a fixed noise.

 Epoch 0 Epoch 1 Epoch 2 Epoch 3 Epoch 4

 Epoch 5 Epoch 6 Epoch 7 Epoch 8 Epoch 9

 Epoch 10 Epoch 11 Epoch 12 Epoch 13 Epoch 14

(Download the word document and Hover over the videos to play, if static)

1. **Conclusion:**

The quality of video Generation can be improved by training the model for a greater number of epochs. We can also use other GAN models for Video Generation such as MoCoGAN-HD, TGAN-F, VideoGPT etc. We may also employ BiGAN to avoid mode-collapse during GAN training. There are other techniques like Appearance Contrastive Learning, Temporal Structural Puzzle proposed by [4] to improve the quality of Video Generation.

1. **References:**

[1] Video Generative Adversarial Networks: A Review [[Link](https://dl.acm.org/doi/pdf/10.1145/3487891)]

[2] Generating Videos with Scene Dynamics [[Link](https://www.cs.columbia.edu/~vondrick/tinyvideo/paper.pdf)]

[3] UCF11 Dataset [[Link](https://www.crcv.ucf.edu/data/UCF_YouTube_Action.php)]

[4] Self-Supervised Video GANs: Learning for Appearance Consistency and Motion Coherency [[Link]](https://openaccess.thecvf.com/content/CVPR2021/papers/Hyun_Self-Supervised_Video_GANs_Learning_for_Appearance_Consistency_and_Motion_Coherency_CVPR_2021_paper.pdf)

[5] Submission materials: [[Drive Link](https://drive.google.com/drive/folders/1qfA483RxFGGiTow2_Kn5_Fie6CmNEUpz?usp=sharing)]

[6] Notebook: [[Kaggle](https://www.kaggle.com/code/sai2021/ggm-project)]