Algorithms and Optimization for Big Data Final Project Report

Probabilistic Future Frame Prediction using convexly optimized Generative Adversarial Networks (GANs)

Group: GriffinClaw RavenPuffs (GCRP)

Anushree Rankawat, Himol Shah, Meet Patel, Harsh Shah, Divya Dass

1. Problem Statement

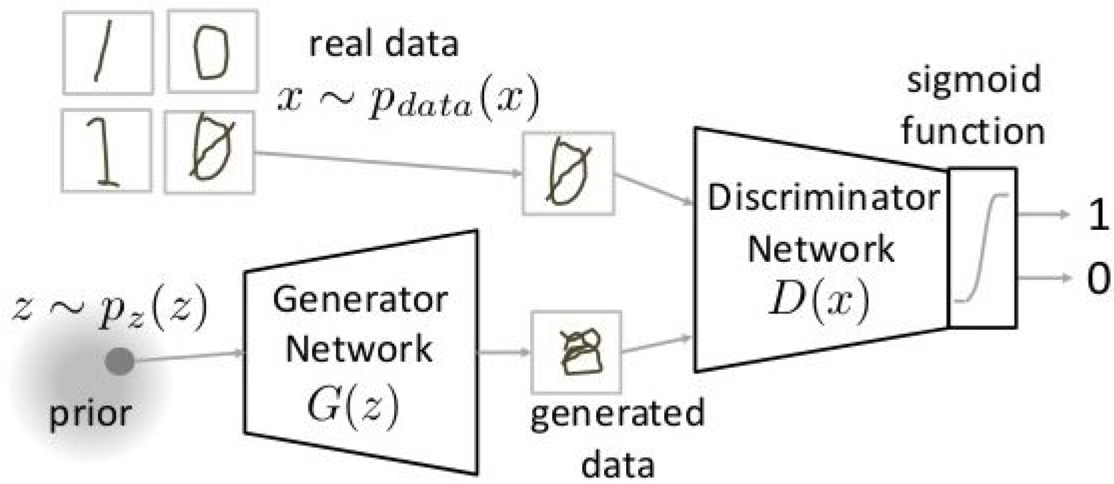
Our project focuses on generating future frames of a video with the help of a window of past frames using the Generative Adversarial Networks. For this scenario, we would like to generate sharper images and reach the equilibrium faster. In order to do so, we adopt the strategy of risk averse players and perform regret minimization in our generator.

Video prediction is a challenging problem because there are various number of ways in which objects and scenes can change. Initially, Generator models lagged behind the accuracy of discriminator models in real world as the images generated were blurry and not realistic enough for real time usage. They gained popularity after the introduction of Generative Adversarial Networks as this research aimed at improving the performance of Generative model using the Adversarial model's discriminatory prowess.

Our work aims at enhancing the sharpness, accuracy and speed of these generator models. We have started with the inclusion of Regret minimization in GANs in both PyTorch and Tensorflow platforms, followed by the inclusion of regret minimization in GANs for the specific application of Video Prediction.

2. Introduction to Generative Adversarial Networks (GANs)

The main idea behind a Generative Adversarial Network is to have two competing neural network models. One takes noise as input and generates samples (and so is called the generator). The other model (called the discriminator) receives samples from both the generator and the training data, and has to be able to distinguish between the two sources. These two networks play a continuous game, where the generator is learning to produce more and more realistic samples, and the discriminator is learning to get better and better at distinguishing generated data from real data. These two networks are trained simultaneously, and the hope is that the competition will drive the generated samples to be indistinguishable from real data.



But, GANs face certain issues as well. While training a GAN, it essentially tries to find the Nash equilibrium of the game that is being played between the generator and the adversary. Gradient Descent can achieve this sometimes, but there is no guarantee of this strategy to work every single time. This raises the question that GANs might be more unstable than their other generator counterparts.

So, to handle this problem, we have tried to include regret minimization which achieves local equilibrium introduced by Hazan et al. [ref] The paper argues that the local equilibrium achieved is similar to the Nash equilibrium and can hence be used for solving our issue.

3. **Regret Minimization**

As per the approach proposed in the paper, we are introduced to an equilibrium state(local minima) which is achieved faster with the help of Regret minimization. By this state, we propose that our generator will be able to learn to produce images closer to real images faster.
We achieve regret minimization by introducing a window over which we take mean of our previous losses, and we then compute gradient over this loss. We update weights of generator by using window loss and supplying this loss over to Adam optimizer.

4. **Video Prediction explanation and why it is an essential problem (Traffic accident detection (speed and accuracy is important), Filling up data loss in videos)**
Video prediction is a field of research which involves generating future video frames with the help of past few frames. It is an essential research area as most of the data on the internet right now comprises of videos and images. There are lots of scenarios where our algorithm could be used to predict future frames, but we would like to explain 2

particular applications in detail which could benefit greatly from the inclusion of regret minimization.

It is essential to forecast future frames for critical applications like autonomous driving. These self-driving cars could use video predictions to prevent accidents by predicting the trajectory of objects before collisions happen. With this prediction, cars could take necessary steps to avoid collision. The tendency to avoid accidents might in turn be better than human instincts, if we increase the speed and efficiency of these methods drastically and hence, speed and accuracy are of paramount importance in these cases. Regret minimization aims at achieving the regret faster than vanilla GANs and hence, is a very important addition to Video Prediction here.

Another application for Video Prediction could be predicting/substituting missing frames in a video. This could be useful for video call conferences where the video lags behind many times and causes interruptions and inefficiencies in the work. This is a non-critical scenario and would require online learning of video prediction.

5. **How can regret minimization help in our application and why is it essential to our application?**
   Regret quantifies the objective of prediction points with small gradients on average. In other words, regret is loss of the generator i.e., the difference between generated output and real input. Regret Minimization is to reduce the time taken by the generator to produce the images as similar as the real images.

   Considering Video prediction, the real inputs are actual video frames and generated images are the predicted future video frames by generator. The generated frames are compared with ground truth by discriminator from the actual video and regret is computed. The generator weights are updated by using Adam’s Optimizer that considers the computed regret along with the adaptive learning rates for all nodes and updates the generator weights in such a way that the generated output is as similar as the real inputs. This process is repeated for certain number of times to achieve the minimum possible generator loss. At the end when generator is totally trained, the probability of the discriminator to predict any frame as real or fake would settle at 0.5.

6. **Video Prediction paper explanation (a few lines) (How they modified normal GANs and how it helps in making the video frames generated to be sharper)**
   There are broadly two problems considering video representations: i) Action Classification ii) Video Prediction. The basic idea for using GAN for video prediction is: generator G tries to produce a video to maximally fool a discriminator D whereas the discriminator tries to distinguish between real and fake videos. The fundamental problem is to learn how scene transforms with time. This is significantly different from problem of Image Reconstruction as the ability of the model to generate accurate non trivial representations based on previous frames is considered.

The paper, Deep Multi-scale Video Prediction beyond Mean Square Error which we referred for approaches of video prediction does adversarial generation that two networks – a generator G and a discriminator D – to improve the sharpness of generated images. Given the past four frames of a video, the generator learns to generate predictions for the next frame. Given either a generated or a real image, the discriminator learns to correctly classify it between generated and real. The two networks compete against each other that is the generator attempting to fool the discriminator into classifying the generated images as real. This forces the generator to create frames that are very similar to the real frames

7. Inclusion of Regret Minimization in Vanilla GANs and Video Prediction

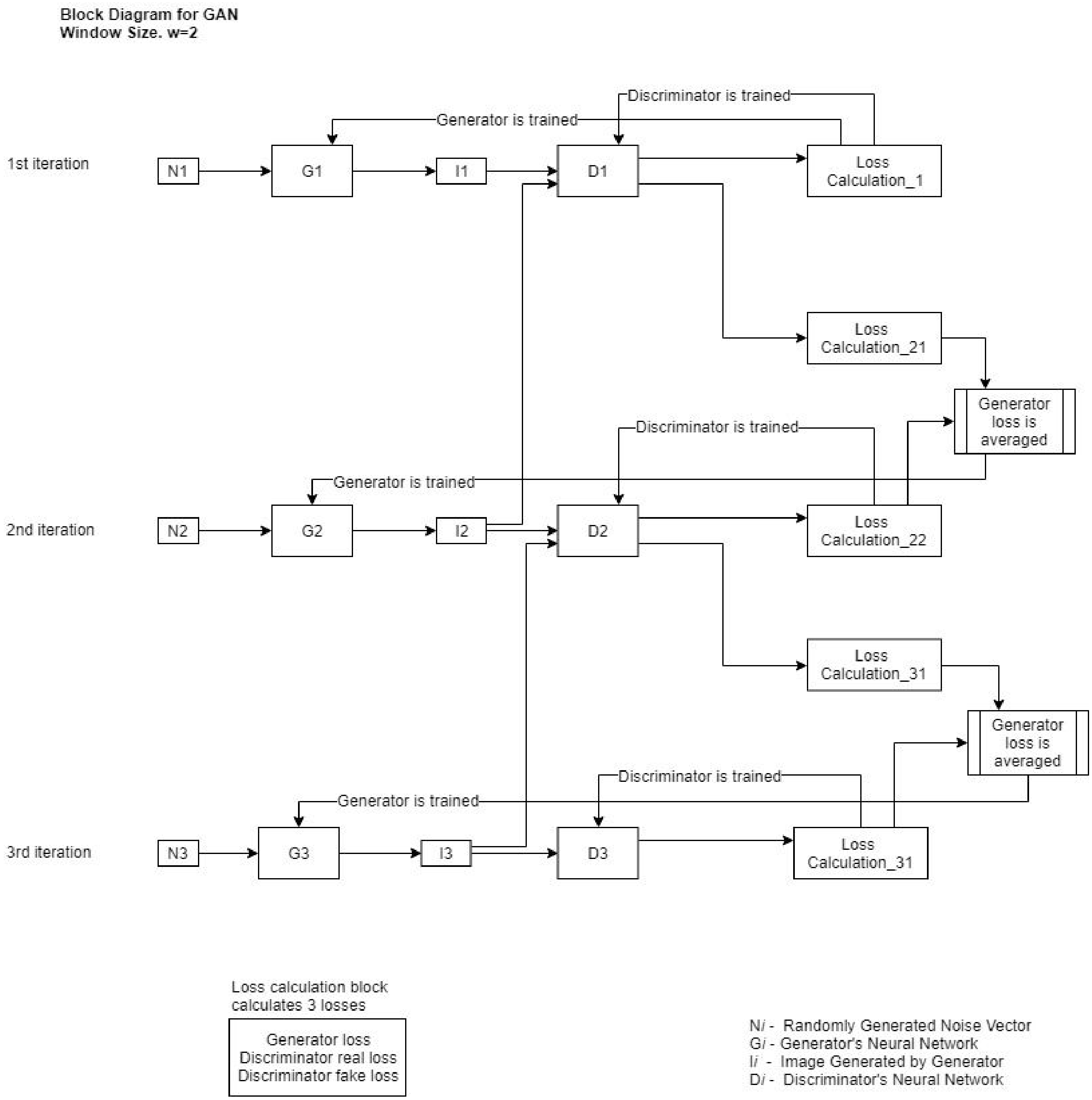
Vanilla GANs

Here, we introduced window into the loss calculation on simple vanilla GANs. We have updated weights by first finding average of losses over the window size, and then we optimized our weights by using this mean loss by using Adam optimizer.

1. We first tried introducing window by:
    - Adding Placeholder for the generator losses.
    - Introducing a method to calculate mean loss during the graph initialization.
    - Giving this mean loss into the Adam optimizer.
    - During each epoch, we passed previous two losses(considering window size = 2) in “feed_dict” for further calculation.
    - **Problem with this approach** : When we obtain the loss of generator in each epoch, we are changing the tensor type of generator loss to numpy array type. Because of this, the connection of generator loss with the graph broke down.

2. Actual runnable approach for sliding window:

System Architecture for window technique:



Video Prediction

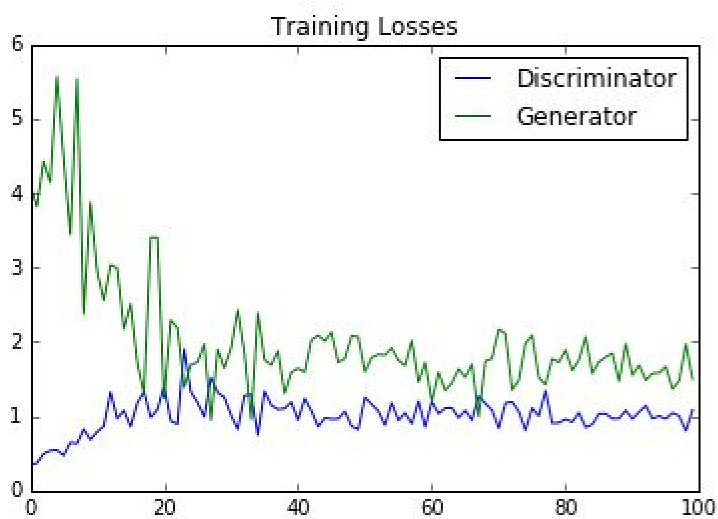
In this approach, we have a discriminator model which discriminates whether the generated images are fake or not. Generator has the task of generating real images.

1. Given these two models, we have introduced a window list for holding the losses of the adversary.
2. We have introduced method to calculate average loss in the generator model.
3. Then, we coded algorithm-1 where the weights are being updated till the average losses become less than our threshold (tolerance/window size).
4. We have calculated adversarial loss for the current generator's weights (which we are trying to update). This loss is being added to the windowed loss.
5. This loss is being averaged. This windowed loss, combined with other losses is being back propagated.
6. Windowed is being passed with the help of a placeholder, which allows these values to be given after computation in our code snippet for algorithm-1.

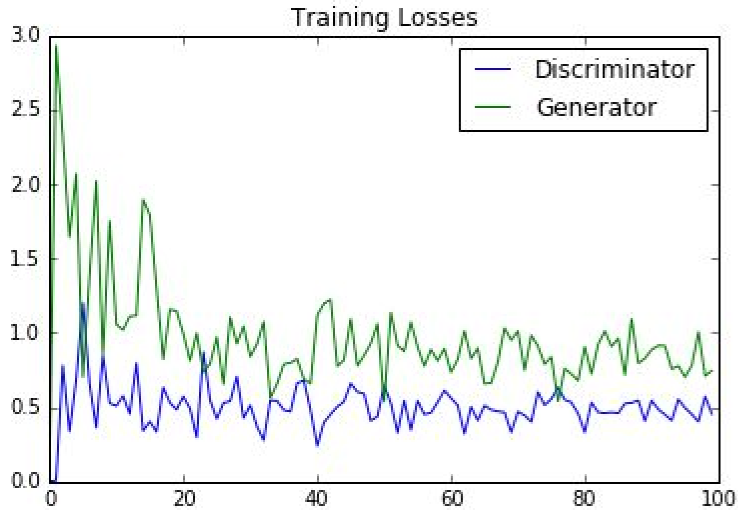
8. Results of GANS with Regret Implementation and window size.

A. Loss Graph for MNIST dataset:

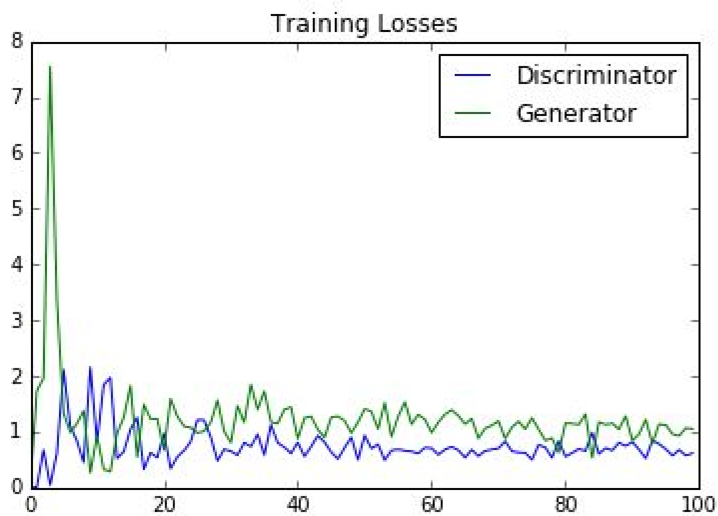
1. Loss for Generator and Discriminator over the epochs (window = 1)



2. Loss for Generator and Discriminator over the epochs (window = 2)

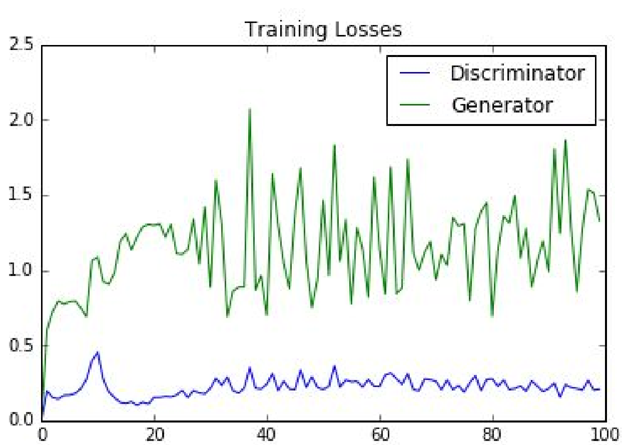


3. Loss for Generator and Discriminator over the epochs (window = 3)

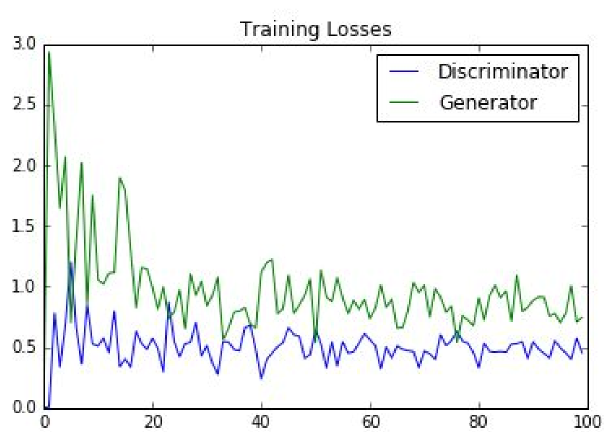


4. Gradient Descent Optimizer vs Adam Optimizer for window size = 2

Gradient Descent

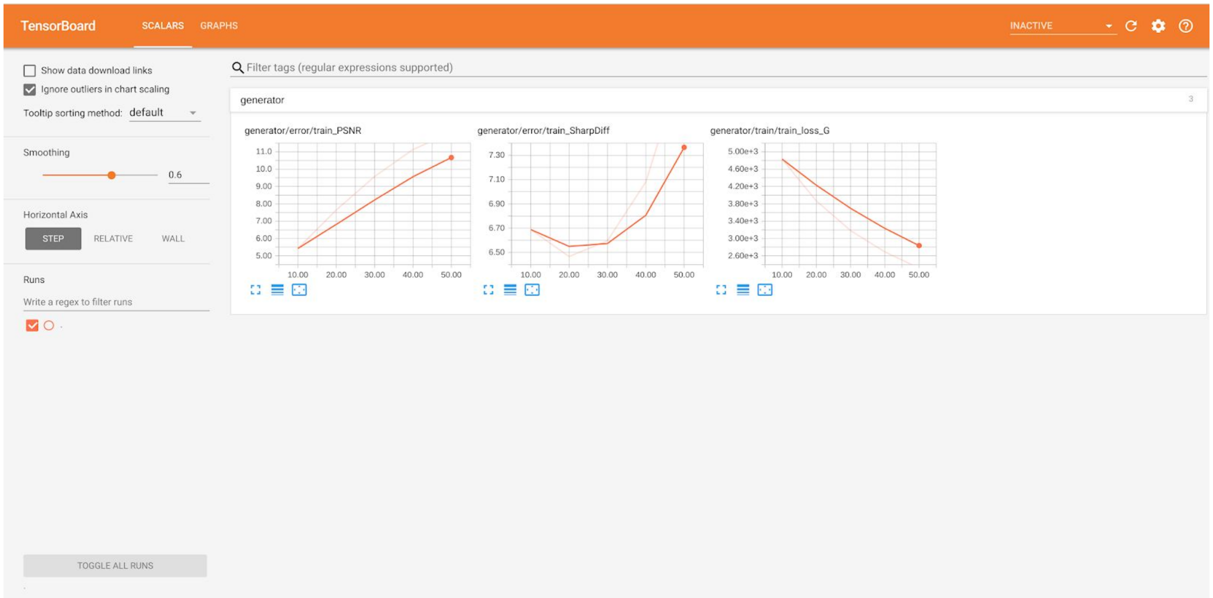


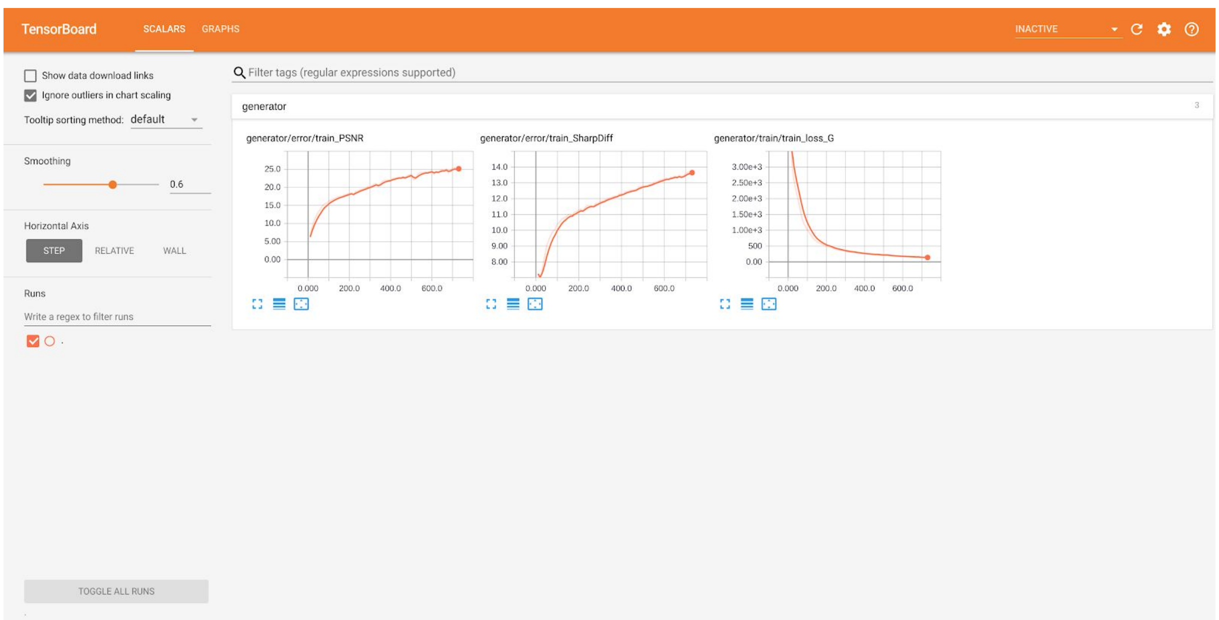
Adam Optimizer



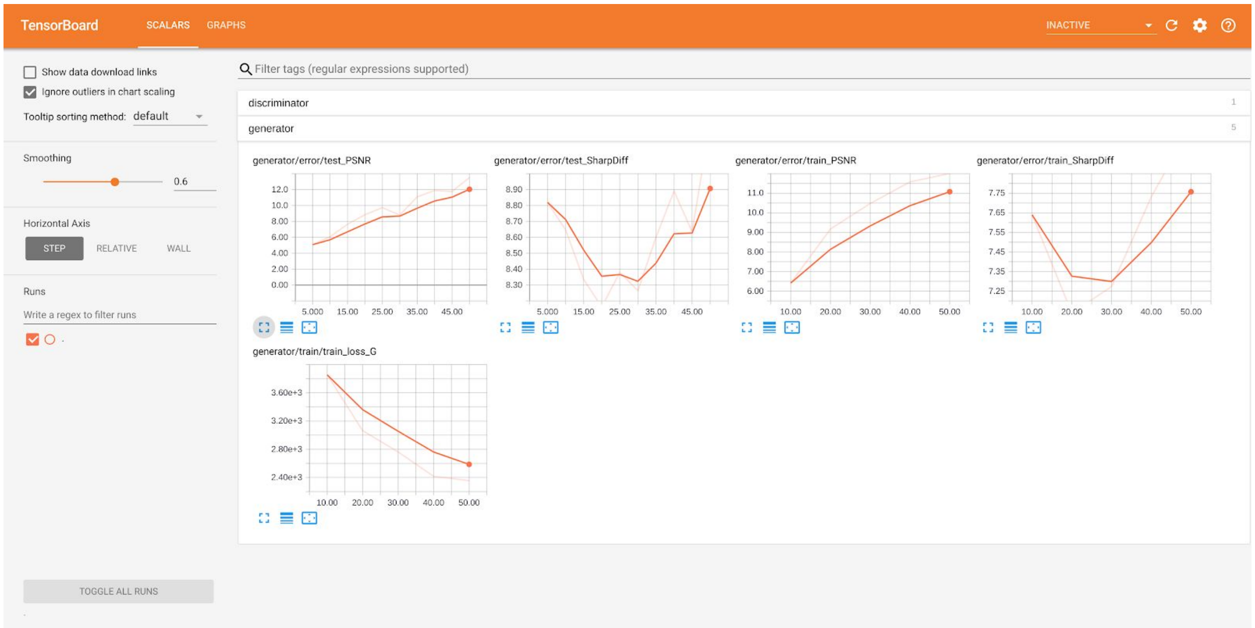
9. Graphs for Video Prediction

Changed Loss Function Graphs





Graphs with Normal GANs:



10. Conclusions and Future Work

Our algorithm works well for finding Local Equilibrium (similar to Nash Equilibrium) for Generative Adversarial Networks with the inclusion of Regret Minimization. It helps us reduce the time taken for convergence and the accuracy of the images generated drastically.

Future work would entail making this algorithm online by not only predicting future frames but also including them in the training data as stream data. This would help us in applications like accident prediction and avoidance in driverless cars and filling up for data losses in video conferences. Another future work would require us to carefully fine-tune the hyperparameters here, like the tolerance and window size for optimal output.