Probabilistic Future Frame Prediction using Generative Adversarial Networks (GANs)

Group: GriffinClaw RavenPuffs
Anushree Rankawat (201501007)
Meet Patel (201501074)
Harsh Shah (201501096)
Himol Shah (201501098)
Divya Dass (1744001)

Problem Description

- Introduction to the problem.
  - Video Prediction: Generate future frames using previous frames.
- How did we approach the problem using GANs?
  - One network to generate images, another to classify
- Why do we use regret minimization?

GAN Implementation for Regret Minimization

- Sliding Window time average of loss functions **f** is calculated as:

$$F_{t,w}(x) \stackrel{\text{def}}{=} \frac{1}{w} \sum_{i=0}^{w-1} f_{t-i}(x).$$

- We have minimized over all regret for window size **w** over **T** iterations.

$$\mathfrak{R}_w(T) \stackrel{def}{=} \sum_{t=1}^{T} \|\nabla_{\mathcal{K},\eta} F_{t,w}(x_t)\|^2.$$

- Results corroborate the convergence obtained by sliding window regret implementation.

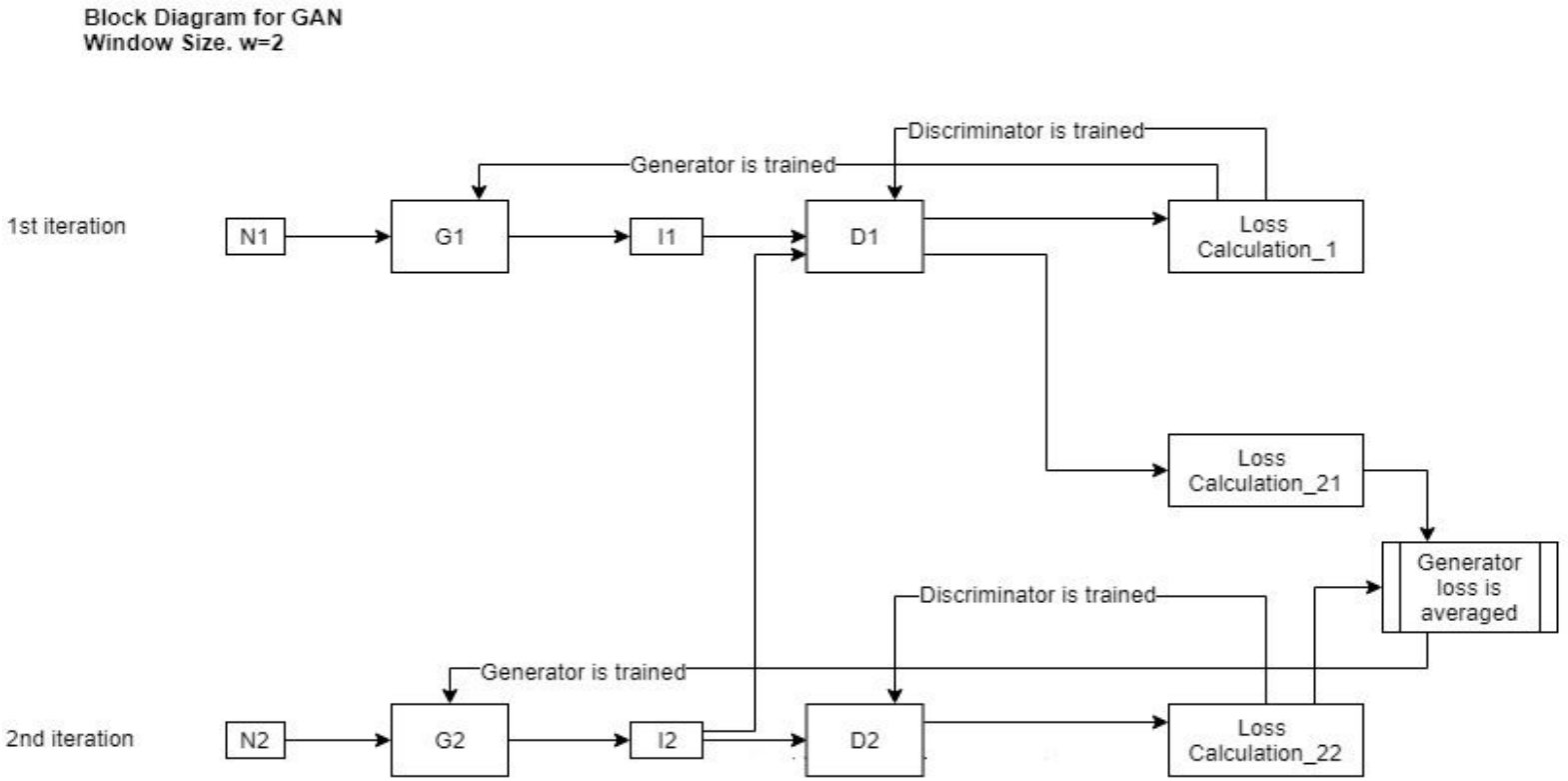
Primary Approach

1. Introduce placeholder for intermediate losses.
2. Save losses of previous iterations.
3. Calculate the mean of all losses.
4. Optimize the mean loss using Adam’s Optimizer.

Problem faced:

The mean losses were saved in a NumPy array. Therefore, connection of mean loss with the computation graph of tensorflow did not hold. Thus, the mean losses were not passed to Adam’s Optimizer.

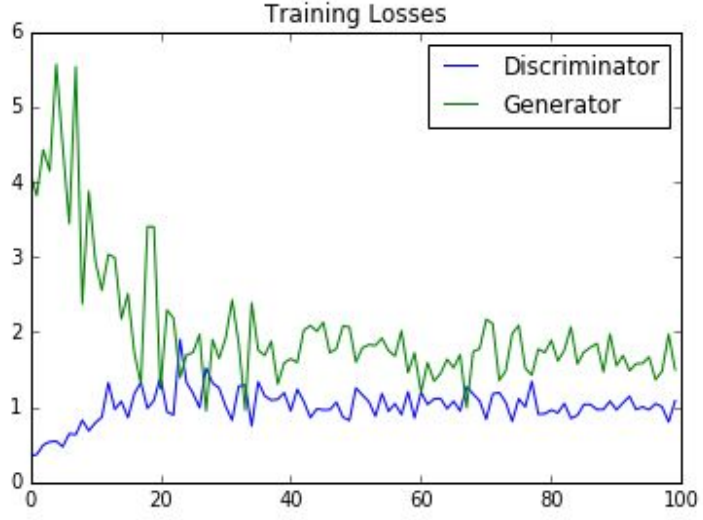
Final Approach



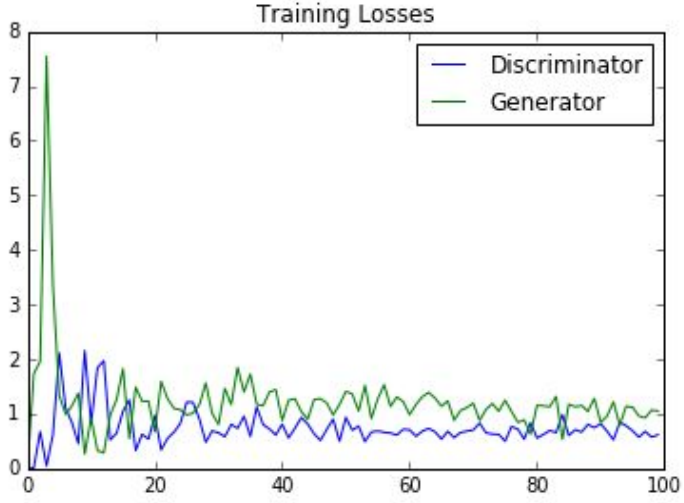
Results: Implementation of GANS in Tensorflow

Generator and Discriminator Loss with and without sliding window. The loss clearly converges when more than one past losses are considered.

Without Sliding Window (w = 1)

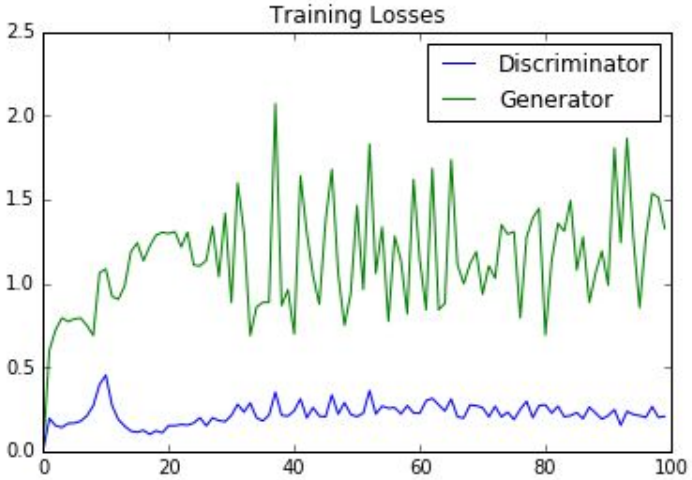


With Sliding Window (w = 3)

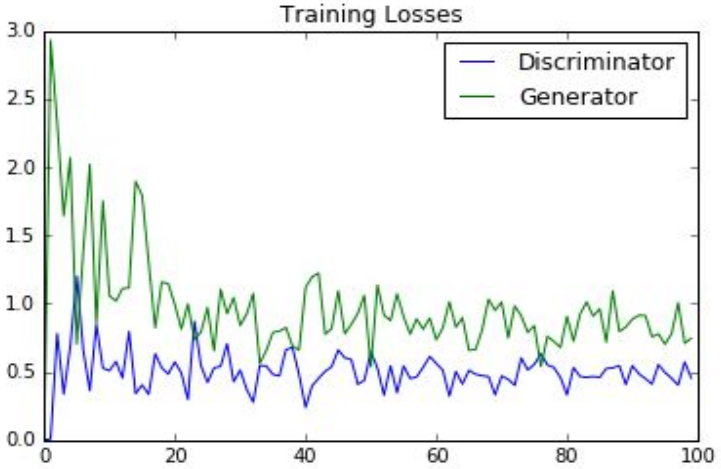


Analysis of Gradient Descent and Adam's Optimizer with window size w = 2

With Gradient Descent



With Adam's Optimizer



Approach for Video Prediction using optimized GANs

1. In the current session, loss given by discriminator on generated images is calculated.
2. A list is maintained for adversarial loss outside the session (in Python) to which discriminator loss for current iteration is added.
3. Average loss function calculated offhand in python and passed to the generator using placeholder (transferring in Tensorflow session)
4. Generator trained in accordance with Algorithm-1 and weights are updated in the training session.

Learning rate: 0.02 (discriminator)

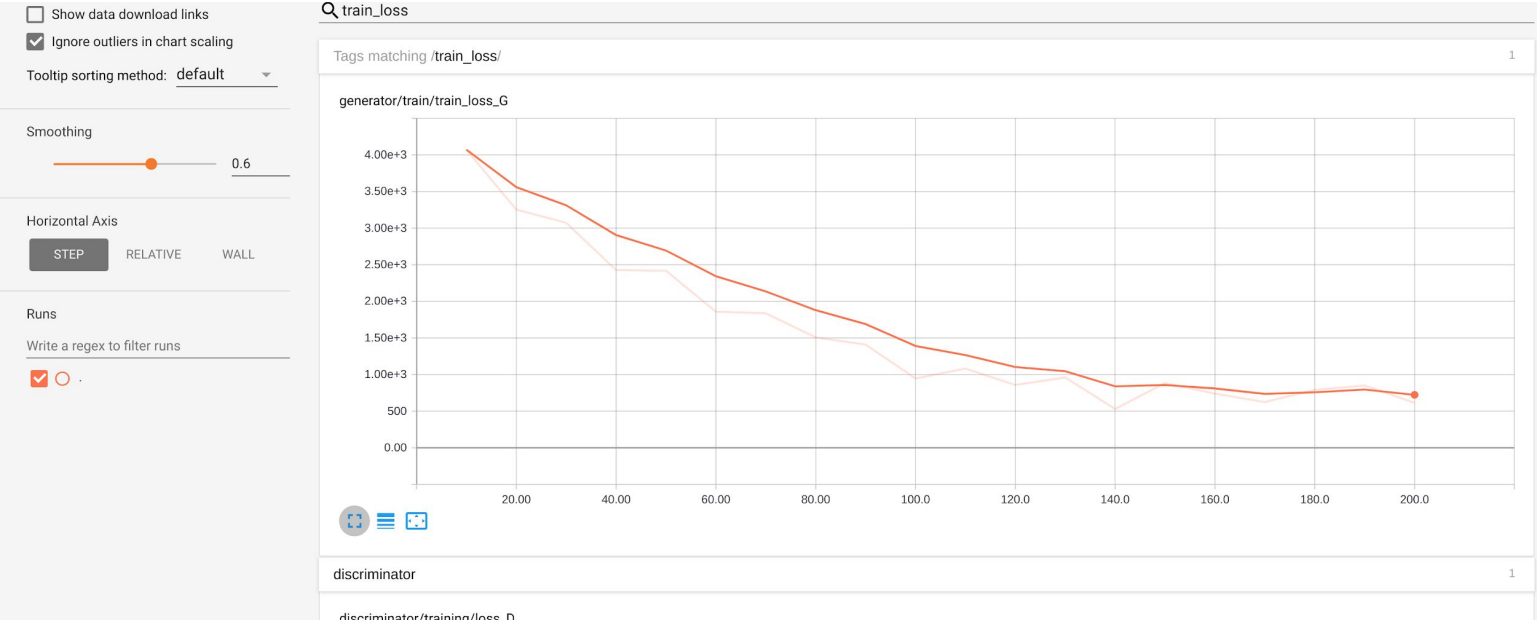
Learning rate : 0.00004 (Generator)

Window size: 20

Tolerance: between 40-50

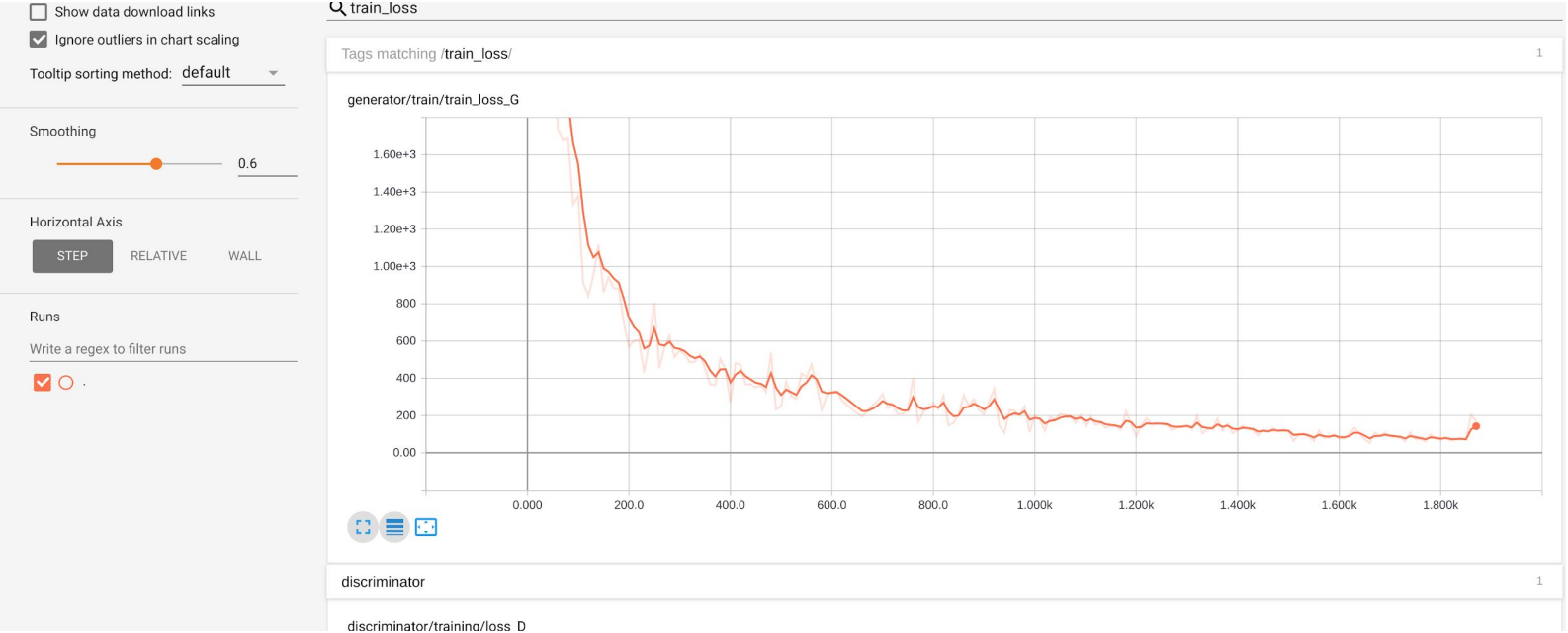
Dataset is PacMan

Video Prediction Results: Normal GANS



Results for Training Loss in Generator for 1.8k steps and 200 iterations

Video Prediction Results: GANs with Regret Minimization



Results for Training Loss in Generator for 1.8k steps and 200 iterations

Images generated by the original GANs

Ground Truth

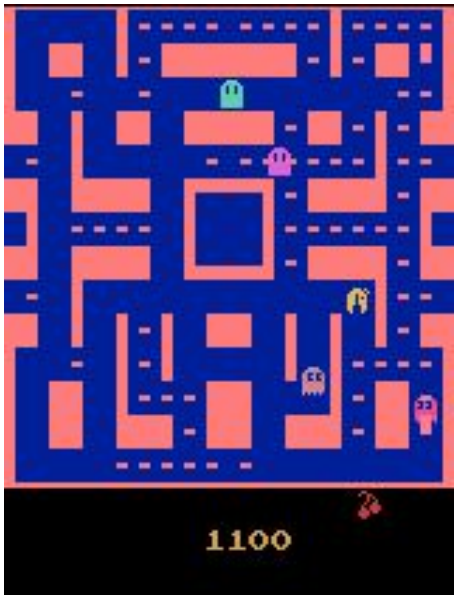
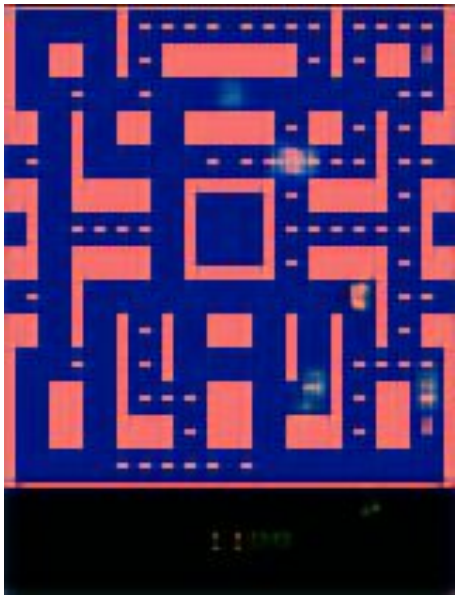


Image generated



Images generated by the updated GANs

Ground Truth

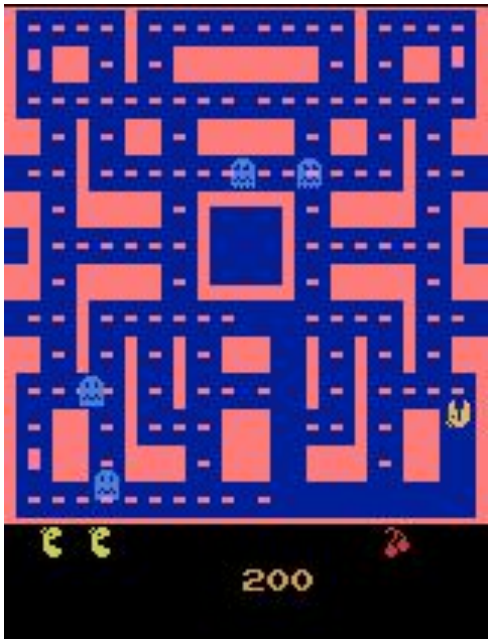
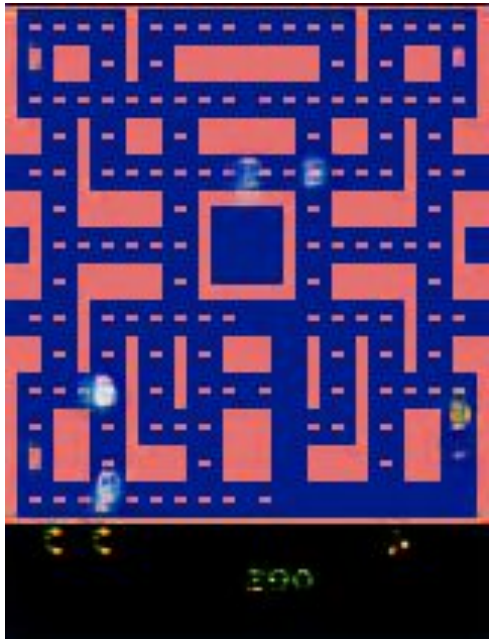


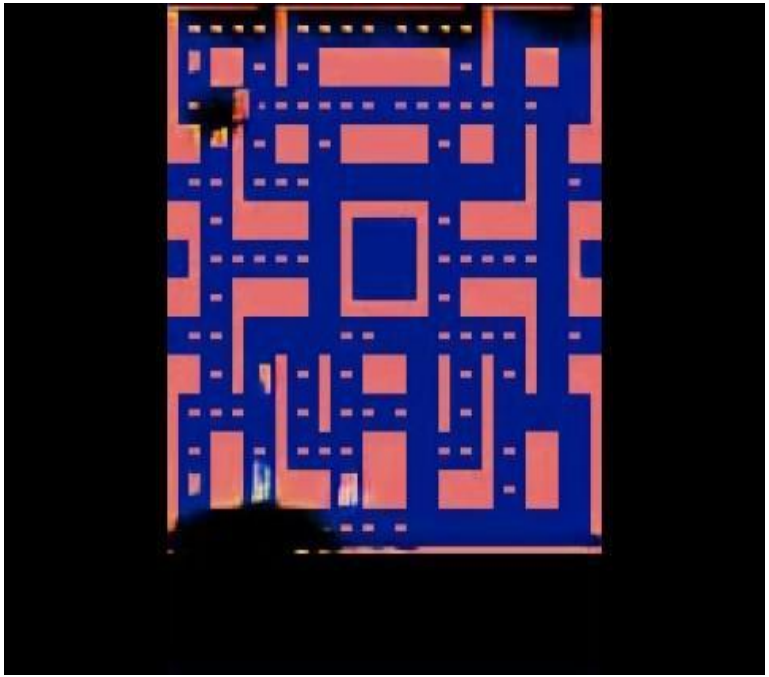
Image generated



Videos for Prediction

Normal GAN prediction

Optimized GAN prediction



Challenges Faced

- Understanding and implementation of GAN on MNIST dataset.
- Understanding actual regret minimization algorithm.
- Basic understanding of optimizers and activation functions.
- Understanding Complex Tensorflow structure and getting familiar with the coding and debugging style in Tensorflow.
- Implementation of regret minimization algorithm in Vanilla GANs and GANs for Video Prediction.

Conclusion

- Video Prediction in GANs using Regret Minimization are shown to perform drastically well and converging faster than normal GANs.
- For 200 iterations and 1.8K step sizes,

  Training Loss for Generator for Normal GAN: 2000

  Training Loss for Generator for GAN using Regret Minimization: 100

  As we can see, the loss drops down drastically and is more efficient than normal GANs.

- The video generated for our model is clearly better and the frames don’t fade away easily

Future Scope

- Implementing Sliding Window in MNIST dataset supports the fact that convergence rate increases with the inclusion of regret minimization.
- Tuning the hyper-parameters, window size and tolerance for optimal results.
- Investigation of why the video frames fade away.
- Making the video prediction algorithm online by using the incoming stream video as training for GANs as well.

References

1. Hazan, Elad, Karan Singh, and Cyril Zhang. "Efficient Regret Minimization in Non-Convex Games." *arXiv preprint arXiv:1708.00075* (2017).

2. Code for MNIST Tensorflow implementation: https://towardsdatascience.com/gan-introduction-and-implementation-part1-implement-a-simple-gan-in-tf-for-mnist-handwritten-de00a759ae5c

Thank you!