Quantum Implementation of GANs and comparison with its classical counter

Presented by:
Samya Sunibir Das (1911563042)
Nazmul Hasan (1911742042)
Sabiha Hossain (1911017042)

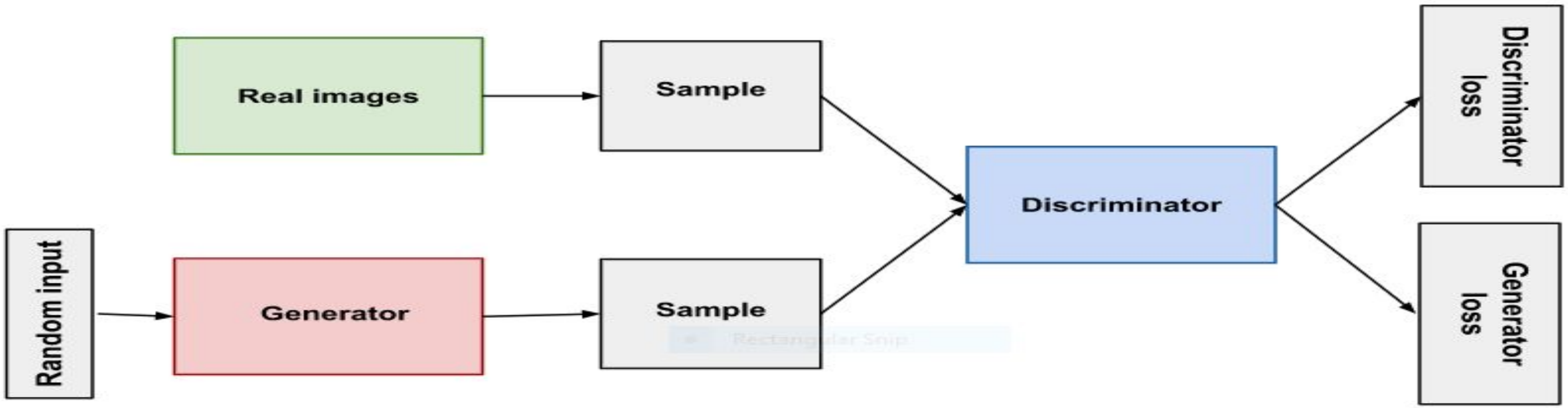
Introduction

- Generative Adversarial Networks, or GANs for short, are an approach to generative modeling using deep learning methods, such as convolutional neural networks. Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset.

- qGAN implementation learns and loads probability distributions into quantum states. More specifically, the aim of the qGAN is not to produce classical samples in accordance with given classical training data but to train the quantum generator to create a quantum state which represents the data's underlying probability distribution.

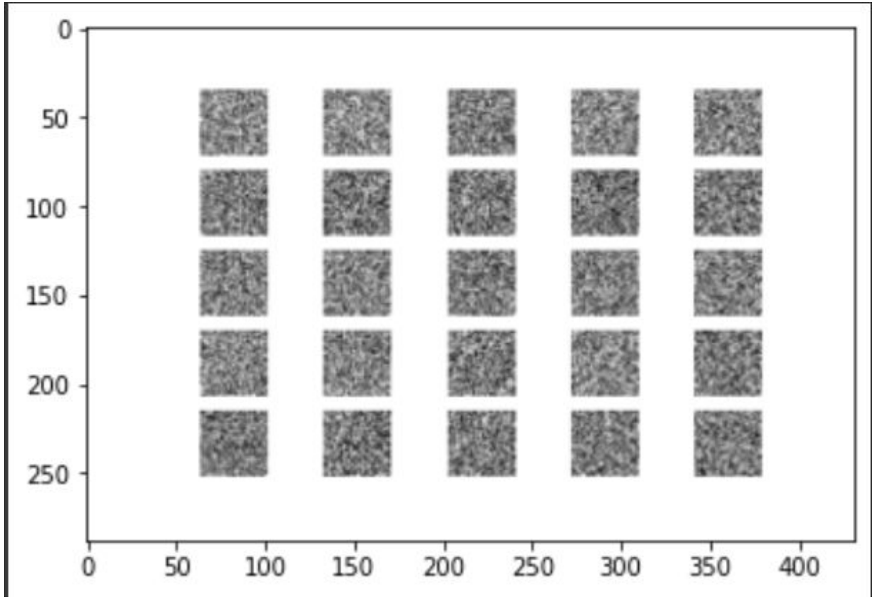
GAN in Depth

- A generative adversarial network (GAN) has two parts:
  The generator learns to generate plausible data. The generated instances become negative training examples for the discriminator.
  The discriminator learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.

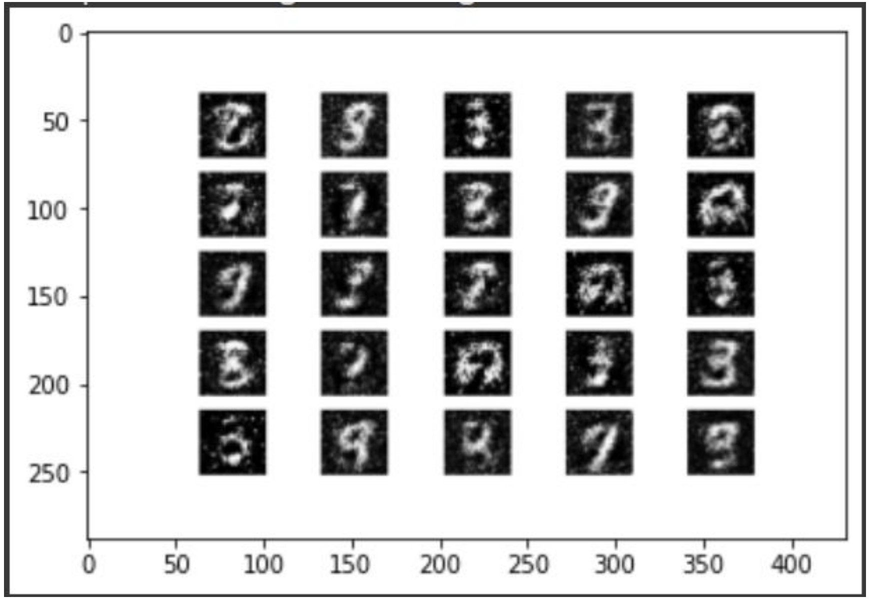


Classical GAN applied on the MNIST dataset

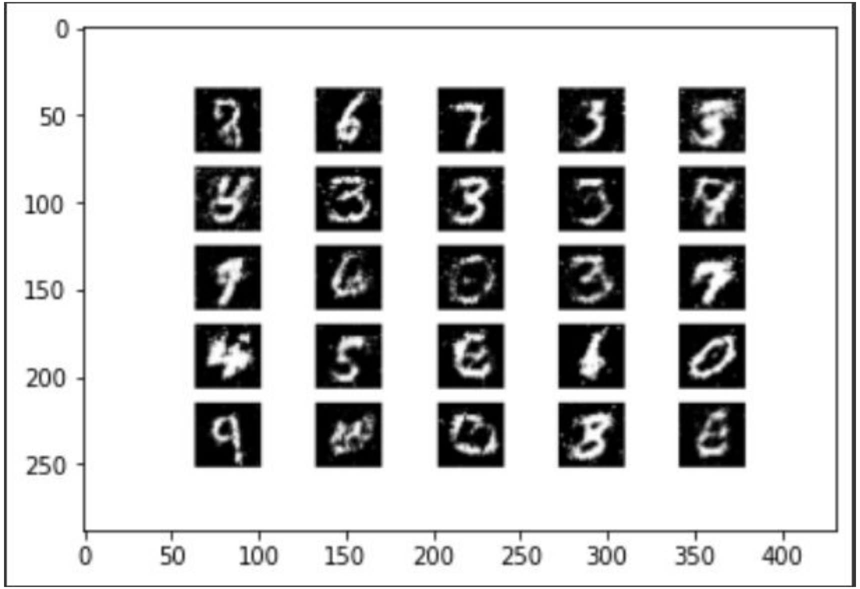
At epochs = 0



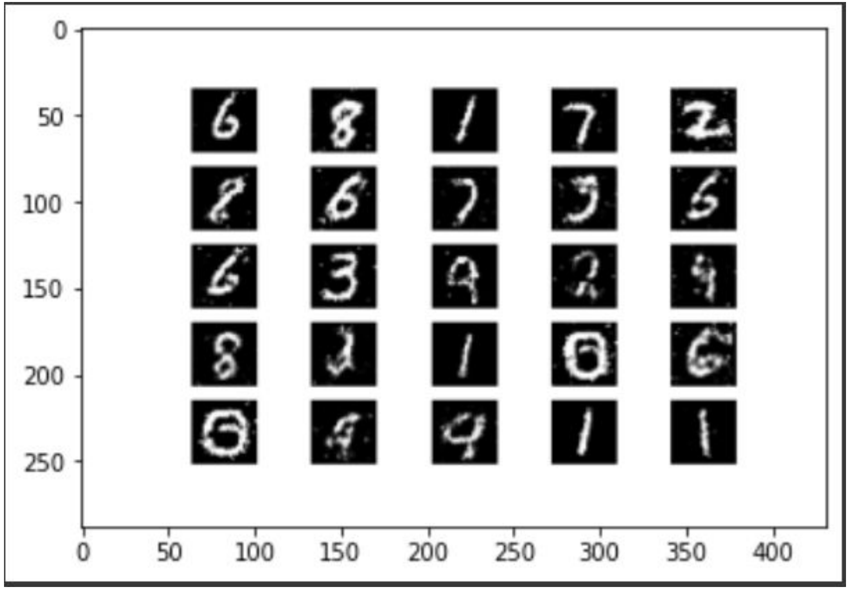
At epochs = 1000



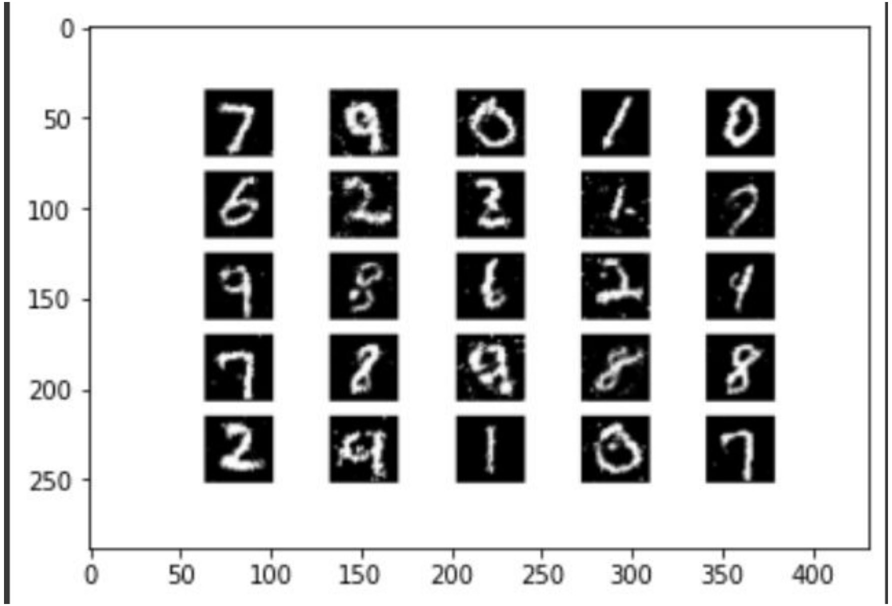
At epochs = 5000



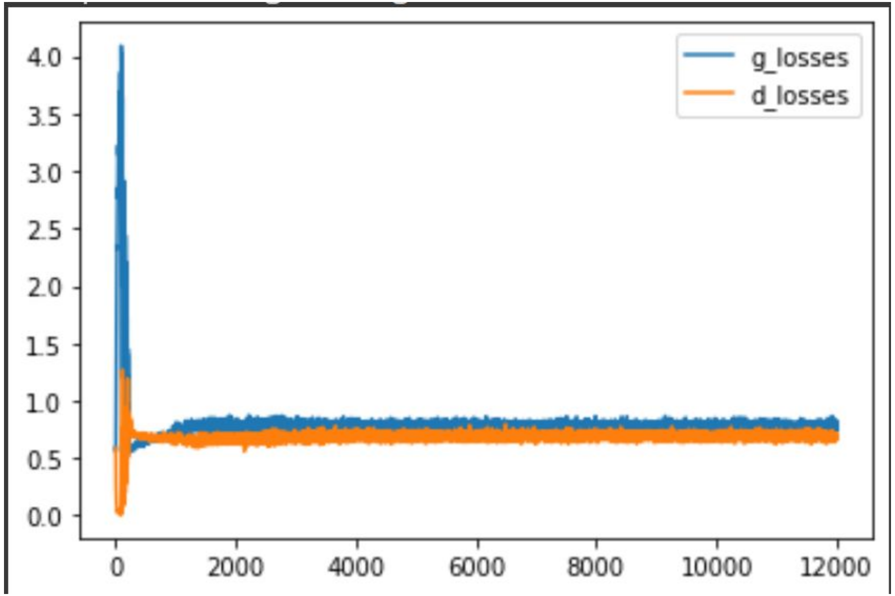
At epochs = 10000



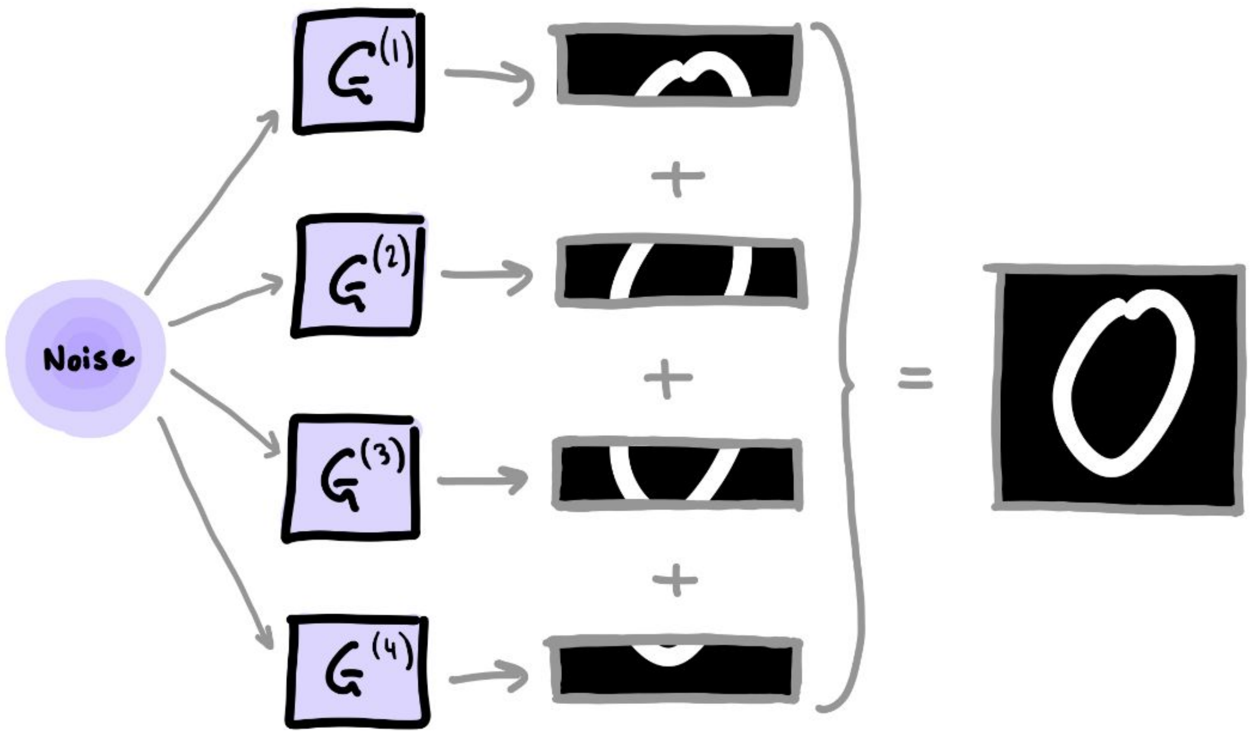
Finally at epochs = 11800



Generator vs Discriminator Loss Curve

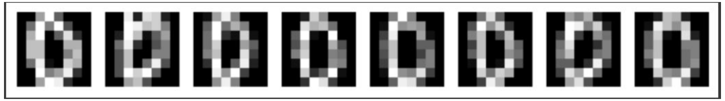


Quantum GAN implementation: Patch Method

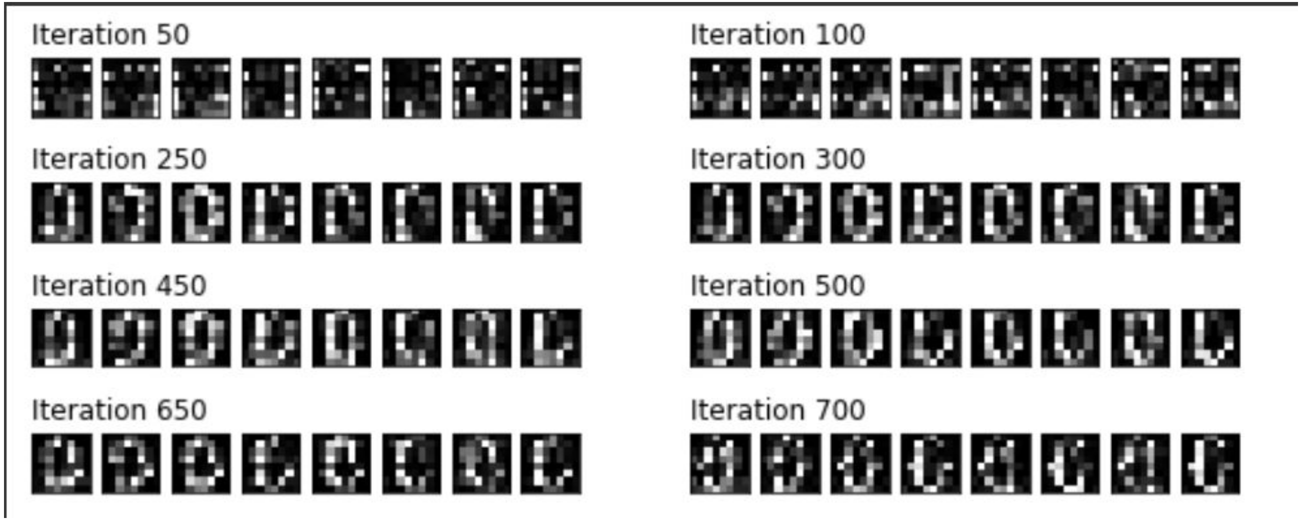


Results: Patch Method

Sample 0s in the dataset

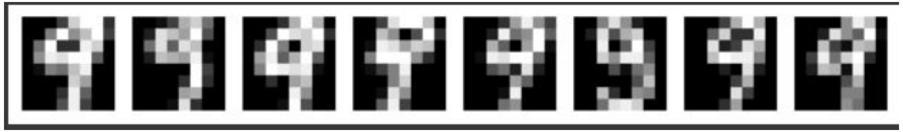


Generated Results for 0s for 700 epochs

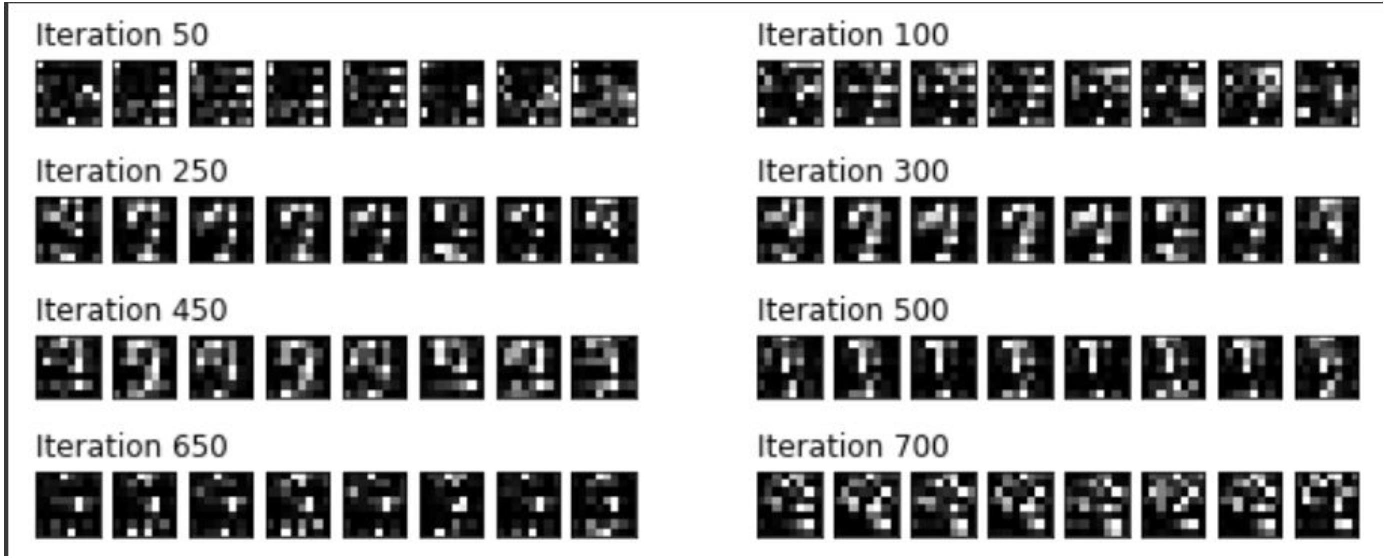


Results: Patch Method

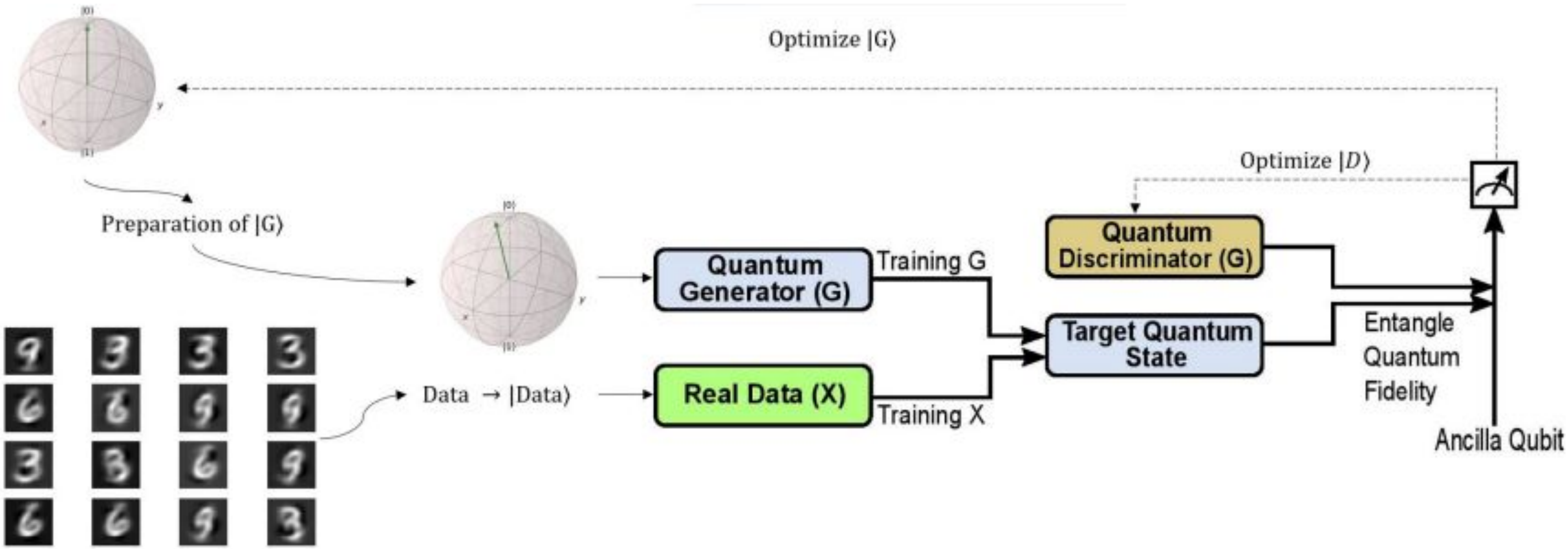
Samples for 9s in Dataset



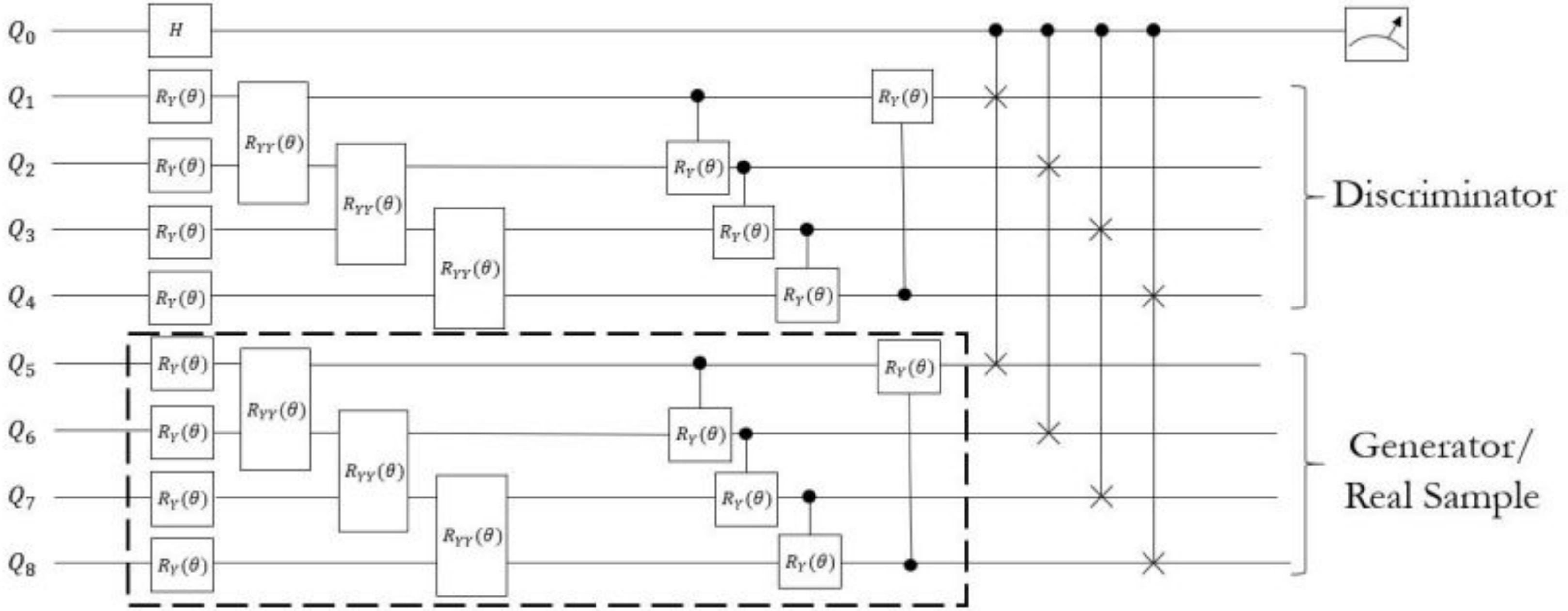
Generated Results for 9s until 700 epochs



Quantum State Based



Structure



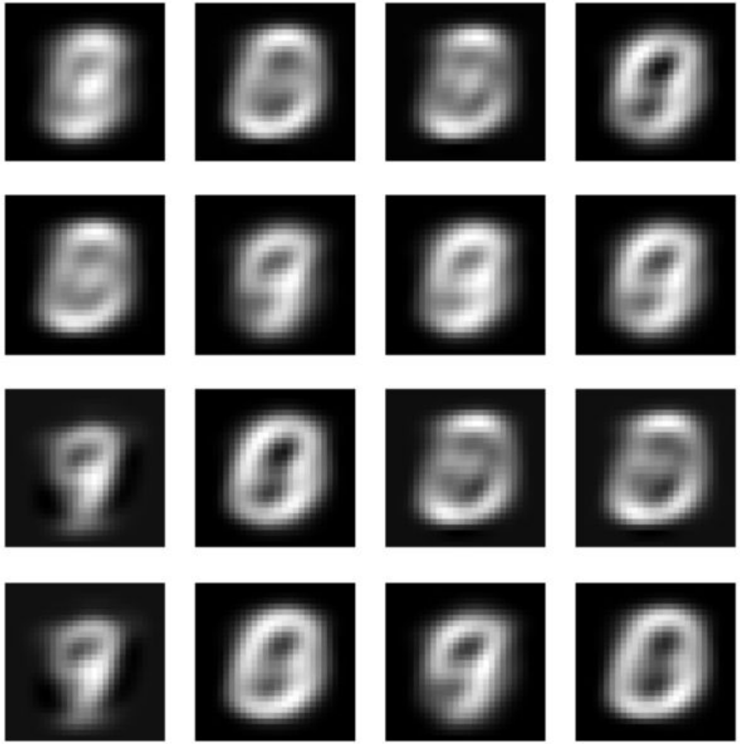
Q_0
Q_1
Q_2
Q_3
Q_4
Q_5
Q_6
Q_7
Q_8
H
$R_Y(\theta)$
$R_{YY}(\theta)$
Discriminator
Generator/
Real Sample

MinMax Loss

- Discriminator: maximize log D(x) + log(1 – D(G(z)))

- Generator: minimize log(1 – D(G(z)))

Results



Epoch 1

Epoch 20

Results

Epoch 50

Epoch 70

THANK YOU!